Achieving High Availability and Fail Over for Subversion with *LVM snapshot*

Rudhuwan Abu Bakar
Software Engineering Lab
MIMOS Berhad, Kuala Lumpur, Malaysia

Nor Izyani Daud
Information Security Lab
MIMOS Berhad, Kuala Lumpur, Malaysia

Abstract- High Availability (HA) and Fail Over (FO) are must have features in any Disaster Recovery (DR) Plan for enterprise today. As in the case of setting up Subversion (SVN)'s FO and HA, it requires high Total Cost of Ownerships (TCO) due to the necessity of add-on features such as multisite capabilities. Such feature is needed to achieve proper DR infrastructure where an instantaneous data replication and synchronization from primary site to the DR site is continually executed. Therefore, an alternative solution with minimal investment and low Recovery Time Objective (RTO) is needed to be explored. Taking those factors into consideration, customization of existing open source technology, would be the most effective solution in order to fulfill the HA and FO requirement for SVN system setup. The proposed alternate solution must include a fast replication mechanism in order for DR site to function as expected and also lower the TCO by eliminating access license limitation. Additional customized functionalities are needed since SVN's built-in mirroring and back-up functionalities; svnsync and svn hotcopy, are not designed to achieve a complete HA solution. The main reason is because svnsync is just a one-way-synchronization tool which creates a READ-ONLY repository on the DR site, while svn hotcopy is a painstaking task that may take hours to complete one cycle of backup depending on the data size and disk write speed. This paper will discuss the chronology of approaches taken and tools that have been experimented with to find the best alternate solution. In the end, the findings show that Logical Volume Manager (LVM) snapshot provides the best solution with the least setup complexity and at minimal cost. Moreover, findings from various experiments and approaches will be shared and a recommendation is proposed which may be beneficial to other individuals or enterprises that are considering taking the same initiatives.

Keywords – fail over, high availability, logical volume manager snapshot, subversion

I. INTRODUCTION

Revision control also known as version control and source control is the management of changes to documents, computer programs, large web sites, and other collections of information [1]. Most of the software development companies would have already established a Software Configuration Management (SCM). SCM is one the most important aspects of Software Development Life Cycle (SDLC). One of the areas that SCM has to deal with is software tool for versioning control. There are lots of tools that can be deployed for this purpose, for example ClearCase [2], CVS [3], etc. However, the paper will only discuss on Subversion (SVN [4]) since it is the current source configuration management tool in the organization.

SVN is a free or open source version control system [5]. It is widely used to maintain current and historical versions of files such as source code, web pages, and other documentations. Besides that, SVN can be used to archive old versions of files and directories, resurrect them, and examine logs of how they have changed over time. It is also used to manage documents (usually over a network) and to keep track of document changes. It supports few network protocols such as HTTP and can be configured to run on top of an Apache webserver with LDAP-enabled. Each user will be assigned a pre-determined access privilege that is either READ/WRITE or READ-ONLY to access specific repositories.

Nowadays, Fail Over (FO) and High Availability (HA) are must have features in any system architecture setup especially in Disaster Recovery implementation where there is cost associated with it. SVN, by default, does not have a built-in feature to support such implementation. The usual solution is coming from commercial products such as WANDisco which provides multi-site solution. Usually implementing such features is deemed as high investment. On the other hand, there is always an open source solution that can achieve the same objective but with lower Total Cost of Ownerships (TCO). Of course, there are disadvantages when deploying alternative solution as well which the paper will discuss in details.

This paper will be discussing on different approaches taken including their disadvantages and actual outcomes of those approaches in order to fulfill the requirement. Lastly, the paper will present a recommendation on the best approach that may benefit others that are considering the same initiative. In the end, it can be demonstrated that even without spending a fortune; an enterprise level solution can be achieved with open source software technologies.

II. PROBLEM STATEMENT

The current SVN infrastructure setup is facing performance issues particularly on daily backup which is very slow and consumes lots of disk space. Due to this, user's satisfaction is very low. Furthermore, the infrastructure potentially has a single point of failure since there is no FO and HA supports capabilities in place.

Because of that, the organization top management has proposed a list of new system requirement. The system has to have the following features in order to properly support day-to-day software development activities:

- It should support HA where users will always able to access the system at all times.
- It should support instantaneous FO where in the event of system failure, the system will run from the remote site and the switch over will be transparent to the users.
- It should have a proper backup to the remote site. The backup should be run daily and replicated to the remote site.
- The backup should not consume lots of time and server resources (CPU, memory and disk space) which affect daily software development activities.
- It should be able to execute a fast data replication, preferably at every hour, to the remote site on daily basis.
- The system Recovery Time Objective (RTO) should take less than one (1) hour where in the event of total system failure or crashes; the system should be operating normally within one (1) hour.

a. Current Limitation

Clearly the current system, which uses the default SVN installation setup, would not be able to meet the requirement as set by the management. Moreover, deploying a commercially available solution is not an option as there is no budget allocated for such implementation. Therefore, there is a need to devise an implementation to support HA and FO within the current infrastructure with a minimal cost only.

High Availability (HA) in SVN basically means users are able to access the repository at any time continuously while Failover (FO) means the access and the functions are assumed by a secondary system at the remote site when the primary system becomes unavailable through either failure or scheduled down time. The main prerequisite for HA and FO to work properly for SVN setup, is to make sure data in both primary site and remote site is always in-sync. This is to make sure, in the event of system failure, users are still able to access the most current data.

There are two built-in functionalities that are originally thought can be used to setup HA and FO for SVN: s*vnsync* and *svn hotcopy*. However, these tools definitely have their disadvantages and lack required features that are needed to fulfill the HA and FO requirement as outlined by the management. Their limitations can be summarized as follows:

- *svnsync* is a one way synchronization tool; thus it creates a READ-ONLY repository in the remote site.
- *svn hotcopy* is a backup tool that creates an exact duplicate of the current data state in the primary site. However, the time taken to complete one cycle of backup depends on the size of the repository.

Each of those functionalities and limitations will be discussed in the next section. Using these default SVN built-in functionalities, it was not be possible to fulfill the FO and HA requirement without adding extra functionalities to them. Of course, there are ready-made solutions, for instance provided by WANDisco, which provides a multisite solution with instantaneous data replication. But it comes with lingering user license cost and that is not the alternative which management agreed to proceed.

III. APPROACHES, EXPERIMENTS AND RESULTS

In view of monetary constraint, various approaches were studied and experimented in order to find the best solution for HA and FO for SVN infrastructure. In general, the approach mainly uses existing SVN functionalities and open source software technologies. Moreover, the results between those approaches are analyzed and compared in order to choose the best implementation that is able to meet the requirement laid by the management. The approach must satisfy the followings in order to be considered as the best HA and FO approach.

i. Users must always able to access the most current data in the repository in the event of FO occurs

ii. Users must always able to update their changes into the repository

iii. Users must always able to access the repository at all time without any service interruption

a) Approach One – using *svnsync* to mirror data to the remote site

In order to satisfy the above scenario as in Section 3.0, both primary and remote sites must be configured as active-active mode. However, as discussed in Section 2.1, based on the way *svnsync* functions; an active-passive mode is the only possible setup with this approach. This is because the remote site will be dedicated as READ-ONLY mirrored repository. This approach is deemed as FAIL as it definitely failed to meet the requirement.

Out of the three criteria listed in Section 3.0, this approach only satisfy criteria (i) and (iii).The findings from this approach can be summarized as follows:

i. *svnsync* a one-way synchronization tool that creates a READ-ONLY remote site.

ii. Due to (i), in the event of primary system failure, users can still access the repository but users by design are not allowed to update any change into the remote site. This breaks the FO requirement even though it is still HA-compliance.

iii. *svnsync* will occasionally run into locking issue if there is an interruption to the network while data synchronization is being executed. This caused data inconsistencies between the primary and the remote site.

iv. *svnsync* require one script per repository. Currently, there are more than seventy (70) repositories in the primary site. Therefore, seventy (70) separate *svnsync* scripts needed to be executed. In order to make sure both sites have consistently latest data, all of these scripts will need to run simultaneously at every hour. These will eats up server resources; CPU and memory; which in turn, affect the application overall performance and increase users' aggravations.

b) Approach Two – using *svn hotcopy* to create backup and *rsync* the backup to the remote site

Within SVN default installation, there is a built-in tool called *svn hotcopy*. This tool is primarily used as to do a full backup of the SVN repository. As best practice, the backup will be stored in a temporary local storage rather than remote storage. Remote backup will result in the full repository being sent over the network each time which can be

painfully slow. Instead, synchronization tool such as *rsync*, is more suitable for data replication to the remote site This approach also does not interrupt user access to the repository as there is no service shutdown required. With that, the approach can potentially provide a FO and HA features into the current SVN infrastructure.

Out of the three criteria listed in Section 3.0, this approach only satisfy criteria (ii) and (iii).The findings from this approach can be summarized as follows:

i. To satisfy Section 3.0 criteria (i), the repository data must be replicated instantaneously to the remote site within a short period of time. However, with this approach, data replication can only be started once *svn hotcopy* is finished. Hence, instantaneous replication is not possible. The time taken to do a full repository backup using *svn hotcopy* is found to be directly proportional to the size of the repository data. As the size grew, the longer it took to complete the full backup.

ii. Due to (i), an hourly data replication is not possible. Users will experience data loss in the event of system FO occurs. The current size of the repository is approximately 350GB, and it took around 45 minutes to complete a full backup. The initial data replication to the remote site over a wide area network (WAN) with *rsync* took approximately one (1) hour. However, this was executed at 12.00 am. It will definitely take a lot more time if the experiment was conducted during working hours.

iii. Replication has to be done on the backup instead of live repository data. Replication over a live repository data generates error due to the data inconsistency. In order to avoid this issue, SVN primary service has to be stopped before the replication can be executed. This definitely interrupts user access and breaks the FO and HA requirement.

iv. The current *svn hotcopy* version has no incremental option; hence each time it took the same amount of time (around 45 minutes) to finish a full backup. However, subsequent synchronization through rsync took only 4 minutes. As the size of the repository data increases, the longer it takes to complete the full backup. Because of this, there is no possible hourly interval data replication can be executed.

v. Even though this approach does not interrupt user access, it aggravates users since the *svn hotcopy* uses lots of CPU and memory server's resources while active. Of course, stopping SVN primary service before the full backup would definitely resolve user's complaint on the system response sluggishness but the requirement stated that the application must be accessible at all time. Moreover, there are scheduled builds that are running and occasionally there are remote developers who work at different hours as well. Therefore, stopping service is not the best option.

vi. Since this is a full backup tool, the storage for backup has to be maintained and each backup will take up around 350GB. In terms of economic value, it is a waste of disk space just for data replication purposes.

Clearly from the findings, this approach seems practical only for smaller size data repository, as the *svn hotcopy* takes a few seconds to complete a full backup. Unfortunately, time taken to complete a full backup is directly proportional to the size of the repository data. In the practicality of the usage, this approach also failed to meet the requirement through experimentation, this approach is deemed as FAIL as well.

c) Approach Three – using *LVM snapshot* for backup and data replication

Two major deficiencies discovered from previous two approaches: service interruption in the event of system failover and server performance degradation during data replication. These issues aggravate users and increase their dissatisfaction towards the system. Furthermore, there is no way to schedule an hourly repository data replication to the remote site. This is a must have for an efficient SVN failover system. While achieving an instantaneous replication may not be possible but minimally there should be a way to schedule a replication to the remote site. Therefore, alternative solution using *LVM snapshot* is the third and final approach which is devised to resolve those niggling issues.

LVM snapshot [6] is a common practice used to do a backup [7] in UNIX system. Since the SVN server is running on Linux, a similar feat can be applied [8]. This approach would be a potential solution in minimizing service

interruption, resolving data replication issue and in doing so; it would not degrade the system performance. Furthermore, in order to satisfy Section 3.0 criteria (iii), the data replication should be done over a live repository data as to make sure SVN service is running normally. It is important to note that doing a replication of live SVN repository will produce file lock error and the data is deemed as corrupt data. This is not a recommended way to do a live replication for SVN. This is an additional issue that needed to be resolved to satisfy the HA requirement.

In Approach One and Two, SVN version 1.7 is deployed and it did not have a feature to manage user connection. Fortunately, SVN version 1.8 came out and it has a feature called *svn freeze* [9]. This option will delay commits while other operations are performed on the repository. While the operation is still running, the repository remains live and users are able to access the repository.

Combining LVM snapshot and svn freeze, issues from the two previous approaches are resolved instantly:

i. Access to the application is maintained at all time since command *svn freeze* will stop any attempt to update changes to the SVN repository while *LVM snapshot* t is being generated. As soon as the snapshot is created, SVN will allow the commit to complete. Since the snapshot generation took around 2-3 seconds, this delay is negligible and users will not aware that the snapshot has been generated.

ii. Users do not experience sluggishness in the application performance anymore as the snapshot data generated for replication purposes took just seconds. This is faster compared to using *svn hotcopy.* It consumes less server's CPU, memory and disk space as well.

iii. Replication frequency to the remote site increased significantly and can be scheduled at a specific time as well: every four hours, every two hours or even every hour. As SVN client takes a snapshot copy from the server, the client will not have to check for server connectivity all the time.. Therefore, a scheduled replication is efficient and effective enough to make sure that data in remote site is as current as the primary site. This in turns meet the FO and HA requirement and satisfy Section 3.0 criteria (i), (ii) and (iii).

d) Solution Implemented

LVM snapshot provides a viable solution to all the requirements requested by the management as proven by the mock setup. With that, the existing system infrastructure was modified accordingly to include mirrored site and a Global Traffic Management (GTM) as shown in Figure 1. Two of the SVN servers will be directly below a load balancer and by default the access will be directed to the primary site. In case of the primary system failure, the access will be re-directed to the remote site. The switch over will be transparent to the users and they should be able to use SVN normally. Meanwhile, in the background, currently set to run at every 4 hours, a data replication to the remote site is executed. To make sure there is no data inconsistencies, the replication will only from the primary site to the remote site. The replication will stop as soon as the failover occurs and the system will send an email notification the system admin on the switch for fallback procedure execution.

Users will be notified on the system shutdown prior to the fallback procedure. Fallback is considered complete once the data between remote and primary sites are in-sync again. Once the primary service is restarted, GTM will automatically re-direct the access to the primary site.

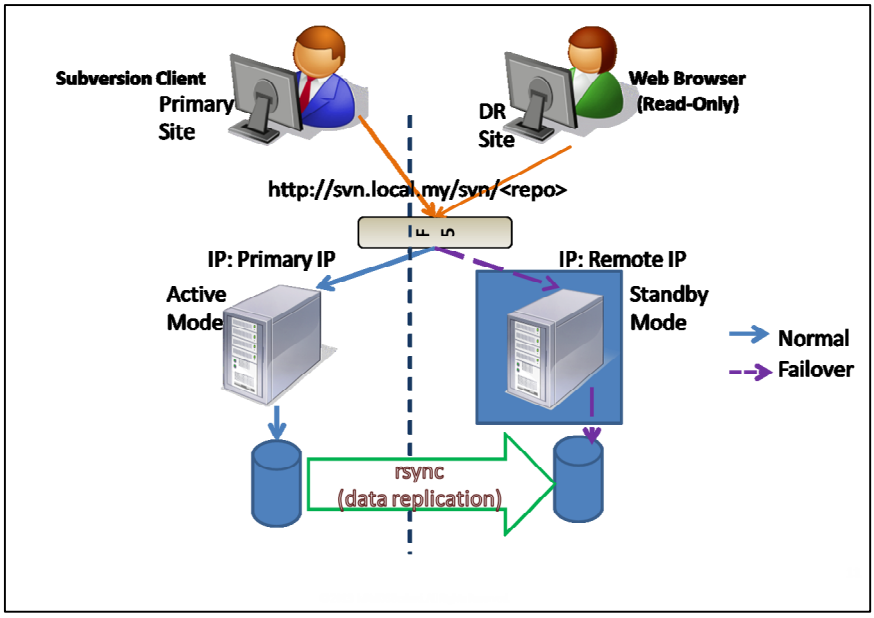

Figure 1: SVN System Infrastructure to cater for Failover and High Availability

The replication process flows can be summarized as below together with the shell script. The replication is currently set to run at every four hours and the whole process took less than 5 seconds to complete which is quicker compared to the other two approaches. Besides being fast and proficient, users still have the complete access to the application without being troubled by any system degradation.

SVN connection freeze➔ *LVM snapshot* created ➔*LVM snapshot* mounted➔rsync *LVM snapshot* to the remote site➔ *LVM snapshot* unmounted➔*LVM snapshot* removed ➔SVN connection unfreeze

Figure 2: Replication process flow

```
#!/bin/bash -x
WDIR="/opt/scripts/replication"
WTMP="${WDIR}/tmp"
DATE=$(cat ${WTMP}/date.txt)
DATALV="$(cat $WTMP/datalv.txt)"
#SNAPSHOT="snap-svndatalv-$DATE"
SNAPSHOT="snap-$(echo $DATALV|cut -d/ -f4)-${DATE}"
```

```
LVCREATE="/sbin/lvcreate"

   $LVCREATE -s -L 5G -n $SNAPSHOT $DATALV
```

Figure 3: Script to create *LVM snapshot*

```
#!/bin/bash -x
WDIR="/opt/scripts/replication"
WTMP="${WDIR}/tmp"
DATALV="/dev/svnictvm01vg/svndatavm01lv"
SVNADMIN="/usr/bin/svnadmin"
REPOLIST="${WDIR}/repolist.prod"
CREATESNAP="${WDIR}/createsnap.sh"
MOUNT="/bin/mount"
RSYNC="/usr/bin/rsync"
DESTSERVER="x.x.x.x"
DESTDIR="/svnict"
LVREMOVE="/sbin/lvremove -f"
DATE=$(date +%H%M%d%m%Y)

echo "=====================================start=============================="
if [ ! -d $WTMP ]; then
      mkdir $WTMP
fi
echo -n "$(date): "
echo $DATE > $WTMP/date.txt

echo $DATALV > $WTMP/datalv.txt

$SVNADMIN freeze -F $REPOLIST $CREATESNAP
sleep 1
if [ ! -d /mnt/snap-$(echo $DATALV|cut -d/ -f4)-${DATE} ]; then
```

```
        mkdir -p /mnt/snap-$(echo $DATALV|cut -d/ -f4)-${DATE}
        chown -R apache:apache /mnt/snap-$(echo $DATALV|cut -d/ -f4)-${DATE}
    fi
    $MOUNT /dev/$(echo $DATALV|cut -d/ -f3)/snap-$(echo $DATALV|cut -d/ -f4)-${DATE}
/mnt/snap-$(echo $DATALV|cut -d/ -f4)-${
    DATE} -o nouuid,ro
    $RSYNC -avh -e "ssh -C" --progress --timeout=60 /mnt/snap-$(echo $DATALV|cut -d/ -f4)-
${DATE}/ ${DESTSERVER}:${DESTDIR}/
    sleep 1
    umount -l /mnt/snap-$(echo $DATALV|cut -d/ -f4)-${DATE}
    $LVREMOVE /dev/$(echo $DATALV|cut -d/ -f3)/snap-$(echo $DATALV|cut -d/ -f4)-${DATE}
    sleep 1
    rmdir /mnt/snap-$(echo $DATALV|cut -d/ -f4)-${DATE}
    echo "====================================end==========================
```

Figure 4: Replication Script

4.1 Summary of Results

Below are the comparison between all three approaches in terms of time taken to complete a full backup and replication frequency. The result basically used as part of selection process to identify best implementation to meet the FO and HA requirement.

Table 1: Time Comparison

Approach	Time taken to take full backup	Time Taken to replicate to remote site	Replication Frequency
svnsync	**Not Applicable**	**Not Applicable**	**Only once daily**
svn hotcopy	**45 minutes**** **** run daily at 10.00pm**	**2 hours**** **Subsequent run: 2 minutes**	**Only once daily**
lvm snapshot	**4 seconds***** *****run daily at 10.00pm**	**4 seconds****** **Subsequent run: 4 seconds** ******run daily at every 4 hours**	**Every 4 hours*** ***may change to every 1 hour in the near future**

IV. CONCLUSION AND RECOMMENDATIONS

To setup SVN infrastructure with failover and high availability, it usually requires high investment but that is not always true. Deploying and customizing of existing free and open source software such as *LVM snapshot* and *rsync*, make it possible to lower TCO to have such system setup. From various experimentation conducted, those software

can be a good alternative to achieve the same objective with minimal cost. This selected approach also has the advantage of having a local backup where the snapshot for the full backup can be executed midnight daily while rysnc is perfect for remote data replication.

The *LVM* technic also is faster than *svn hotcopy-then-sync* method and will use less disk space. Even though, *LVM snapshots* do affect write performance, the difference is very negligible and unlikely to be noticeable and the system performance returns to normal as soon as the snapshot is generated. Therefore, the users will not experience the performance drop. The following are recommendations for others that plan to setup SVN infrastructure. These are findings that can be considered as lesson learnt.

i. Minimum SVN version required is 1.8. This is because the feature *svn freeze* is only available in the said version [9]. Without such feature, it would be hard to have uninterruptible SVN service and through this only replication frequency is able to be scheduled.
ii. Knowledge in *LVM snapshot* is a must and additional disk spaces are required for snapshot creation.
iii. Since, replication is executed very rapidly; there should be regular data verification on the data in the remote site. It is advisable to plan this as part of monthly system maintenance activities.
iv. It requires proper load balancer setup either with software based balancer or hardware based balancer.

Therefore, in any case of primary system failure, the routing of service will be done automatically.
For data repository storage, network storage such as SAN is recommended as compared to NFS. SVN over NFS is very slow as compared to SAN. This will have an impact on user's satisfaction.

REFERENCE

[1] http://svnbook.red-bean.com/en/1.6/svn.intro.whatis.html, last accessed 9 June 2014.
[2] http://www-03.ibm.com/software/products/en/clearcase
[3] http://www.nongnu.org/cvs/
[4] http://www.wandisco.com/subversion/download
[5] http://en.wikipedia.org/wiki/Version_control, last accessed 5 June 2014.
[6] http://tldp.org/HOWTO/LVM-HOWTO/snapshots_backup.html, last accessed 5 June 2014.
http://www.howtoforge.com/linux_lvm_snapshots, last accessed 5 June 2014.
[7] http://www.mysqlperformanceblog.com/2006/08/21/using-lvm-for-mysql-backup-and-replication-setup, last access 20 June 2014
[8] http://serverfault.com/questions/133698/svn-backup-using-rsync-command, last access 20 June 2014
[9] http://subversion.apache.org/docs/release-notes/1.8.html#svnadmin-freeze, last access 20 June 2014