ISTITUTO NAZIONALE DI FISICA NUCLEARE

CNAF

INFN-16-02/CNAF
4th February 2016

Feasibility study and performance evaluation of a GPFS based storage area accessible simultaneously through ownCloud and StoRM

Antonio Falabella, Sonia Taneja and Matteo Tenti

INFN CNAF, Viale Berti Pichat 6/2, Bologna, 40127, Italy

Abstract

In this work, we demonstrate the feasibility of a GPFS based storage area accessible through both ownCloud and StoRM, as solution which allows VO's with storage manager service to access directly the files through personal computer. Furthermore, in order to study its performance, in load situation, we set up two load balanced ownCloud web servers and a front-end, back-end and gridFTP StoRM server. A Python script was also developed to simulate user performing several file transfers both with ownCloud and StoRM. We employed the average file transfer time and efficiency as a figure of merit, which was measured as a function of the number of parallel running scripts. We observed that the average time increases almost linearly with the number of parallel running scripts, independently of the used software.

Pubblicato da SIDS–Pubblicazioni
Laboratori Nazionali di Frascati

1 INTRODUCTION

In order to fulfill new requests coming from small VOs which need a storage area, file manager and transfer tools but also a direct access to the files through personal computer (i.e. direct visualization through the personal computer of images residing at CNAF storage area), we investigated the following solution:

- ***GPFS*** as filesystem;
- ***StoRM*** as file manager and transfer tool;
- ***ownCloud*** as tool for web access to files and synchronization of folders.

GPFS (General Parallel File System) [1] is a high-performance clustered file system developed by IBM. StoRM (STOrage Resource Manager) [2] is a light, scalable, flexible, high-performance, file system independent, storage manager service (SRM) for generic disk based storage systems. ownCloud [3] is a suite of client-server software for creating file hosting services and using them.
Our aim was to test the feasibility, reliability and performance of this solution under high load situation. For this purpose, we employed 10 machines (named ds-06-xx.cr.cnaf.infn.it with xx ranging from 01 to 10) with configuration listed below in Table 1. It is worth to note that all machines mounted the same GPFS filesystem.

RAM	16 GB
Number of CPUs	2
Number of Cores	8
Disk Size	126 GB
GPFS Area Size	72 TB
Operating System	Scientific Linux 6.3

Tab. 1 - Configuration of machines used for the tests.

The StoRM server setup is described in section 2. The ownCloud instance implementation is reported in section 3, while the setup for the tests, their description and results are summarized in sections 4, 5 and 6 respectively. The conclusions are in section 7.

2 STORM

StoRM provides data management capabilities in a Grid environment to share, access and transfer data among heterogeneous and geographically distributed data centers. The main components of StoRM implementation are:

- Front-end
- Back-end
- GridFTP

We installed all the components on a single machine (ds-06-10) and the machine was configured so that storage area resided in the GPFS filesystem.

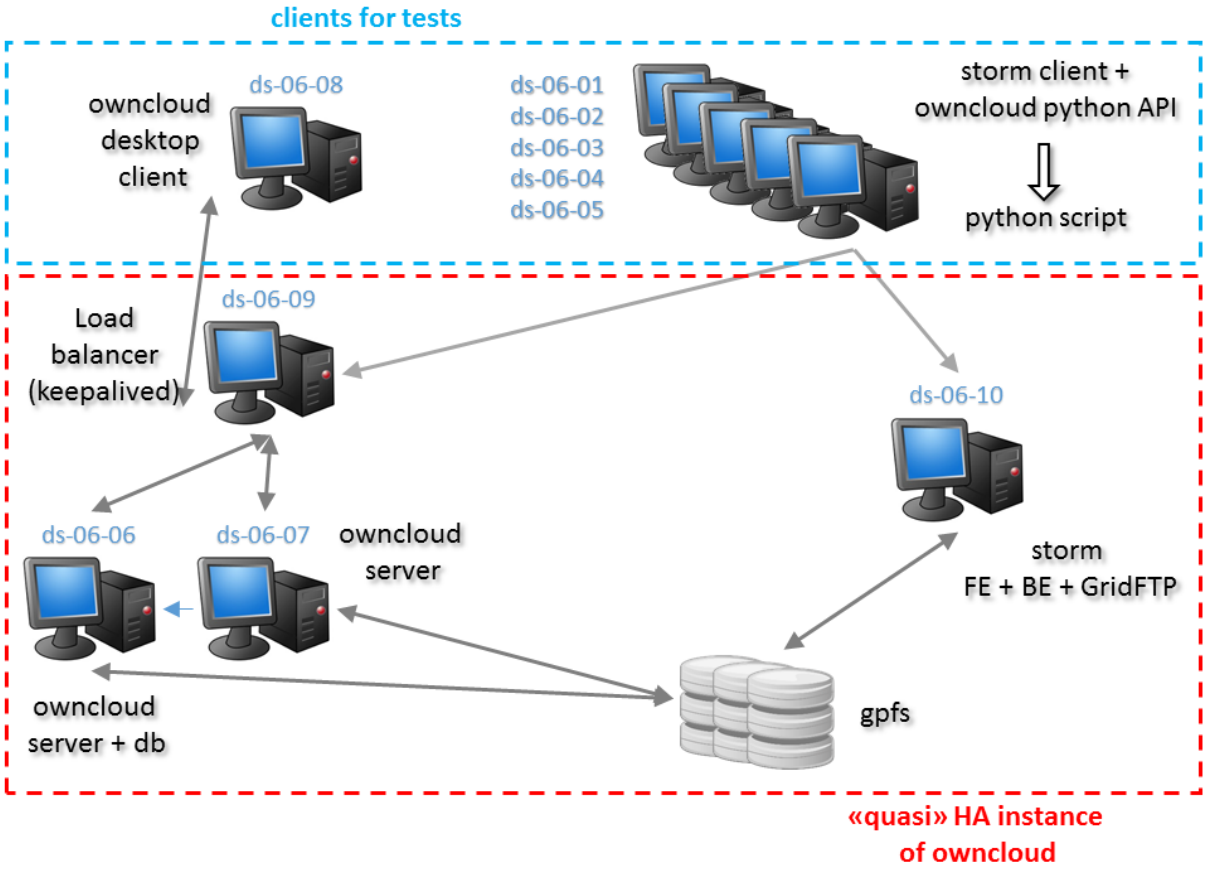

Fig. 1 - Schema of the "quasi" high available instance of ownCloud.

3 SETUP OF OWNCLOUD INSTANCE

OwnCloud is a self-hosted file synchronization and share server. It provides access to data through a web interface, synchronization clients or WebDAV protocol [4] while providing a platform to view, synchronize and share data across devices. Its main components are:

- Storage area;
- Database;
- Web server.

3.1 Storage Area

In our study we used the same GPFS based storage for StoRM and ownCloud storage areas. Since StoRM requires the ownership (*storm* user) of all the folders and files inside its storage area, we made the *httpd* service (of the ownCloud web interface) running as *storm* user. In this

way files uploaded with one system were automatically available to the other one. In our implementation since ownCloud area was inside the StoRM one, user's trash and cache is available through StoRM to all other users and this problem can be easily solved by setting ownCloud and StoRM into two independent areas and then linking (soft link) the StoRM area in each user's ownCloud area.

3.2 Web Server

We set up a round-robin based load balancing between the two ownCloud web servers (ds-06-06 and ds-06-07) and we used keepalived [5] as load balancer which was installed and configured on ds-06-09.

3.3 Database

We used MySQL [6] as database which resides on one of the web server (ds-06-06) while the other one (ds-06-07) was configured to perform query to the remote database hosted on web server ds-06-06. We deliberately did not set up a high availability instance of the database because it is out of the scope of these tests. Thus because of the lack of the database redundancy, we state the ownCloud instance we set up as a "quasi" high availability instance and not a fully high availability instance.

4 SETUP FOR THE TESTS

A sketch of the schema of the machines and their purpose is shown in Figure 1. In order to perform the tests, we filled our storage area to simulate a real storage area. Then we tested the ownCloud desktop client functionality and prepared the clients for the stress tests on the ownCloud instance.

4.1 Storage area setup

In our storage area we created a very structured directory tree, that starts from one single folder; then each folder contains two subdirectories for a total 10 levels corresponding to about 1000 directories. Later on we filled the storage area using 5 differently sized files (bitmap, jpeg and tif images). The files and their size are listed in Table 2. In this setup, we filled the storage area randomly using about 40k files for a total of about 30 TB.

File name	File size (bytes)
100kB.jpg	86950
1MB.jpg	1302600
10MB.jpg	14744835
100MB.png	113357104
1GB.tif	1376498512

Tab. 2 - Name and size of the files used for the tests.

4.2 OwnCloud desktop client

Once the storage area was set up, we installed the ownCloud desktop client on ds-06-08 and we

synchronize a subdirectory which can be seen from the screenshot of the configuration process, shown in Figure 2. The synchronization was successful and the client worked fine. However, we observed that, when a subfolder of the ownCloud storage area, is synchronized through the client, all the folders belonging to the direct tree from the ownCloud parent folder to the selected directory are also synchronized. This feature seems to be a bug and should be investigated further.

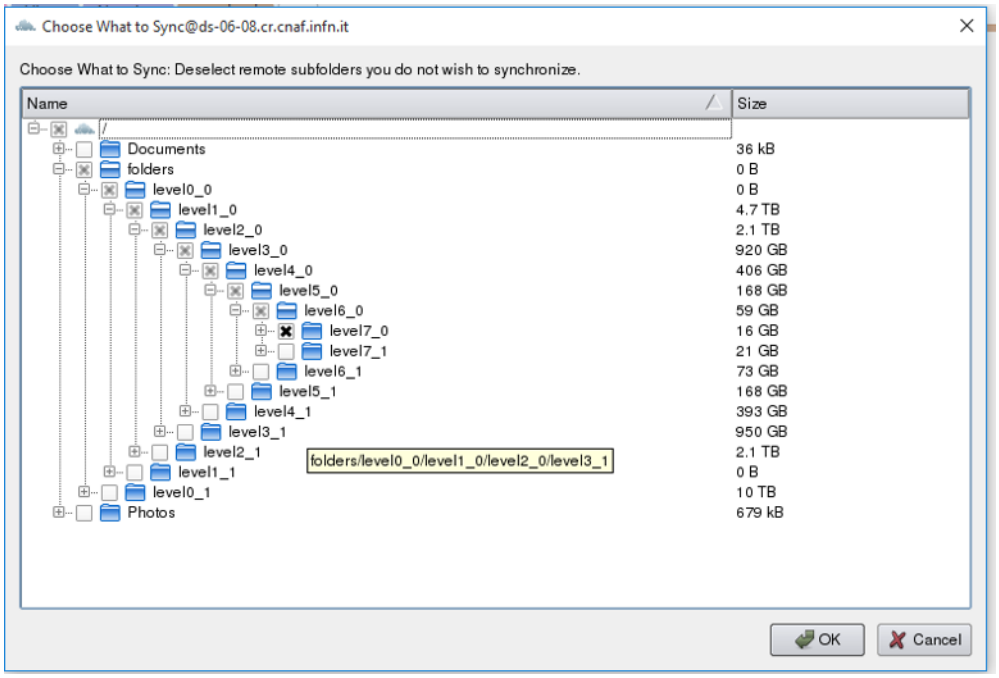

Fig. 2 - Screenshot of ownCloud desktop client configuration process.

4.3 Setup for stress tests

To perform the load test we used 5 machines (ds-06-xx, with xx from 01 to 05), where we installed Python client library for ownCloud [7] and the StoRM client [8]. We developed a python script to simulate a typical usage. The script performs several sequential transfers and for each transfer it chooses randomly:

- the service (ownCloud/StoRM)
- the operation (upload/download)
- the file size (100kB/1MB/10MB/100MB/1GB)
- the user (only for ownCloud transfer)

For an upload process an existing local file was selected and a remote not-existing file name was chosen and vice versa for a download process. Local and remote files were different for each process to avoid concurrent access to the same file by different processes. It has to be noted that local files are not under StoRM/ownCloud control but the remote and local files reside on the same GPFS filesystem.

5 STRESS TESTS

To perform the stress tests, we ran N_s scripts in parallel each performing N_t sequential transfers on N_m machines simultaneously, for a total of $N_p = N_s \times N_m$ scripts running in parallel and a total $N_{tot} = N_t \times N_s \times N_m$ transfers. As shown in Table 3 below, six tests were performed with different configurations. For all tests, the average transfer time and the standard deviation was evaluated for each software, operation and file size. We also calculated efficiencies which are defined as fraction of successful transfers.

N_s	N_m	N_p	N_t	N_{tot}
1	1	**1**	1000	1000
1	5	**5**	200	1000
5	5	**25**	50	1250
10	5	**50**	50	2500
20	5	**100**	50	5000
40	5	**200**	20	4000

Tab. 3 - Configurations of the six performed tests.

6 RESULTS

The results of the tests are reported in Appendix A. In Table 4 and Table 5 the ownCloud download and upload transfers time for different number of parallel running scripts are listed respectively, while Table 6 and Table 7 shows the download and upload transfer time for StoRM. OwnCloud and StoRM average transfer time for 1GB file size as function of the number of parallel running scripts is compared in Figure 3, as a representative result.

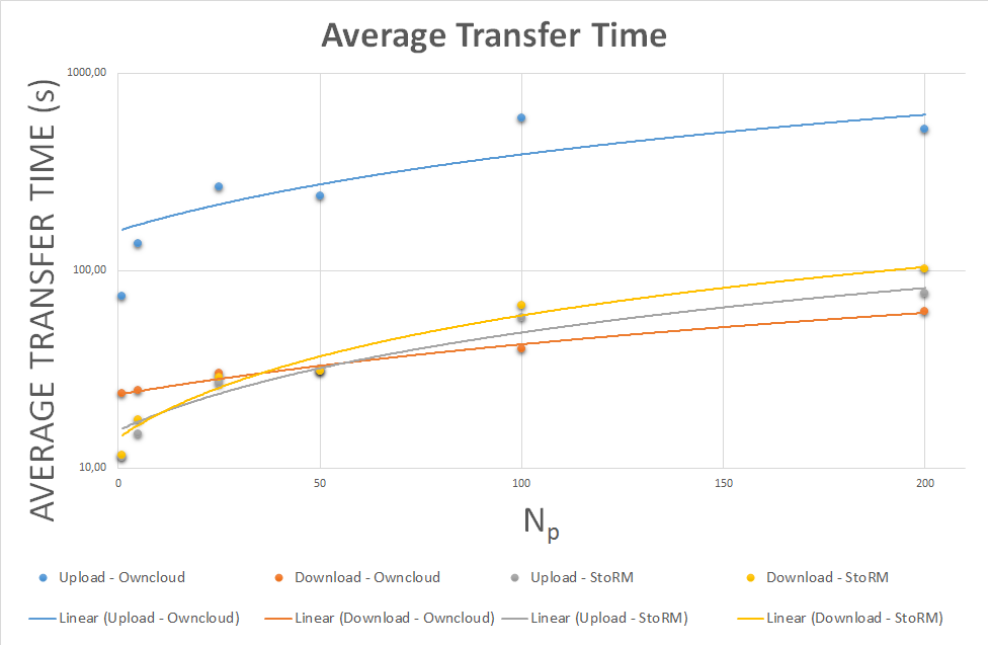

Fig. 3 - Comparison of the average download/upload transfer time for 1GB file size obtained with ownCloud (orange/blue) and StoRM (yellow/grey).

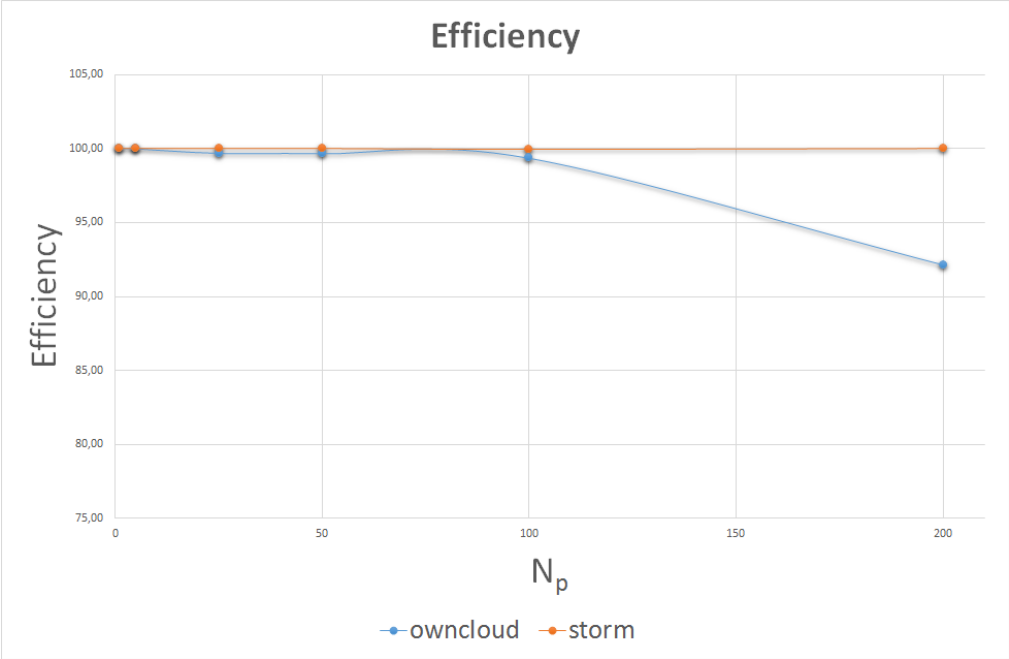

Fig. 4 - Efficiencies of the upload process for ownCloud (blue) and StoRM (orange). The download efficiencies are 100% for both ownCloud and StoRM.

Fig. 5 - OwnCloud efficiency obtained with the quasi high availability instance (blue) while the orange line represents the efficiency when the test was repeated with only a single ownCloud web server.

In general, the transfer time increases linearly with the number of parallel running scripts. This feature is due, at least for high number of parallel scripts running, to bandwidth saturation. For

what concerns the uploads StoRM performs better than ownCloud: StoRM transfer time is one order of magnitude smaller then ownCloud for 1GB file size and Np = 200. The performance for downloads instead are similar. It is to be noted that this result was expected given the different purpose and implementation of the two software products. Figure 4 shows the efficiency for ownCloud and StoRM as function of the number of parallel running scripts. The StoRM upload efficiency is always around 100%, while the ownCloud upload efficiency starts to decrease with Np = 100 and reaching about 92% for Np = 200. In Figure 5 the efficiencies of ownCloud upload efficiency obtained with the quasi high availability instance (blue line). For comparison, the test was repeated with a single ownCloud web server; the resulting efficiency is also shown (orange line). It is evident that the performance improves: the efficiency increases from 39% to 92% for the case with Np = 200.

7 CONCLUSIONS

In conclusion, we set up a simple hybrid architecture for storage management based on GPFS filesystem and accessible through both StoRM and ownCloud. Two load balanced ownCloud web servers and StoRM backend, frontend and gridFTP were set up. We demonstrate the feasibility of such a solution for the use case of small VOs requiring direct access to the files through personal computer as well as a storage area, file manager and transfer tools.
We developed a python script to simulate the user actions. Using the script, we performed load tests measuring the ownCloud and StoRM performance in terms of average transfer time. The transfer time increases linearly with the number of parallel transfers. In particular, during uploads StoRM performs better than ownCloud, as expected given the different purpose of the two software solutions however the download performances are similar. The ownCloud efficiency is above 90% up to N_p = 200 and it starts to decrease with N_p = 100.

8 APPENDIX

In conclusion, we set up In this appendix, we reported the measurement of the ownCloud (download: Tab. 4 and upload: Tab. 5) and StoRM (download: Tab. 6 and upload: Tab. 7) performance in term of average time transfer.

OwnCloud / Download transfer time (seconds)					
N_p	**1GB**	**100MB**	**10MB**	**1MB**	**100kB**
1	23.9	2.4	0.7	0.4	0.3
5	24.8	2.6	0.8	0.4	0.3
25	30.1	3.5	1.5	0.6	0.8
50	31.6	4.2	1.4	0.7	0.6
100	40.2	5.7	2.6	1.6	1.7
200	62.3	7.4	2.9	1.5	1.2

Tab. 4 - Time of the download transfers performed with ownCloud.

OwnCloud / Upload transfer time (seconds)					
N_p	**1GB**	**100MB**	**10MB**	**1MB**	**100kB**
1	74.6	5.9	1.8	1.2	1.0
5	137.0	8.1	3.4	2.2	2.7
25	266.5	22.0	12.4	7.2	5.5
50	239.1	18.9	13.3	6.0	7.6
100	590.9	58.4	28.7	23.6	23.5
200	518.0	61.4	25.5	20.8	23.1

Tab. 5 - Time of the upload transfers performed with ownCloud.

StoRM / Download transfer time (seconds)					
N_p	**1GB**	**100MB**	**10MB**	**1MB**	**100kB**
1	11.56	1.37	0.66	0.39	0.26
5	17.48	2.00	0.63	0.30	0.24
25	28.73	3.29	1.01	0.41	0.38
50	30.85	3.06	1.08	0.46	0.31
100	66.87	6.75	1.71	0.86	0.51
200	101.53	11.05	2.28	0.95	1.26

Tab. 6 - Time of the download transfers performed with StoRM.

StoRM / Upload transfer time (seconds)					
N_p	**1GB**	**100MB**	**10MB**	**1MB**	**100kB**
1	11.40	1.24	0.41	0.26	0.22
5	14.93	1.76	0.61	0.30	0.24
25	26.97	2.75	0.97	0.45	0.25
50	31.04	3.03	0.86	0.38	0.31
100	57.82	6.10	1.46	0.66	0.40
200	76.95	8.22	2.03	0.79	0.87

Tab. 7 - Time of the upload transfers performed with StoRM.

9 REFERENCES

[1] www-03.ibm.com/software/products/en/software
[2] italiangrid.github.io/storm/index.html
[3] owncloud.org
[4] L. Dusseault, "HTTP Extensions for Web Distributed Authoring and Versioning (WebDAV)", RFC 4918, June 2007.
[5] www.keepalived.org
[6] www.mysql.com
[7] github.com/owncloud/pyocclient
[8] italiangrid.github.io/storm/documentation/clientsrm-guide