Performance comparison of filesystems on RAID for e-VLBI data

Matteo Stagni - Francesco Bedosti - Mauro Nanni

April 23, 2013

IRA 470/13

Abstract

A thorough description of several tests conducted on a single node and multiple nodes computer environments. The investigation aims to discover which are the best ways to store and correlate e-Vlbi data using off-the-shelf equipment and assuming certain standard filesystem and RAID configurations as the best possible. In-depth testing and comparison is carried on Lustre distributed filesystem and Vlbistreamer e-Vlbi data streaming software.

Contents

1 Test Environment

Following the previous tests background on different filesystem in paper IRA 458/12 [6] we have decided to do further testing comparing a new machine purchased recently having a new internal raid configuration.

These new tests aim is to find how much a different internal raid configuration can improve file writings when streaming data from antennas, in order to achieve the best possible performance requested by the FlexBuff design in the *NEXPReS* project [2]. Our FlexBuff implementation is based on a RAID 5 deployment of a single available archive space for each machine.

The following round of tests will include the XFS filesystem format that proved to be the most suitable for the FlexBuff needs, further investigating new viable options that the recently purchased machine offers.

1.1 Previous tests on XFS

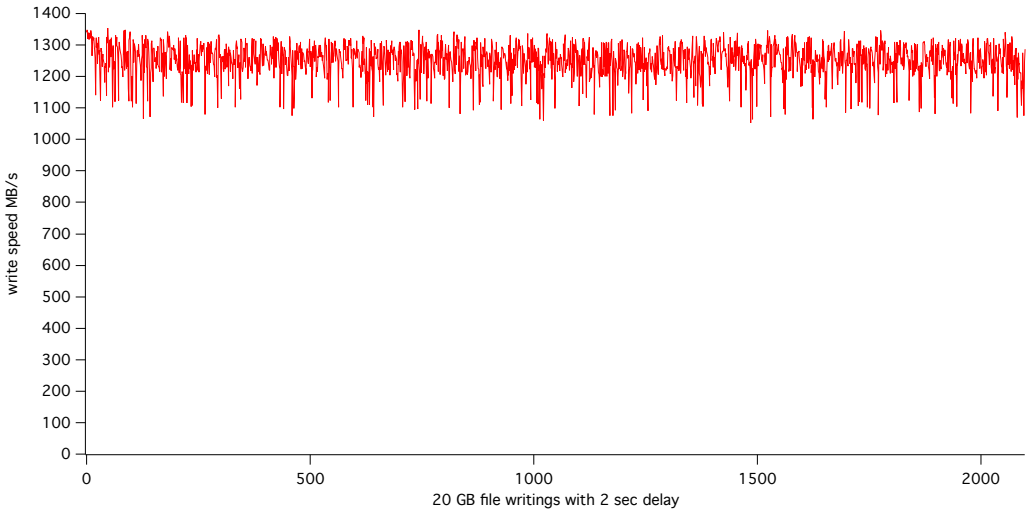

Figure 1: XFS on RAID5

Figure 1 is our starting point for tests with linux dd utility . We had previously purchased 3 machines whose purpose is to act as storage components for the three Italian antennas: Medicina, Noto and SRT.

OS	CPU	RAM	Motherboard	PCI-express	GT/s	3WARE RAID card
Scientific Linux 6.2	2 XEON E5620 @2.4GHz	12GB	X8DTH-IF	7x16	6.4	1x 9750DR

Table 1: Tanks configuration

The system configuration of the previously cited machines, from now on called *Tanks*, described in Table 1, sets a benchmark environment were we experimented with Btrfs, Jfs, Ext4 and XFS filesystems on a RAID5 setup of disks, proven there was no significant difference confronted with a RAID0.

		Configuration on XFS	
Disk N.	**RAID**	**MB/s**	**Notes**
1	JBOD	91	
3	RAID5	207	Balanced
6	RAID5	545	Balanced
12	RAID5	1100	Balanced
24	RAID5	1200	Balanced
24	RAID0	1200	Balanced

Table 2: Systems configurations test results

1.2 New machine environment

The newly purchased machine in 2013 has a different internal setup concerning mainly the RAID arrays. Three LSI 9750-8i [4] supporting 8 disks each were installed along with a greater amount of RAM memory. Another difference from the previous setup are the SATA disks now having a 3 TB dimension making up a total 72 TB available on a single machine.

OS	**CPU**	**RAM**	**Motherboard**	**PCI-express**	**GT/s**	**3WARE RAID card**
Scientific Linux 6.2	2 XEON E5-2609 @ 2.4GHz	32GB	Intel E5-2600 C602	2x16, 4x8	8	3x 9750-8i

Table 3: 3 RAID card machine

This machine setup was chosen due to its future deployment as a host for users' data which will host the institute services. Though we already achieved positive results on the previous machines considered, this new environment provided us an opportunity to verify whether a different hardware setup could lead us to different, maybe better results for the purpose of writing e-vlbi data in a more efficient way.

2 Tests on 3 RAID5 with 8 disks, same machine

The environment depicted in Table 3 provided us different testing options. There was a need for us to understand whether the low growth of performances displayed in Table 2 comparing a 12 disks configuration to the 24 one is due to motherboard bus limits, CPU limits or to the RAID card. First we needed to investigate what could have happened using a single RAID card. Next we moved to using the three of them, still in an independent fashion, but writing at the same time. Then it was chosen to get them working as an aggregate, setting the three RAID5 as a single volume through LVM.

2.1 Testing on a single RAID

Taking into account Figure 1 as our reference, the dd writing test did not reveal unexpected results. The lower writing speed can be attributed to the reduced number of disks, and in part to the larger size of them, now being 3 TB instead of the previous 2 TB.

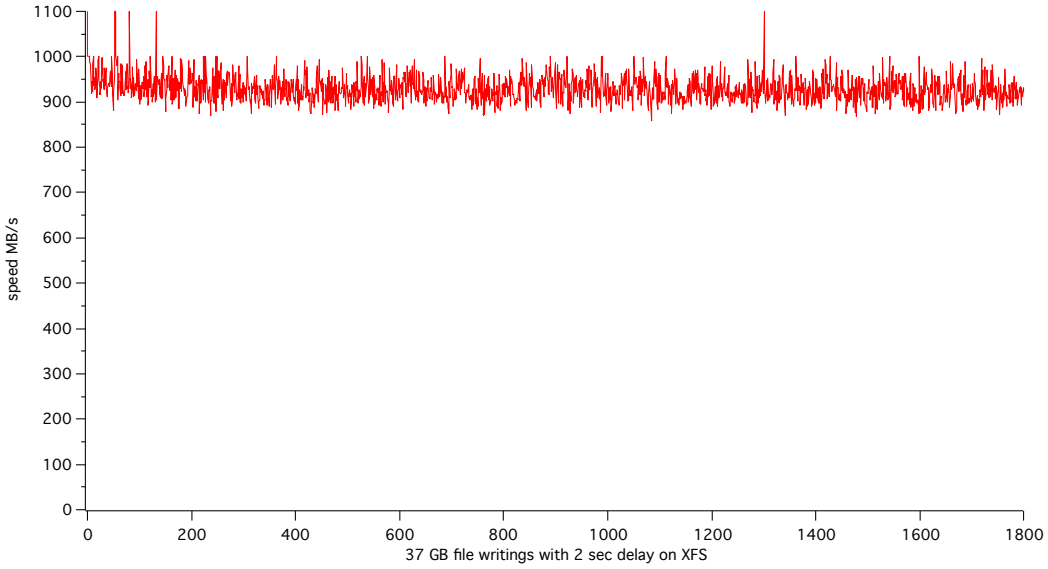

Figure 2: RAID5 8 disks full

The filesystem underlying the RAID is still XFS, whereas the file size written is now 37 GB. This is due to the larger amount of memory described in Table 3 compared to the *Tanks* where we had 12 GB. The increased file dimension was chosen to minimize any writing cache. The previous test results confirm what described before in Table 2 where the last dd writing performance places itself among the 12 disks and 6 disks writing speed.

Next natural step was to tell if there was any difference between the three cards considered. For this test a lower number of writings was chosen and the three RAIDS were labelled as *upper*, *middle* and *lower*, considering the disks relative position on the

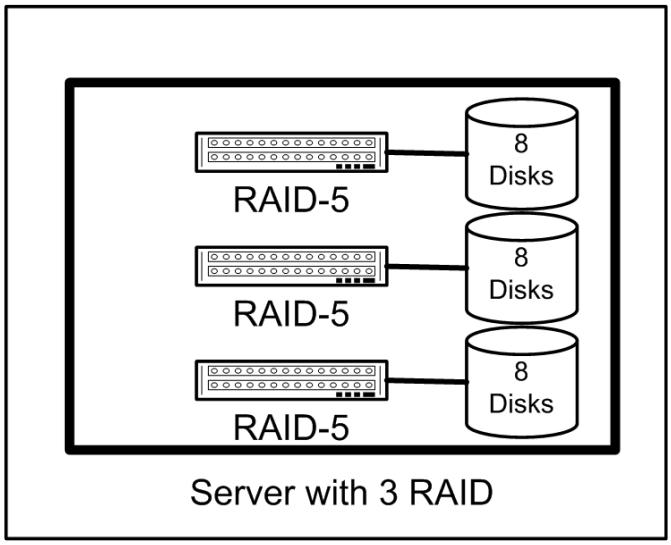

Figure 3: RAID cards configuration

machine which each RAID card was in charge of controlling as shown in Figure 3. Each writing session was performed at different times. Much to our surprise was to discover that the labelled *lower* RAID did not behave the same way as the other RAIDS did. The previous test in Figure 2 was done on the so-called *middle* RAID, in fact the following result shows little deviation from the results obtained before.

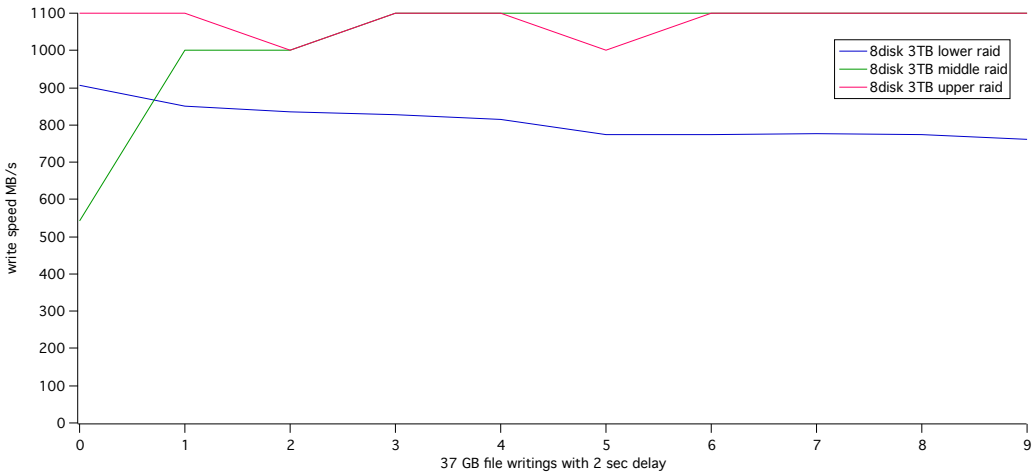

Figure 4: 3 RAIDS dd test

Though we did not get any warnings or failure alerts from the LSI 3dm2 monitoring system this unexpected behavior left us puzzled about asking LSI for a substitution. Due to the little time available before this machine entering production status, we decided to continue the job trying to involve the supposed faulty card the least possible.

The following test involves the *middle* RAID with IOzone [1] choosing the same parameters described in the previously mentioned report [6].

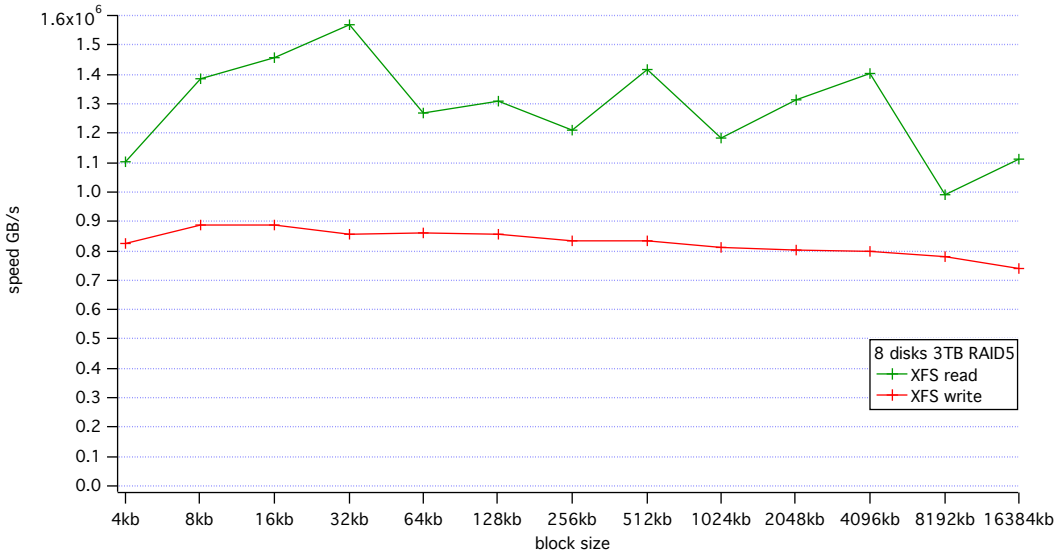

Figure 5: 3 RAIDS performance iozone

In Figure 5 above there is no significant difference to tell between the behavior of the new RAID cards and the elder ones on *Tanks*. The XFS filesystem shows the same irregular pattern while reading at different block sizes, partially due to the 256 kb block size setup on the RAID5. The writings confirm the average speed we got in the dd test.

2.2 Simultaneous acting performance

Moving forward to squeeze the best possible performance out of our new system we decided to do a dd test on the three RAIDs simultaneously writing. The test confirms the poorly behaving *lower* RAID whereas the latter two RAIDS show a similar average writing speed at sound 1 GB/s. If it weren't for the previous test where writing in separate moments on the three cards we could say there was a decay on the *lower* RAID due to the OS having difficulties handling the three cards, but even monitoring the system resources through a top command we were notified system resources weren't strained from performing the simultaneous task.

The last IOzone test is more complicated to interpret. We had seen a regular writing pattern in Figure 5, but this time the three RAIDs acting together messed the results up. Although we had reckoned a sort of better behavior between block sizes 256kb and 512kb in the previous report [6], we are now experiencing a similar writing performance on the three RAIDs ranging from 0.8 GB/s to 1 GB/s whereas the readings have a completely unpredictable behavior. Strain on the system wasn't too hard, though worst than in the previous test.

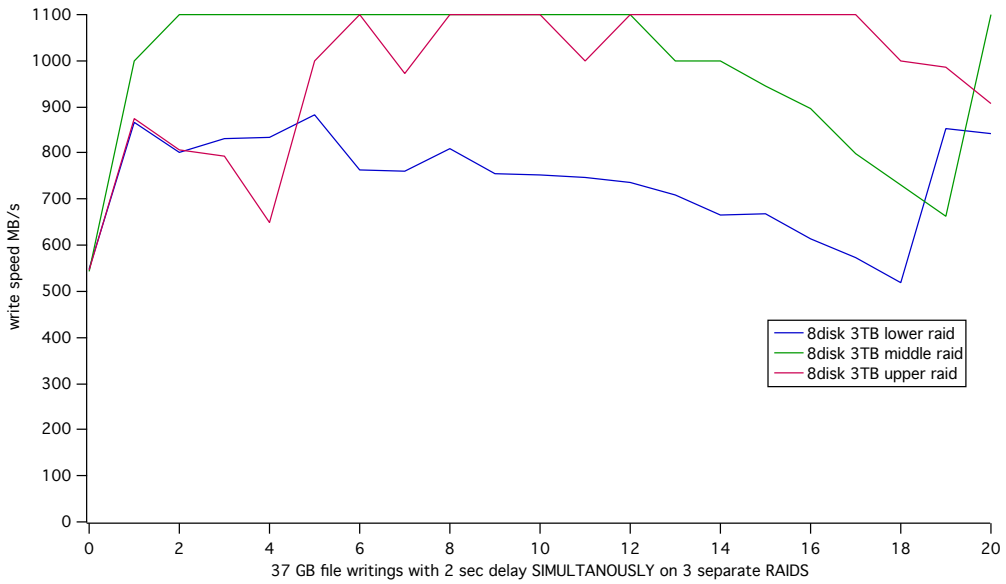

Figure 6: 3 RAIDS writings

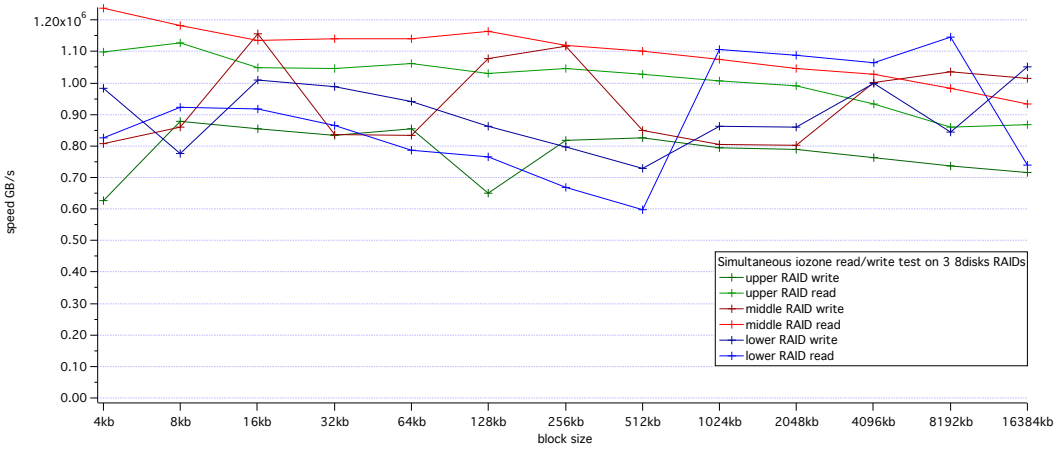

Figure 7: 3 RAIDS writing and reading simultaneously

2.3 LVM

Tightening the bonding between our RAIDs we decided to experiment with a Logical Volume Manager setup. LVM is best known for its ability to resize partitions and take snapshots for backups. In this case we wanted to explore a less common option which is to create a stripe using the *-i* option. On a first unsuccessful attempt we forgot to take into account this option so we obtained a contiguous space made of the there raids adjoined together but acting independently. That meant, on the first dd test performed, only a single RAID card acting: the other ones would have performed the task once the first array had been filled with data. After formatting again the three partitions with a *lvcreate -i 3 -I 256* that meant to create a RAID0 with a 3 drives striping block size 256kb like the LSI RAID5 setup, we were able to see the three RAIDs writing at the same time.

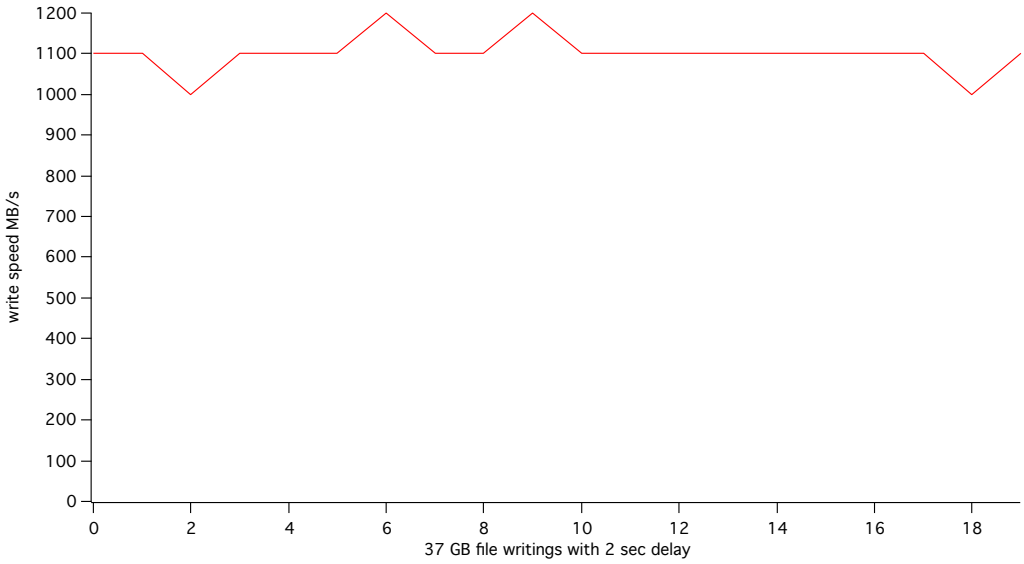

Figure 8: dd LVM test

The result after the dd test properly performed in Figure 8 does not diverge from what we obtained in Figure 2 or Figure 4. Instead it improves the results seen in Figure 6 mitigating somehow the three simultaneous writings on the three separate RAIDs. The LVM bond definitely strengthens the joint cards performance.

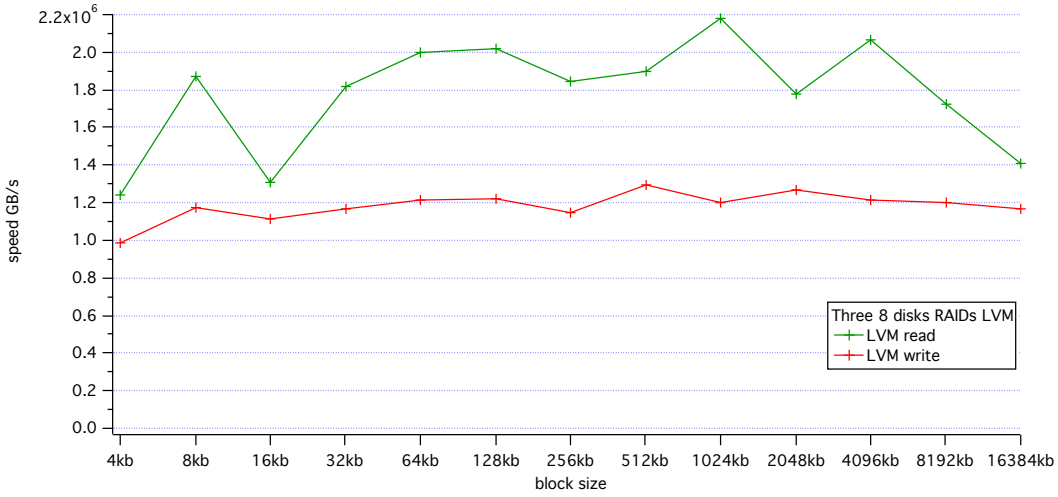

Figure 9: IOzone LVM

Following this promising result the IOzone test was not disappointing either. Writings confirmed the cap of 1.2 GB/s best performing when writing a 512kb block, but the most satisfying result was a much improved reading speed peaking almost 2 GB/s. We can say with confidence that so far the LVM setup enabled us to get the best possible writing and reading performance using an XFS filesystem.

3 Lustre Filesystem

A Lustre file system has three major functional units.

A single metadata server (MDS) that has a single metadata target (MDT) per Lustre filesystem that stores namespace metadata, such as filenames, directories, access permissions, and file layout. The MDT data is stored in a single local disk filesystem, which may be a bottleneck under some metadata intensive workloads.

However, unlike block-based distributed filesystems, such as GPFS, where the metadata server controls all of the block allocation, the Lustre metadata server is only involved in pathname and permission checks, and is not involved in any file I/O operations, avoiding I/O scalability bottlenecks on the metadata server.

Lustre and GPFS are the mainstream option for a great number of nodes in multiuser environment, where a lot of concurrent applications are supposed to run smoothly.

One or more object storage servers (OSSes) store file data on one or more object storage targets (OSTs). The capacity of a Lustre file system is the sum of the capacities provided by the OSTs.

Lustre presents all clients with a unified namespace for all of the files and data in the filesystem, using standard POSIX semantics, and allows concurrent and coherent read and write access to the files in the filesystem.

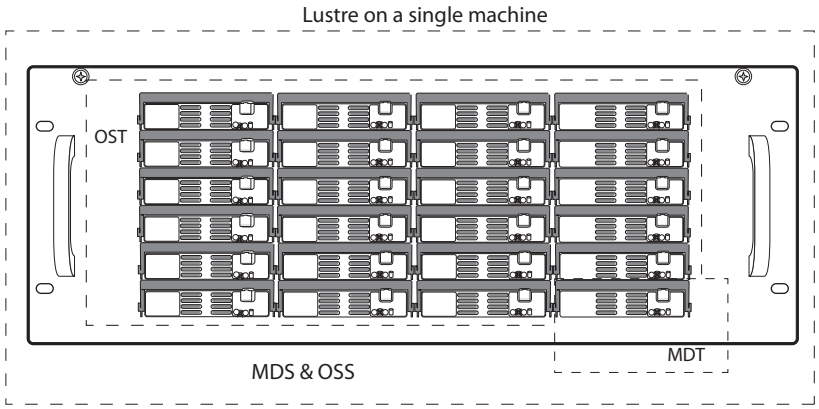

Figure 10: Lustre setup on a single machine

Our investigation aimed at discovering whether this filesystem would behave properly in case of a few data streams that required the fastest writing speed possible.

Provided this preliminary description [7], we need to say that tuning Lustre is not straightforward. In our case we needed to adapt the architecture of a distributed filesystem onto a single machine. To achieve this we have chosen a disk as MDT and the latter disks as OST making both the server and client live in the same place.

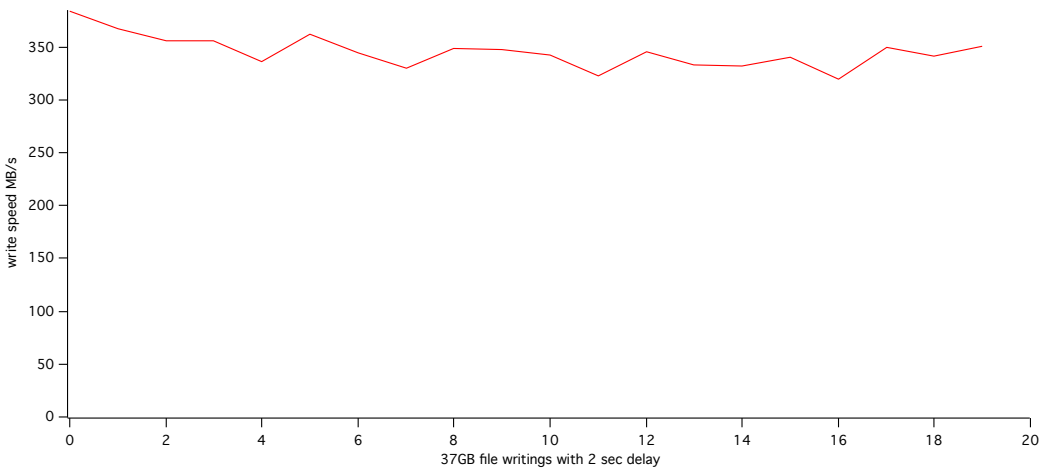

Figure 11: dd Lustre

Even though we tried to achieve the best possible configuration for Lustre, the writing performance tested with dd writings is not good. The average speed of 350 MB/s is far lower compared to the previous configurations. This is mainly due to the atypical single node configuration where Lustre cannot leverage its concurrence abilities.

The agent of change we wanted to stress on Lustre was the 3 separate RAID configuration on the machine, to verify if it had any positive influence. Unfortunately this was not so, and the result is furthermore confirmed by the IOzone test.

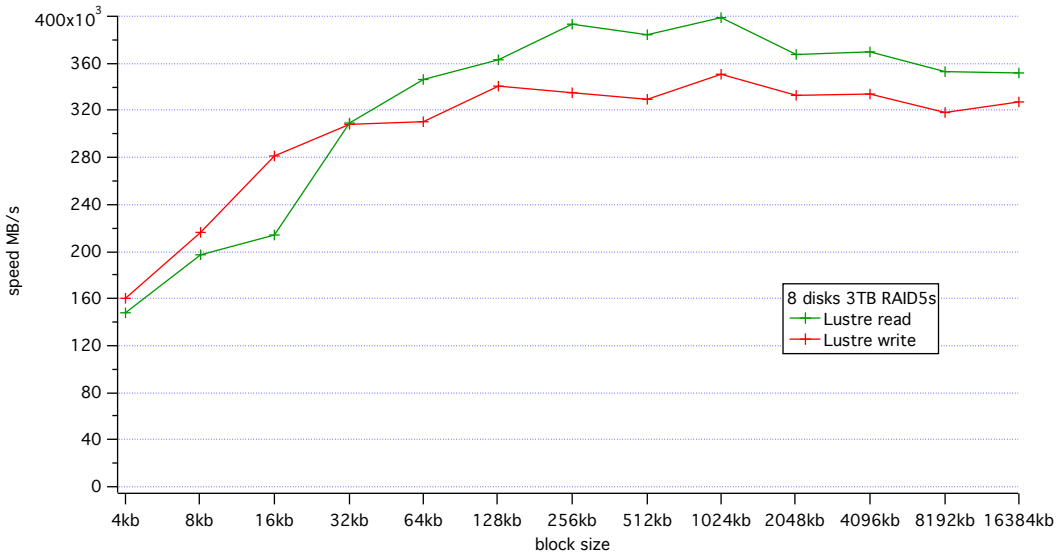

Figure 12: IOzone Lustre

In Figure 12 the results confirm the previous block sizes trend. The reading speed instead shows a more regular behavior following the writing speed.

3.1 Lustre on *Tanks*

Resuming from the last experience we had testing Lustre on a single node, we decided we had better performed another test on the *Tanks* where a 3 node environment could resemble more a real-life deployment of a Lustre filesystem. The MDS implemented on the previous machine was involved in the setup of the three *Tanks* that became OSTs of each respectively 22 TB on a RAID5 configuration.

The confrontation between our *Tanks* with a single 24 disks raid card versus the machine with 3 raid cards on the same bus has provided us with interesting results showed later on.

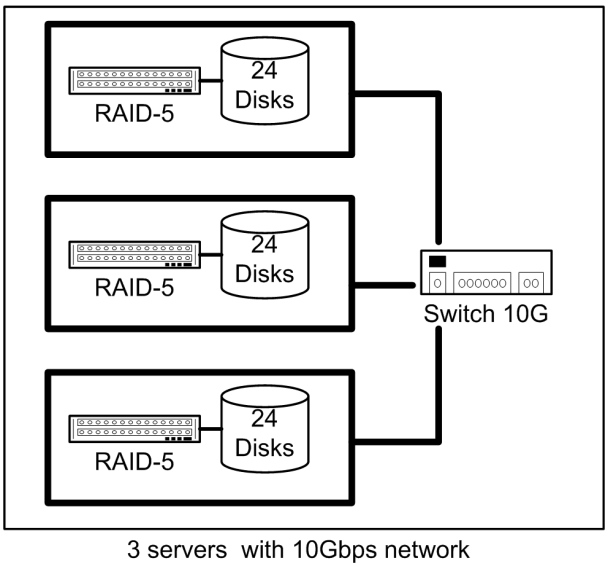

Figure 13

In Figure 13 is defined a schema where Lustre is set up on a 10 Gbit connection. A *Tank* still serves as both MDS and OSS like before, but now the latter two nodes are plain OSSes. The three *Tanks* also have an InfiniBand connection, but we were not able to test it due to the lack of documentation regarding Lustre implementation on that network protocol.

3.2 dd Lustre

The dd test on Lustre is significantly poorer then the other environments. Though the advantage of a distributed filesystem is indisputable on the side of an easier expandable and manageable archive space, the writing speed is not adequate to the purpose of writing fast enough the amount of data the *Tanks* are supposed to receive in a short amount of time.

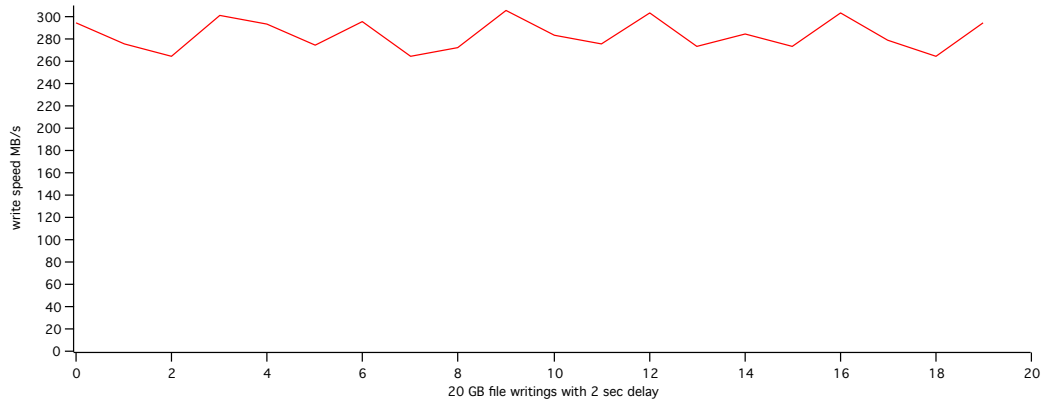

Figure 14: dd Lustre single writing

Although the result was not satisfactory, provided the nodes Lustre was working onto were three, we decided to evaluate its behavior while performing two and three simultaneous writings. The results shown in Figure 15 do not fall too far from the single writing test we had before though there is a constant decrease of the writing speed the more writings performed.

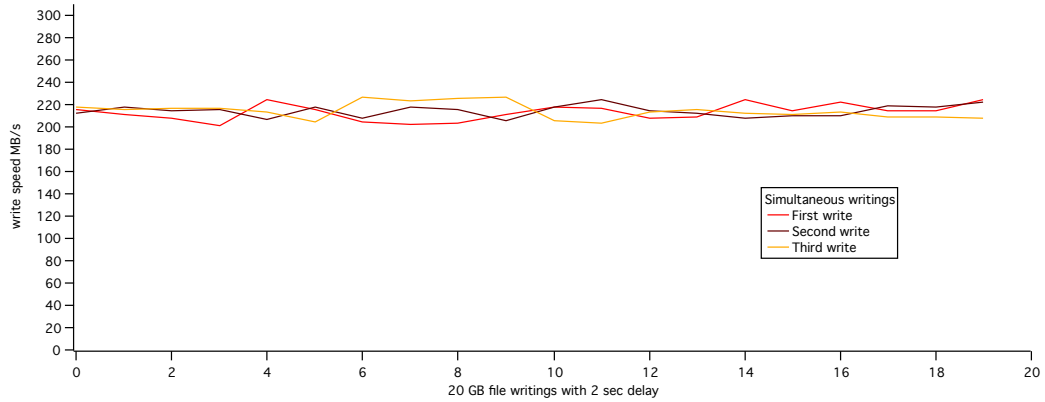

Figure 15: dd Lustre triple writing

3.3 IOzone Lustre

The IOzone test performed on the three nodes *Tanks* is slightly worst than the one done on the single machine. Figure 16 shows the reading and wryting speed have the same trend as the ones in Figure 12 getting the best behavior between 256 and 512 kb block size.

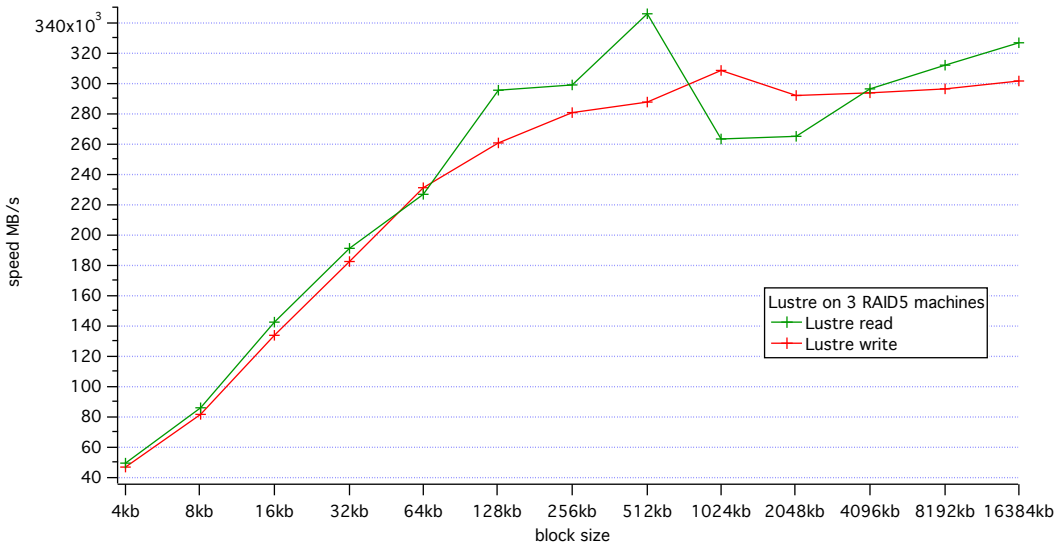

Figure 16: IOzone Lustre on Tanks

At the end of this detailed analysis of Lustre we came up to the conclusion that this kind of distributed filesystem is not suitable for our purposes. There may be a chance of deeper testing if only we had a more conspicuous number of nodes available, though a visit to the Bonn correlator, where the production environment is made up of 80 nodes, definitely put us off because Lustre had already been tested intensively there and the system engineers had better decided to set up an XFS environment with NFSv4 exports on automount on each node to provide filesystem distribution.

4 Vlbistreamer

A *NEXPReS* [2] core development for building a reliable storage and data transfer service throughout Europe is Vlbistreamer [5], developed for the purpose of storing and sending data on the network at at the highest speed possible (from 1 to 16 Gbit/sec). Then comes the storage capacity on a common filesystem installed on the so-called FlexBuffs (our *Tanks*). Finally to enable the data once stored to be sent out to other FlexBuffs or to a correlator.

Vlbistreamer architecture is based on ring buffers of memory that are loaded by data received from the network and then flushed to disk. The way this task is performed could be either on multiple disks formatted and mounted singularly or to a single disk as should be in the case of a RAID array. In order to enable the unrelated disks modality the data is usually split into files of a maximum 2 GBs each, in this way the work is parallelized and the writing speed optimized. Afterwards a contextually written .cfg file is able to trace back the single scan data and through a FUSE mount enable correlation.

File size	**write speed**
256 MB	5250Mbit/s
512 MB	5606Mbit/s
1 GB	5819Mbit/s
2 GB	5841Mbit/s

Table 4: Vlbistreamer write speed on a RAID array as single disk

The situation presented in Table 4 shows there is a steady increase of writing performance while the file size dimension grows. Data in this tests comes from a network stream. This means there would be no contraindications in developing a version of Vlbistreamer that could write the scan stream to a single file because at present data on discs cannot be used as straightforward input to a correlator like DiFX, or sent through the network via standard internet tools, so the original stream has to be re-built with a FUSE-like software.

Our peculiar hardware configuration compelled us to enable Vlbistreamer to write to a single mounted RAID array in a single disk fashion for the program. Because the software was not developed keeping hardware RAID into consideration we decided also to include a test with 12 disks in a JBOD fashion set on the RAID card.

These trials were performed by sending a dummy stream of UDP data to localhost and then trying to catch it through Vlbistreamer. The *Tanks* were set up with a RAID volume of 12 disks which had been recognized by Vlbistreamer as a single disk.

As a matter of fact we decided we had better chosen to keep a FlexBuff machine with

both a single RAID volume and 12 single JBD 2-4 TB disks.

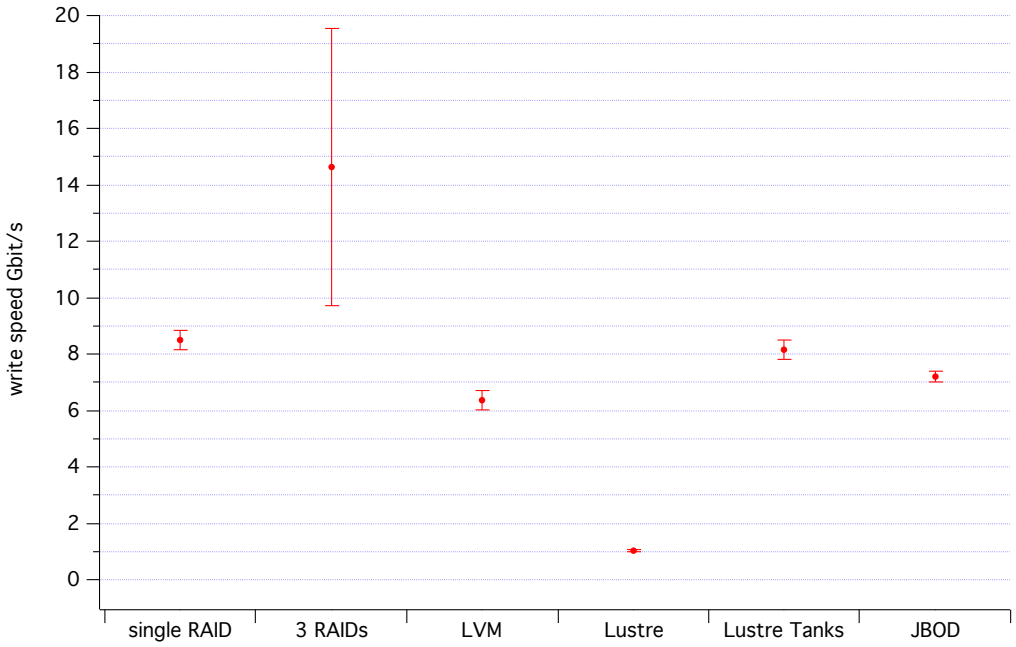

Figure 17: Vlbistreamer performances compared

The results in Figure 17 show some remarkable differences compared to previous tests.

First column represents the results obtained from a single writing performed on a 8 disk RAID array formatted with XFS.

Second column is about results obtained using the three RAID volumes as different disks for Vlbistreamer.

Third column is a test run on the three 8 disks RAID arrays formatted with XFS and made up as a single array with LVM, whereas fourth and fifth column are about Lustre, on a first instance on the single machine, then on the three *Tanks* nodes.

Finally last column displays Vlbistreamer writing speed using 12 JBOD disks on a *Tank* computer.

There is also a noticeable improvement though when testing on Lustre on the three *Tanks* nodes. It seems the much smaller file writings of Vlbistreamer affect positively the performance though caches may have taken part into this positive result.

The latter columns present more interesting behaviors of the software.

There is a significant writing speed variance when writing to the three RAIDs three separate streams simultaneously, maybe still due to the affected *lower* RAID, as shown in second column. Lustre on a single node still gives out poor results, but it has to be taken into account that a configuration where the Lustre OST and ODT live on the same machine is not recommended.

That said, we need to stress again that Vlbistreamer was tested providing a stream of

data coming from the same machine in a localhost environment. The JBOD array when tested from an outer machine source is not doing that well. In fact the maximum writing speed reached was about an average of 5 Gbits/sec.

4.1 Pre-production test

The main aim that stands behind Vlbistreamer development is to catch a stream of output data produced by a dBBC through a FILA10G card [3]. The FILA10G is now able to send a stream of maximum 4 Gbits/sec. Only Mark5s could record the data on a proprietary filesystem up to 2 Gbits/sec until recently.

All in all we can affirm that in most situations we are able to handle with a good confidence margin an output stream from a FILA10G card. A dBBC was acquired by IRA in Medicina the months these tests were conducted. Though not completely ready for production we had a chance to try a stream of dummy data from a real life FILA10G card pointing at one of our *Tanks* in Bologna.

Distance between Bologna and Medicina is approximately 30 km, speaking in network terms is 5 hops on a 10 Gbit fiber. We tested the speed of a FILA10G stream at 512, 1024, 2048 and 4056 Mbit/sec not to be sent only locally to a recorder like a Mark5 or a dBBC.

A 10 minutes stream was successfully recorded in Bologna with Vlbistreamer at 4 Gbits/sec on a RAID5 single disk configuration.

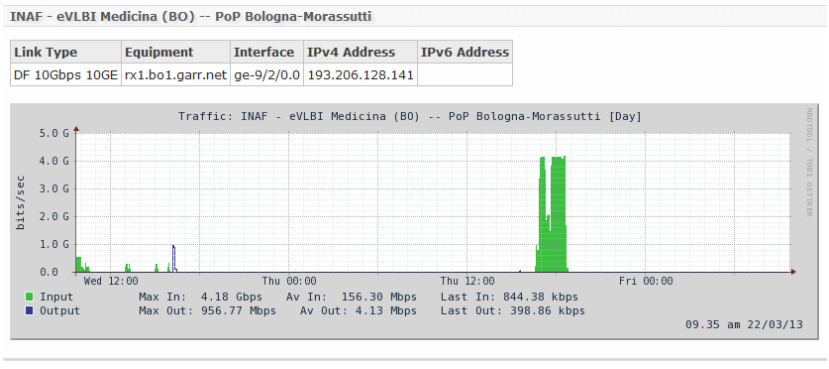

Figure 18: Data streamed from Medicina to Bologna

5 Conclusions

The final goal of our tests with different filesystems and storage architectures was to understand if it were possible adopting a standard storage system, based on a commercial RAID board, to capture and save the telescopes data streams.

While using XFS filesystem we can read and write at a speed of 8Gbit/s on a RAID with 8 or more disks. When stripping the data with LVM on 3 Raids we get a more constant speed, but we do not obtain significant improvements.

Lustre filesystem is not a good choice for our work. Vlbistreamer seems to work well on a cluster of storage systems using Lustre, but provides poor results on a single computer, also with 3 independent Raids volumes.

In our test Vlbistreamer seems to works fine at 8 Gbit/s, but only when the data stream is generated by the same computer. The speed definitely goes down when we send data via network, however this network overhead should be investigated in depth. Although the performance of Vlbistreamer are lower working through the network, we have been able to work securely at 4Gbit/s saving the stream of data send by Fila10 board from Medicina antenna to Bologna.

All these tests were put up using the current vlbistreamer software that split the stream in a lot of files also when working on a single volume. It could be useful to implement an option to verify the packets coherence and to write a single file per stream. In this way we will capture a dBBC stream in a file ready for the correlator.

References

[1] IOzone. http://www.iozone.org.

[2] JIVE. http://www.jive.nl/nexpres/.

[3] HAT Lab. http://www.hat-lab.com/hatlab/products.

[4] LSI. http://www.lsi.com/channel/products/storagecomponents/pages/3waresas9750-8i.aspx.

[5] Tomi Salminen. https://code.google.com/p/vlbi-streamer/.

[6] Matteo Stagni, Francesco Bedosti, and Mauro Nanni. Analysis of high capacity storage systems for e-vlbi. Technical Report 458/12, IRA - INAF, 2012.

[7] Wikipedia. http://en.wikipedia.org/wiki/lustre.