Distributed Storage on Networks of Linux PCs Using the iSCSI Protocol

Andrea Bianco *, Jorge Finochietto †, Marco Modesti *, Fabio Neri *
* Dip. di Elettronica, Politecnico di Torino, Italy, Email: {firstname.lastname}@polito.it
† CONICET - Universidad Nacional de Cordoba, Argentina, Email: {jorge.finochietto}@ieee.org

Abstract— **iSCSI is a protocol standardized by IETF that runs on a TCP/IP network to transfer standard SCSI commands. In this paper, we evaluate the performance of iSCSI using standard PCs running a software implementation of the protocol, with the aim of assessing the performance of low-cost distributed storage solutions. First, we compare the performance when using a local disk with those of a virtualized disk connected through a standard Gigabit Ethernet LAN during file I/O operations. Then, we emulate the characteristics of a WAN/MAN setup by using a software emulator within the Linux kernel to generate delays and packet losses, and evaluate the ability of iSCSI to offer Disaster Recovery solutions over high-speed long-distance links. We found that, even if TCP can be tuned to support links with large bandwidth-delay products, iSCSI does not reach good performance due to its windowing scheme that is not large enough, in the used software implementation, to support long-distance links. Finally, we test the iSCSI behavior when other types of traffic share network links with the iSCSI protocol.**

I. Introduction

The availability of high-bandwidth network infrastructures and of low-cost high bit-rate devices and network cards has enabled interesting new applications of distributed storage and storage virtualization techniques for Business Continuity and Disaster Recovery.

Internet SCSI, or iSCSI, is an IETF standard [1] that permits to transmit SCSI commands over existing TCP/IP-based networks. The iSCSI protocol is based on two components: the iSCSI initiator, which resides on a client computer, and sends commands to the iSCSI target. The target performs the work requested by the command sent by the initiator, and sends a reply back. All communications take place via TCP/IP. Recently, there has been a lot of interest in the research community for the role of iSCSI, as it provides lower overall cost and higher scalability than current Storage Area Networks that typically rely on the Fiber Channel technology. Moreover, the characteristics of TCP/IP networks are better understood, and using the standard Internet protocol would ease management tasks due to the convergence of network technologies, since there is no need to run networks based on different technologies.

In this paper, we present an experimental analysis of iSCSI performance: we compare and contrast iSCSI access speed when running the storage transfer over a high-speed 1Gbit/s Ethernet link in our labs with the access speed of a local disk. Furthermore, we evaluate the performance of iSCSI emulating the characteristics of WAN links such as delay and packet loss to understand the potential use of iSCSI for disaster recovery applications when a backup copy of data is remotely stored to increase data resilience to faults. Finally, we show the benefits that can be obtained by reserving bandwidth for iSCSI traffic when the storage and the data networks are sharing resources, i.e. they are run over the same layer 2 technology. During the analysis of iSCSI, we provide specific insights into key performance characteristics that would affect the backup of data for Disaster Recovery operations.

Relatively little attention has been paid up to now by the scientific community to the performance of iSCSI. An analytical model of iSCSI Storage Area Networks was presented in [2].

The remainder of the paper is organized as follows. In Sec. II we describe our experimental environment, including the PC configuration and measurement tools. Sec. III details our performance analysis of iSCSI, including comparison with local disk performance, performance over an emulated WAN and the interaction with TCP/IP data traffic. In Sec. IV, we provide concluding remarks and describe future tests we intend to run to better assess the iSCSI capabilities.

II. Test-Bed Configuration

A. Hardware

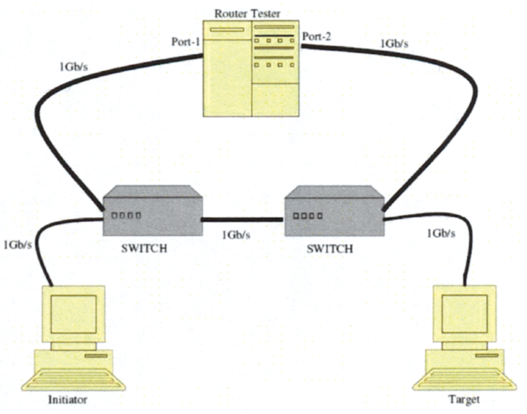

Fig. 1. Test-Bed configuration

Fig. 1 depicts the reference scenario of our tests. The iSCSI initiator is a PC with an AMD Athlon 64 X2 4200+ processor and 2GB of RAM. This PC is equipped with one Intel Gigabit Network Card Dual Port over 4X PCI Express Bus. The targets are PCs with an AMD Athlon 64 3200+ processor, with 2GB of RAM and one Intel Gigabit Network Card. Each machine is running Fedora Core 6 Linux kernel 2.6.18; PCs are equipped with a SATA Seagate Barracuda 7200.9 hard-disk, with 120GB of disk space, 7200rpm and roughly 300MB/s of access bandwidth. An Agilent NX2 Router Tester, capable of generating and measuring Gigabit Ethernet traffic at line rate, is attached to the two Gigabit Ethernet switches.

B. *iSCSI Software*

During our tests, we focus our attention on software implementation of iSCSI. To this aim, we select two open-source projects that provide good performance and regular updates: Open-iSCSI [3], for the initiator side, and iSCSI Enterprise Target [4] for the target implementation.

C. *Measurement Tools*

The *Iperf* benchmarking utility [5] was used to analyze the raw performance of a TCP network. *Iperf* was chosen for the large availability of tuning parameters, such as TCP_window and TCP_segment_size, and for its capability to report bandwidth, delay jitter and datagram loss figures.

The *dd* block device measurement tool, available in standard Linux distributions, was used to gather data from low-level transfers between block-devices. In our Linux Distribution, *dd* makes available statistics like throughput and amount of transferred blocks.

The *Bonnie++* benchmarking tool [6] was used to analyze the performance during a sequential file I/O, that allows us to compare iSCSI's performance when running a file-system on top of it.

The *IOZone* file-system benchmarking tool [7] measures performance for a wide range of file operations. We will focus on the effects on performance of different data request sizes during write operations.

III. Performance Result

A. *Local Disk Performance*

In our first experiment we measure the performance of the SATA disk inside the target PC. The data size used is up to 4GB, to reduce the cache effects of RAM and file-system, and timing of every operations include buffer flushing. We ran five test for each experiment, and averaging results out, we found that the SATA disks have an average throughput of about 45MB/s when writing with Linux file-system *ext3*, and roughly 50MB/s with *dd*; the average performance while reading is of almost 60MB/s in both cases. The write performance with file-system are worse because the file-system used is journaled; thus, a write command has to update also the metadata and the journal. These values were expected given the used disks, which are not high-performance storage devices. Moreover, the *IOZone* tool highlights the independence of write commands from the *RecordSet* value, which determines the size of the atomic write operation. As shown in Fig. 2, even when varying the value of the *RecordSet* parameter, constant performance are obtained regardless of the file size; this trend was also confirmed by using the *dd* tool with different block sizes.

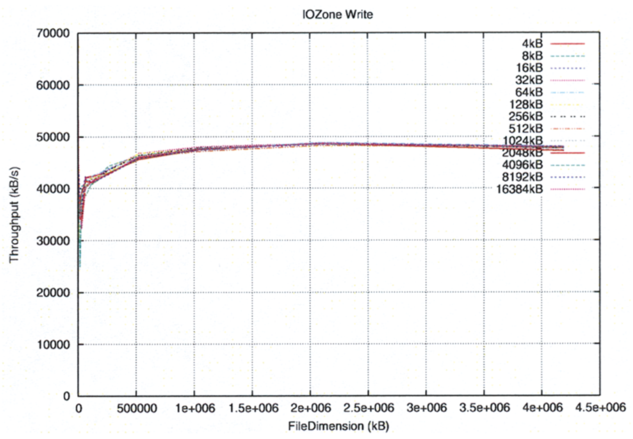

Fig. 2. *IOZone* Write operation performance for different *RecordSet*

B. *iSCSI Performance*

To evaluate iSCSI performance, we made available on the iSCSI target a disk over a Gigabit Ethernet LAN using two different virtualization modes offered by the software on the target: *fileio* and *blockio*. In the *fileio* mode, the page cache provided by the kernel is used to map the disk into memory on the target side; instead, in the *blockio* mode, commands are executed directly on the disk, by-passing the memory. Write performance of both modes are roughly equal to 45MB/s, similar to the case of the local disk, but in the *fileio* mode we had to use partitions aligned to 4KB boundaries. Indeed, mapping the disk in RAM using the paged cache that has a 4KB default size, implies that if the first sector of the partition is not a 4K-multiple, a read command of a block is performed by 2 physical read operations from the device, thus worsening performance. Finally, the overhead introduced by iSCSI is more significant when files are of smaller size; indeed, the worst performance are observed with small-size files.

Read performance are different in the two virtualization modes: in *blockio* mode, the throughput is roughly 50MB/s, whereas in *fileio* mode a throughput of 60MB/s is achieved, as when using the local disk. To obtain the best performance in the above described test, we must tune the value taken by the *readahead* parameter, that populates the page cache with data from a file so that subsequent reads from that file will not block the disk I/O. Unfortunately, this tuning is system-dependent; indeed, with other PCs the optimal value of *readahead* is different from the one we use in this scenario. In our test, the best values were 64 blocks (one block being of 512byte) for devices virtualized with *fileio*, and 16384 blocks when using *blockio*.

C. *iSCSI Scalability*

Considering that iSCSI performance are satisfactory with one iSCSI target, we evaluated the iSCSI scalability when using more targets from the same initiator. To this aim, we use a RAID-0 system on the initiator side, composed by iSCSI disks from other PCs. Data are striped and distributed among the available disks; as a consequence, read and write performance should improve. Fig. 3 shows write and read performance of a RAID-0 composed by local disks compared with a RAID-0 system of iSCSI disks. Read and write performance increase linearly with the number of disks in both cases, and iSCSI performance is close to the performance obtained with the local disk. In tests with more than 2 iSCSI disks, we use 2 physical ports with bonding on the initiator PC. Thus, the achieved aggregated bandwidth is 2Gb/s, enough to test a RAID-0 system with up to 4 disks hosted in 2 different target PCs with a single 1Gb/s port.

These results show that the access performance for a disk local to the PC running an applications can be largely improved by a storage virtualization approach that stripes data over a high-capcity network.

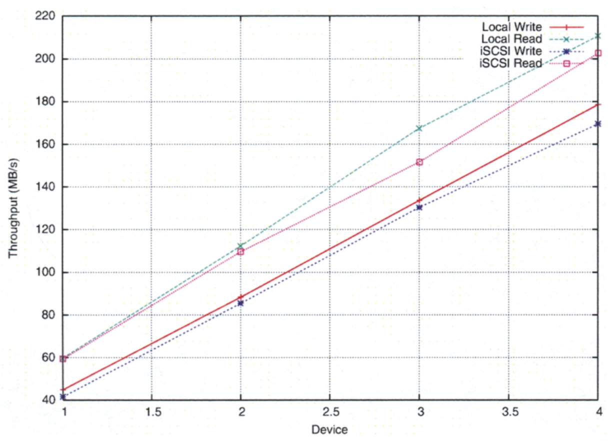

Fig. 3. Performance of RAID-0 system

D. *iSCSI on WAN/MAN*

Previous results were obtained using a LAN with a single switch, in the absence of losses and with negligible delays. Wide Area Networks (WAN) or Metropolitan Area Networks (MAN) are characterized by larger and variable delays, and by potentially significant packet losses. To emulate this condition to test iSCSI performance in a WAN/MAN environment, we used a Linux tool called *Netem* (Network Emulator), controlled by the command line tool *tc* which is part of the *iproute2* package of tools. Through the use of this software emulator, we artificially add to our physical link variable or fixed delays and random packet losses to evaluate the performance of iSCSI on this type of network.

Fig. 4 shows the performance of TCP/IP and iSCSI protocols with increasing average delay, the delay distribution being uniform with a jitter of 70% of the average value. The TCP/IP curve refers to the performance of host-to-host data transfer (with no disk involvement). iSCSI was tested in the *fileio* mode described above, and in the *nullio* mode, in which the target does not transfer data to the disk (writes are acknowledged when data is written into the target memory, and reads generate random data with no real access to the disk in the target PC). The performance of TCP/IP was measured with *Iperf*, while iSCSI *nullio* was tested with *dd* and iSCSI *fileio* with *Bonnie++*. When the RTT increase, as expected, the performance of TCP/IP decreases and, as a consequence, also the iSCSI performance suffers.

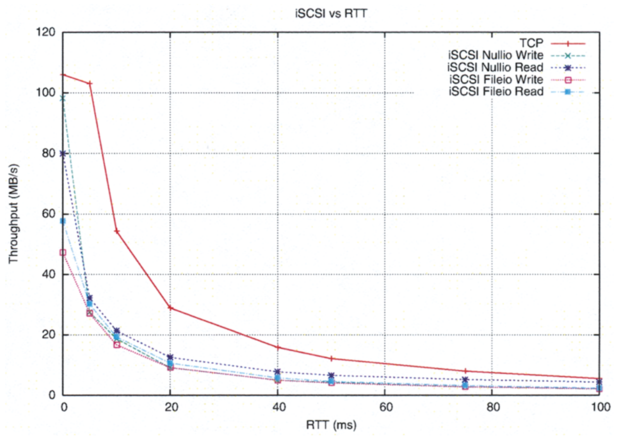

Fig. 4. Performance vs RTT

The first impression is that the bottleneck is represented by the TCP/IP window protocol. Thus, we tuned the TCP protocol parameters to improve the performance over a large bandwidth×delay product (BDP) network. The TCP receiver window size, that determines, through the flow control algorithm, the maximum amount of data a transmitter can send without waiting for an acknowledgment, must be scaled up, to avoid throughput limitation due to the increased RTT. This implies also increasing the memory devoted to each TCP connection.

Fig. 5 reports the performance when increasing the TCP window at the receiver. It can be seen that, with a 16MB window, TCP/IP can reach almost 100% of network utilisation for the emulated delay. However, iSCSI can't reach optimal throughput even in *nullio* mode, demonstrating that disk performance are not the bottleneck in these tests. Indeed, the iSCSI window protocol is the limiting factor in this case. The software implementation permits to set up a window size up to 256Kbyte, not enough to operate in 1Gb/s networks with delays larger than 10ms.

Other tests were made to evaluate the performance of iSCSI when packet loss is present. Differently from a Fiber Channel network, where the loss probability is zero and congestion may create only a delay increase, in an Ethernet network packets may be dropped in case of congestion.

Fig. 6 reports the behavior of TCP/IP and iSCSI protocol when packet losses occur with no additional delay inserted. In this case, the sender can compensate data loss by re-retransmitting the lost data. However, TCP reduces the window size at the transmitter limiting the throughput. Indeed, with a packet loss of just 5%, performance reduction is dramatic, being close to 90%.

E. *iSCSI with other TCP/IP traffic*

During the previously described tests, the Router Tester device was turned off. Thus, only iSCSI traffic was transported

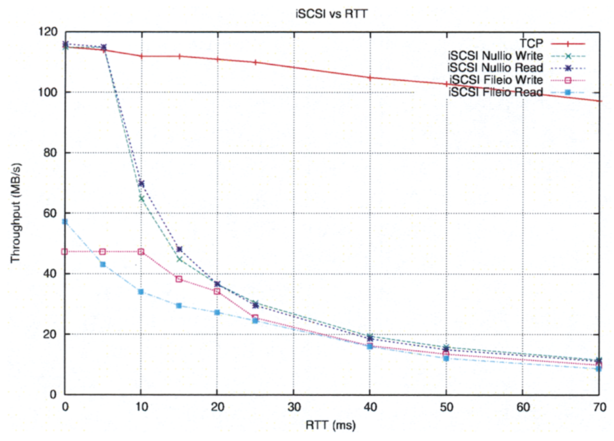

Fig. 5. Performance vs RTT with large TCP window

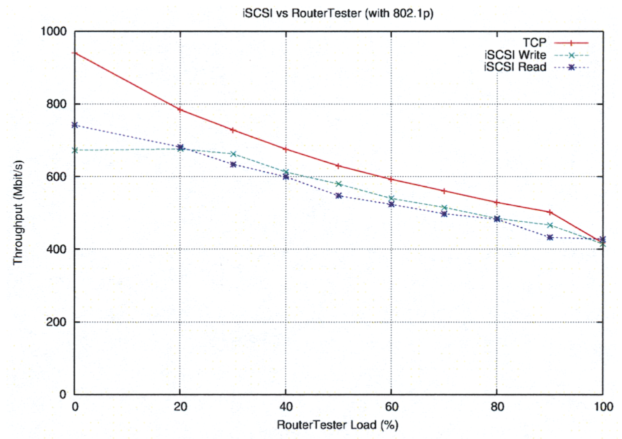

Fig. 7. Performance vs RouterTester load

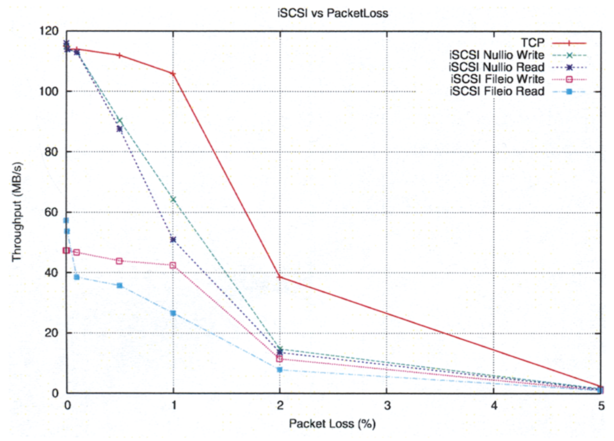

Fig. 6. Performance vs Packet Loss with large TCP window

over the network and the full available bandwidth was devoted to storage traffic. In the next tests, we want to evaluate the behavior of iSCSI when other data traffic exist on the same network infrastructure. A data flow from portA to portB of the Router Tester is generated; thus, the 1Gb/s link between the two switches is the system bottleneck, where iSCSI and data traffic share the bandwidth.

Fig. 7 shows, as a function of the Router Tester load, TCP and iSCSI RAID-0 system performance, the RAID-0 system being composed by 2 target devices. Since two TCP/IP flows share the link, when the Router Tester generates 1Gb/s of data traffic, the resource sharing is fair, and, as a consequence, each connection reaches a data transfer rate of 500Mb/s.

This behavior may not be acceptable for storage traffic sharing the same infrastructure with best-effort data traffic. To avoid this fair sharing, which may penalize too much storage traffic performance, we may configure the network to support VLANs and assign to the iSCSI VLAN a larger weight than the weight assigned to best-effort data traffic. However, we were unable to test this situation, since our Ethernet switches do not support VLAN management. We thus used the IEEE 802.1p protocol instead, which permits to manage the priority level of Ethernet packets in switch queues. The protocol supports a 3-bit priority tag in the Ethernet header, therefore up to 8 different priority levels can be defined, level-0 being assigned to best-effort traffic, with no delay guarantees, up to level-7 left for network control traffic. In our tests, we assigned level-7 priority to iSCSI traffic and we map the RouterTester traffic to level-0 traffic. Fig. 8 shows the performance reached by iSCSI and TCP/IP in this configuration. Even if the RouterTester traffic increase, as expected, the TCP flow between the initiator and the target can reach high throughput, and the iSCSI RAID-0 system obtains the same performance when no losses in the network exist, regardless of the data traffic load. Exploiting the priorities permits also to reduce the delay between the initiator and the target in case of switch congestion. This solution can be adopted also in non-manageable switches, provided they support the 802.1p standard.

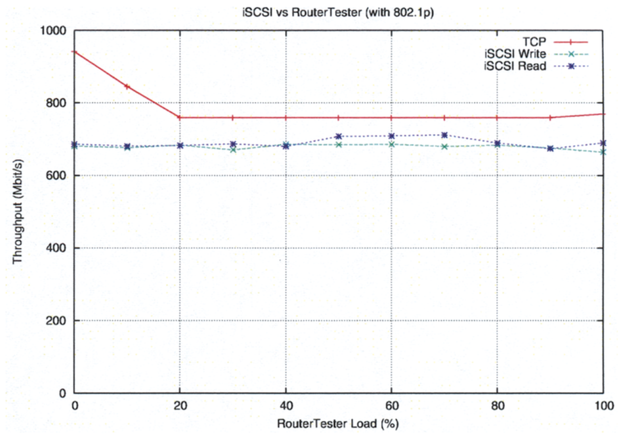

Fig. 8. Performance vs RouterTester load with 802.1p

IV. Conclusion

In this paper, we report some measured results on iSCSI performance using standard PC hardware and software implementation of protocols. No performance impairments were observed in a LAN environment with respect to the raw performance obtained when running the system on a local disk. When emulating a MAN/WAN environment, i.e., increasing RTT, or increasing the packet loss porbability, the iSCSI protocols does not obtain satisfactory performance in networks with large bandwidth×delay products. This problem should be tackled to efficiently use iSCSI over real networks for disaster recovery or remote storage applications. One possiblity may be to proxy data at intermediate points between the initiator and the target.

In our future work we will further analyze the reasons for, and the remedies to, the iSCSI performance degradation. We also want to test iSCSI performance when using specialized hardware like the Host Bus Adapters (HBA), or network cards with TCP Offload Engines (TOE) on board. Finally, we plan to measure the performance when more initiators are connected to one target and analyze how iSCSI manages conflicts when using distributed file-systems.

ACKNOWLEDGMENT

The authors wish to thanks the TOP-IX (TOrino Piemonte Internet eXchange) consortium for their support and Diego Feruglio and his group at CSI Piemonte for many fruitful discussions on storage virtualization. Finally, this work was performed in the context of the FEDERICA project, funded by the European Commission.

REFERENCES

[1] IETF, RFC7320, `http://www.ietf.org/rfc/rfc3720.txt`
[2] C.M. Gauger, M. Köhn, S. Gunreben, D. Sass, S. Gil Perez, "Modeling and Performance Evaluation of iSCSI Storage Area Networks over TCP/IP-based MAN and WAN networks", *Proceedings Broadnets 2005*, Oct. 2005, pp. 850-858
[3] Open-iSCSI project, `http://www.openiscsi.org/`
[4] iSCSI Enterprise Target, `http://iscsitarget.sourceforge.net/`
[5] Iperf tool, `http://dast.nlanr.net/Projects/Iperf/`
[6] Bonnie++ file-system benchmarking tool, `http://www.coker.com.au/bonnie++/`
[7] IOZone file-system benchmarking, `http://www.iozone.org/`