IP SAN – From iSCSI to IP-addressable Ethernet Disks

Peter Wang
Intransa, Inc.
Peter.wang@intransa.com

Robert E. Gilligan
Intransa, Inc.
robert.Gilligan@intransa.com

Henry Green
Intransa, Inc.
henry.green@intransa.com

Jeff Raubitschek
Intransa, Inc.
Jeff.raubitschek@intransa.com

Abstract

The initial iSCSI products provide a means to connect FC SAN islands across IP networks. This paper describes the implementation of an IP-SAN where the disk subsystem is a virtual array of individually Ethernet attached IP-addressable disks. By replacing the conventional peripheral bus and loop interconnects with a switched Gigabit Ethernet network, the virtual disk array scales continually and dynamically with the simple addition of Ethernet switches and disks, as well as system-wide disk sparing and inherent high availability. In fully exploiting the IP and Ethernet technologies and user knowledge, this architecture pushes the IP SAN evolution toward a truly scalable, manageable, yet flexible and cost-effective data storage system that will be a seamless part of the networked infrastructure.

1. Introduction

Much industry effort has gone into defining the iSCSI standards [1] and developing associated products [2]. The initial application of iSCSI products is in bridging the IP and the Fibre Channel (FC) worlds. The industry expects the long-term payoff to be a much lower cost IP SAN that will fill the gaps left open by the expensive and complex FC SAN implementations.

The full benefit of IP SAN [3-7] is well beyond the projected low acquisition cost. In this paper, we discuss an IP SAN architecture and implementation that exploits the capabilities, trends and wide spread knowledge base of IP and Ethernet networks to form the framework for building intelligent IP SANs. These SANS will deliver inherently scalable, distributed virtual storage services as a native part of the IP infrastructure that can be managed centrally.

By deconstructing the conventional peripheral bus and loop interconnects based RAID subsystem structure, this IP SAN architecture utilizes the standard gigabit Ethernet (GbE) network as the interconnect fabric and individually IP-addressable disks to achieve fully virtualized storage arrays.

2. IP-SAN Top to Bottom

The first generation iSCSI storage arrays are primarily existing FC RAID arrays with GbE interface and iSCSI target support or a NAS subsystem with the addition of an iSCSI target. We extend the basic IP connectivity concept inside the storage subsystem to create a SAN architecture that employs IP from the application hosts to the storage array controller and then to the IP-addressable disks.

2.1 Motivation

Putting disks directly onto the Ethernet network and making them IP addressable yield a number of benefits. The use of switched Ethernet breaks the limitation of internal busses and loops of the conventional storage arrays. One can extend and expand the switched network easily and quickly while riding the price/performance improvement trends of Ethernet.

All of the IP disks are reachable by the storage controllers that are on the same IP network. As such, the IP disks form a single storage pool and disk sparing becomes network-wide and is not limited to a specific RAID controller. Storage capacity can be expanded and reconfigured by plugging additional IP disks into the network without impacting the operation of the rest of the virtual arrays. This offers the benefit of incremental storage growth of the conventional modular storage arrays without the associated high management overhead.

Similarly, since the association between a storage controller and the disks are formed logically and dynamically over the network infrastructure, the number of storage controllers can be expanded independently based on the needs of the application server farm. Any storage controller on the network can assume the responsibility of another failed storage controller. Thus, additional levels of fault tolerance and high availability that have traditionally been the domain of high-end FC storage subsystems are implicitly available now at a much lower cost.

2.2 The IntraStor™ Architecture

The IntraStor™ architecture is a 3-tier architecture comprising the application hosts, the storage controllers and the IP disks (see Figure 1). The IP/Ethernet networks that connect the storage controllers to the application hosts and to the IP disks are logically separate, but can be physically constructed as a single infrastructure that's partitioned into VLANs.

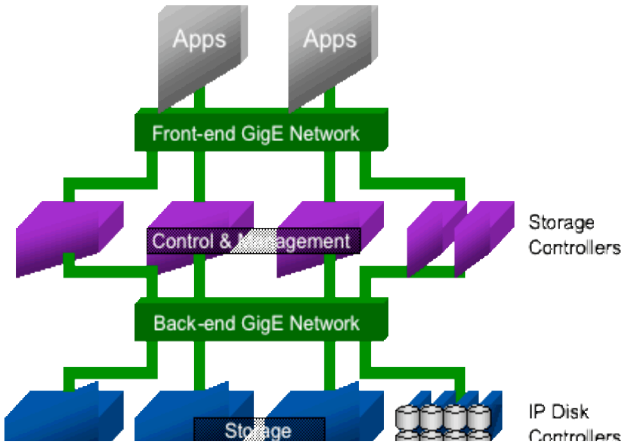

Figure 1 - 3-tier IntraStor™ Architecture

The application hosts access the storage controllers using iSCSI, or other network file access protocols such as NFS and CIFS. The storage controllers access and manage the IP disks using a block oriented transport protocol named xBlock. Each IP disk acquires its IP address via DHCP. A device discovery capability is provided.

The storage controllers form a loosely coupled cluster. Each can perform the same functionalities and thus can assume all or some of the responsibility of a peer for fail-over or load-balancing. In addition, since all of the storage controllers are IP servers, advanced capabilities such as long distance mirroring and remote replication can be achieved natively within the IP network without specialized gateways and extenders.

2.3 Design Considerations

This architecture allows for the offloading of storage functions to each disk node, and the distributed processing of some RAID functions at each node. For example, distributed RAID-5 functionality allows for complete offload of parity calculations from the front-end controllers. Dividing storage functionality allows for abstraction of the physical storage medium, thereby isolating the front-end controllers from specific protocol-related details of the medium.

System flexibility is achieved by abstracting the physical storage medium protocol from the IP storage protocol. This allows the IntraStor™ system to use parallel ATA, Serial ATA, SCSI, and other media protocols without affecting the system architecture.

3. IP Disk Controller

The IP Disk Controller (i.e. IPDC) gives a storage medium (disk) an IP presence by providing low latency protocol conversion between a media-generic storage protocol (xBlock) and a disk protocol such as ATA at gigabit line speed. The primary IPDC design goal is to strike a balance between providing a lightweight, inexpensive protocol processor and a performance-oriented processor that allows for the offload of both storage and RAID functionality.

3.1 IPDC Architecture

The IPDC core architecture is divided into four major functional blocks – the network interface, the disk interface, the Packet Processor and Command Execution Engine (see Figure 2).

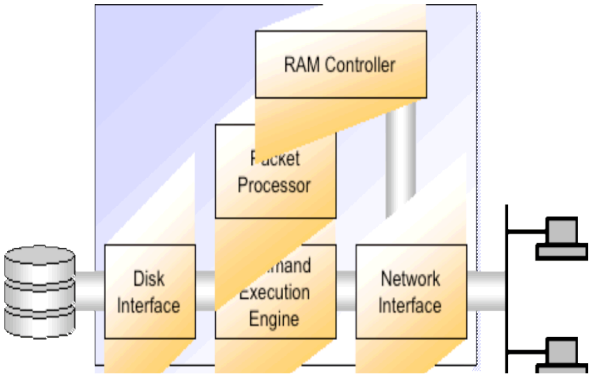

Figure 2 – IPDC Architecture

The network interface provides a GMII MAC that connects to an external PHY chip along with an interface to an internal RAM controller. The network interface allows for packet capture and queuing at line speed.

The disk interface is a direct one-to-one connection to the disk, regardless of specific physical protocol. Since there are no other devices on the connection, the full bandwidth of the physical disk protocol is available. In the case of ATA disks, there are no slave devices to consume available bandwidth. Specific disk interface features include Command Corruption Protection, which

ensures command integrity over the ATA interface (not native to the parallel ATA interface protocol), and Command Replay, which reduces command issue overhead.

The Command Execution Engine (CEE) contains the datapath and control logic for executing in-band data commands as well as out-of-band configuration and status-related commands. The CEE also contains a parity function for offloading RAID parity calculations. The Command Execution Engine has a sustained throughput exceeding the peak throughput of current disk technologies.

The Packet Processor (PP) serves as a front-end command fetch unit to the CEE. The PP also handles out-of-band IP packets such as ARP packets. The Packet Processor scans the incoming packet queue for supported protocols and schedules xBlock execution within the CEE.

3.2 IPDC Implementation

While ATA disks can achieve a specified maximum transfer rate (i.e. 100 MB/s) with occasional bursts, average transfer rates are typically between 30MB/s and 35MB/s. It would be less than optimal to pair one disk with a GbE link, since each GbE link supports transfer rates of about 125MB/s (minus header overhead and inter-frame delay).

To balance the disparity of transfer rates seen on each side of its interface and help saturate the Ethernet link, the IPDC design shares one GbE port per pair of IPDC cores. Each core represents one instance of the outlined IPDC architecture. The port not only services two IP addresses, but also services two MAC addresses. Received frames are issued to both cores, and each core must arbitrate for use of the transmit channel.

The IPDC supports jumbo frames of up to 8 KB as well as streaming operations, which include reading up to a 127 KB stream from the disk in one xBlock command.

4. The xBlock Protocol

xBlock is the protocol spoken between the storage controller modules (SCMs) and IPDCs. The network model envisioned consists of one or more SCMs connected with multiple IPDCs over a switched gigabit Ethernet network as shown in Figure 1. Each SCM may have multiple links to the switch for fault tolerance and to achieve greater throughput. The network is assumed to be dedicated for exclusive use by the storage system, which frees the protocol from the need to implement authentication or confidentiality. In practice, this assumption is easily satisfied by dedicating a switch or VLAN. This assumption can be eliminated in the future by incorporating IPsec support.

The largest constraint on the structure of the xBlock protocol is that it must be able to be implemented in hardware state machines (using FPGA or ASIC) on the IPDC to achieve high throughput, low latency and inexpensive chip implementations. This requirement expresses itself in the simplicity of the xBlock protocol.

4.1 Protocol Design

Communication between the SCMs and IPDCs strictly follows the client/server model. xBlock is a simple datagram protocol layered directly above IP so that it could be used in a routed IP network. In future revisions, it could be cast as a UDP protocol. The SCM is always the client (initiator) and the IPDCs always operate as the server (target). The IPDCs are essentially "stateless" with respect to the xBlock protocol: they transmit only in response to requests sent by the SCM and maintain no state information about requests once they are completed.

xBlock provides the channel for both control and data operations between the SCM and IPDCs. Data operations consist of disk block read and write commands, while control operations include getting and setting control registers in the IPDCs, performing low-level ATA commands to the drive, and identifying and locating IPDCs on the network.

All requests consist of a single request packet, and every request elicits exactly one response packet, except for disk reads, which may generate multiple responses. These response packets serve as the only acknowledgement for the request; there is no separate acknowledgement message in the protocol. Responses are matched up with their corresponding request via a 32-bit sequence number, which is carried in both packets. The SCM assigns sequence numbers to requests in increasing order.

Disk read and write operations transfer data in multiples of 1k bytes (two 512 byte sectors). Disk read requests for more than 1k of data elicit multiple response packets, each of which carries 1 KB of data. Disk write requests may carry up to 8k of data in a single packet, in units of 1k, with the larger sizes possible only if the network supports "jumbo" Ethernet frames.

The protocol allows multiple requests to be outstanding at a time. The xBlock target (i.e. IPDC) services requests from a FIFO (first-in-first-out) queue, synchronously processing one request at a time in the order received. If the queue is full at the time a request is received, it is dropped. The two types of operations – control and data -- can be interspersed.

A simple credit-based flow control mechanism is employed to ensure that the SCM does not send more

requests than the IPDC's receive buffer can accommodate, and that the IPDC does not over-run the SCMs buffers with its responses. For every IPDC it is communicating with, the SCM maintains two counters representing the number of 1 KB buffers on the IPDC that would be consumed by all outstanding requests, and the number of response packets that would be elicited by all outstanding requests. These values are updated whenever a new request is sent or a response received. When either of these counters goes above a pre-configured value, the SCM refrains from sending.

A fixed timeout retransmission scheme is employed to re-send requests that are lost or whose response is lost. The SCM keeps a copy of every request that has been sent. When the matching response is received, this copy is deleted. When a retransmission timer fires, all requests outstanding for longer than the retransmission interval are retransmitted. Retransmissions continue at this fixed interval until a response is received, or until a retransmission abort interval is reached, at which point the SCM stops retransmitting that request, indicates failure to the initiator of that request, and signals the problem to the management layer in the system.

Most xBlock operations are idempotent, i.e. if a request is retransmitted because its response is lost, it can be executed a second time by the IPDC without ill affect. Special care must be taken to prevent the re-ordering of disk and certain configuration register write operations in the network, since such re-ordering may have an unintended affect. For example, consider the case where the SCM transmits two write requests to a particular LBA, W1 and W2, to the IPDC in that order, but W1 is lost and has to be retransmitted. If the retransmission of W1 arrives at the IPDC after W2 is processed, then the data from W1, rather than from W2, will reside on the disk. These problems can be mostly overcome if the SCM limits itself to having only one write operation to a particular disk LBA or configuration register in progress at a time. To guard against the highly unlikely possibility that a copy of W1 might be delayed in the network and delivered to the IPDC after it has processed W2, the SCM sets a flag in the xBlock header instructing the IPDC to drop the request if its sequence number is less than the largest sequence number recently received. The SCM sets this flag only on write requests that require strict ordering.

5. Performance

IPDC core has been designed as a line-speed protocol bridge with as little latency as possible. Benchmarks have demonstrated that average throughput seen at the IPDC host is very close to the average throughput of the disk configured as a single master in a high performance PC workstation.

Figure 3 and Figure 4 compare the IPDC write and read throughput to that of a high performance PC workstation. The PC workstation measurements were taken on a 1.7 GHz Pentium® 4 system using IOmeter. The same Maxtor Ultra-ATA/100 160GB disk was used for all measurements for both IPDC and workstation configurations.

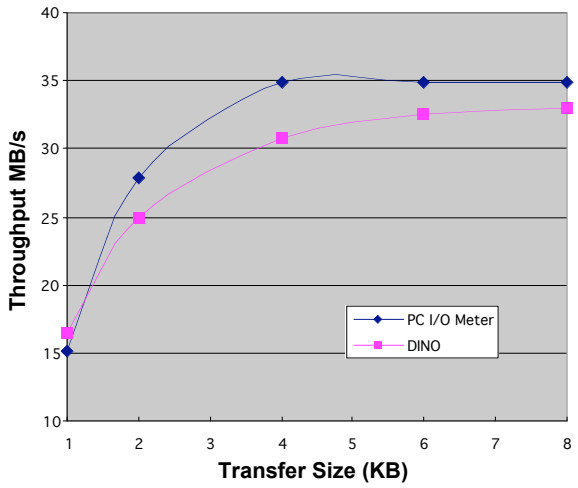

Figure 3 – IPDC vs. PC Write Throughput

As seen from the Figure 3, IPDC's write performance is on par with disk capability and there are several optimizations that can be made to close the gap. Increasing the transfer size above 8KB does not appear to buy additional performance.

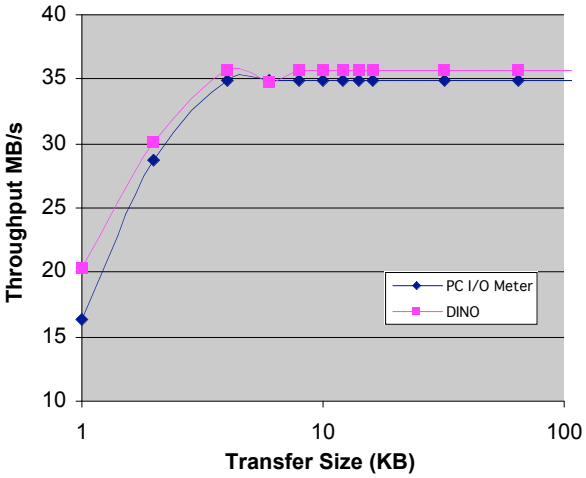

Figure 4 – IPDC vs. PC Read Throughput

Figure 4 show that the IPDC read performance exceeds that of the PC workstation. With streaming-reads, the IPDC can push throughput to over 35.6 MB/s

(8KB transfer size). Much like the case with writes, the throughput levels off at transfer sizes greater than 8 KB, indicating that the disk is reaching its maximum sustained sequential throughput for non-cached data.

Maximum performance was achieved using both jumbo-write and streaming-read features of the IPDC core. Packet overhead (less than 6% for xBlock) is not as significant a throughput bottleneck as is ATA command issuing. Therefore, it is advantageous to issue as few possible commands to the disk, as is the case with jumbo-writes and streaming-reads.

6. Discussions

The IPDC architecture with the xBlock protocol has shown performance on par with a disk attached to a high performance PC workstation. Given the conservative design goals, there is room for improvement in the current design.

Additional processing can be added to the IPDC to optimize performance further with features such as out-of-order execution of disk commands. A greater benefit may be seen when combined with next-generation Serial ATA features such as native command queuing. The IPDC design greatly facilitates the migration to new technologies such as Serial ATA by simply replacing the disk interface block of the IPDC core.

Reads with 8 KB jumbo frames could be implemented to reduce the protocol processing overhead at the SCMs at the cost of slightly greater latency. In a similar vein, streaming-writes could be added to achieve jumbo-write performance where the network doesn't support jumbo frames end-end.

7. Conclusion

The measured access performance of the Ethernet-attached disks showed minimal degradation compared to direct host-attached disks. The switched topology of the IntraStor™ architecture and the point-to-point connectivity to each disk provided by the IPDC offers linearly scalable throughput to the SCMs with all of the advantages of an IP network. The IP addressability of the Ethernet-attached disks enables a whole new dimension of flexibility in constructing distributed, networked storage. IP SANs that leverage such distributed, networked storage with intelligent management such as self-healing will go beyond the simple iSCSI connectivity to achieve storage services as an integral part of the network infrastructure.

Acknowledgement

The authors acknowledge the contributions by the many hardware and software engineers at Intransa who participated in the development of the technology and product. Our special thanks go to Minh Pham, who assisted with a number of the disk performance measurements.

References

[1] J. Satran, K Meth, C. Sapuntzakis, M. Chadalapaka, E. Zeidner, "iSCSI," draft-ietf-ips-iscsi-18, Sept. 28, 2002

[2] P. Sarkar, K. Voruganti, "IP Storage: The Challenge Ahead," IBM Almaden Research Center Publication, 2002

[3] Gibson, G.A., R. Van Meter, "Network Attached Storage Architecture," Communications of the ACM, Vol. 43, No 11, Nov. 2000

[4] R. Van Meter, G. Finn, S. Hotz, "VISA: Netstations's Virtual Internet SCSI Adapter," 8th International Conference on Architectural Support for Programming Languages and Operating Systems, Oct. 1998

[5] E. Riedel, C. Faloutsos, G.A. Gibson, D.F. Nagle, "Active Disks for Large-Scale Data Processing," IEEE Computer, June 2001

[6] G.A. Gibson, D.F. Nagle, W. Courtright II, et. al., "NASD Scalable Storage Systems," USENIX99, June 1999

[7] E.K. Lee, C.A. Thekkath, "Petal: Distributed Virtual Disks," Proceedings of the 7th International Conference on Architectural Support for Programming Languages and Operating Systems, 1996