Storage and Print Solutions for SOHOs

Rajiv Mathews, Ravi Shankar T, Sriram P, Ranjani Parthasarathi
Department Of Computer Science & Engineering, College Of Engineering, Guindy, Anna University, Chennai, India

Abstract—**Small offices/home offices have storage and print requirements that are very hard to fulfill. They require that the methods used are both economical and efficient. Network printers can be very expensive, and to provide a print server for a small network (say 15 stations) is also not economically viable. On the same lines, providing specialized network-attached storage devices too is an expensive option. We aim to solve this problem, by providing a cost-effective, yet efficient solution for storage and print requirements for SOHOs by making efficient use of the IXP425 Network Processor. Internet SCSI (iSCSI) is an official standard that allows the use of SCSI over TCP/IP networks. The recent shift to gigabit ethernet has seen a rapid acceleration in the usage of the iSCSI protocol. Although iSCSI has predominantly seen use in Storage-Area Networks (SANs), nothing in its architecture prevents its use in other devices. We exploit this device independent nature of iSCSI, and that which is seen inherently in SCSI, by providing both storage and print solutions, with the same back bone architecture. We also provide host independent printing, which is to say that, we provide a medium for printing directly without the intervention of a host machine. We use a network processor (IXP425) to implement the solution effectively, making use of its inherent support of networking features, so that we can provide an efficient solution to the task at hand.**

Index Terms—**iSCSI, IXP425, print, small office/home office, storage**

I. Introduction

OVER the years, storage architectures have seen flavors such as SANs and NASs. These solutions though feasible and proved, are not suitable for a Small Office/Home Office environment. Similarly, print solutions such as print servers, referring to simplistic designs such as LAN adapters with parallel ports to specialized routers attached to printers that implement networking protocols are also not suitable for such environments. Effective storage and print solutions for SOHOs must meet the following targets,

a) The network-attached storage device must integrate seamlessly over a network of stations running modern day commodity OSs.

b) The network-attached printer device must appear as a directly attached printer device to the host

c) Enable host independent printing service, that is, to be able to print directly without out the intervention of a host machine.

In designing the device, it is required that the protocol employed should be device independent in order to make the end product generalized, as well as extensible. SCSI is widely known to be a standard that promotes device independence, which means that SCSI can be used with any type of computer hardware. In order to extend this over a network, we build an iSCSI stack over the network processor device.

iSCSI is touted to be a de-facto standard for such purposes. Using iSCSI also has the additional advantage that many implementations of iSCSI initiators for modern day commodity OSs already exist, thus making the device universal. Added with the fact that once we've provided an iSCSI target on the network processor board we will be able to attach other SCSI devices, and provide network access to them too. So in essence we have a network based adapter to a SCSI device.

The device also features host independent printing, that is, the ability to print from a directly connected flash device, without the intervention of a host machine.

II. Related work

A. Contemporary Print Systems

There are many popular solutions for network-based printing. The Internet Printing Protocol or IPP is a standard printing protocol. IPP, similar to most IP based protocols can be used over the Internet. However, since it is built over HTTP, it makes for a more complex and bloated protocol. We judged this unfit for a small office/home office environment as the compromise on performance, due to a taller protocol stack is not justifiable [8].

A print server is a host computer or device that can accept print jobs over a network of stations connected to it. To have a dedicated host machine for a print server is not an economically sound solution. Instead, having an embedded device providing the same interface and functionality is preferred. We try to justify this with our device.

B. *Contemporary Storage Systems*

Storage Area Networks (SANs) and Network-Attached Storage Devices (NASs) have existed for nearly a decade now.

A *Storage Area Network* is a network designed to attach computer storage devices such as disk array controllers and tape libraries to servers. A SAN allows a machine to connect to remote targets such as disks and tape drives on a network for block level I/O. There are two variations of SANs,

1) A network whose primary purpose is the transfer of data between computer systems and storage elements. A SAN consists of a communication infrastructure, which provides physical connections, and a management layer, which organizes the connections, storage elements, and computer systems so that data transfer is secure and robust. The term SAN is usually (but not necessarily) identified with block I/O services rather than file access services.

2) A storage system consisting of storage elements, storage devices, computer systems, and/or appliances, plus all control software, communicating over a network.

Network-Attached Storage (NAS) is the name given to dedicated data storage technology that can be connected directly to a computer network to provide centralized data access and storage to heterogeneous network clients. NAS uses file-based protocols such as the Network File System (NFS) (popular on UNIX systems) or Common Internet File System (CIFS) (used with MS Windows systems). Contrast NAS's file-based approach and use of well-understood protocols with storage area network (SAN) which uses a block-based approach and generally runs over proprietary protocols.

For a small office/home office environment a NAS like architecture is best suited since it blends performance with cost [4].

III. Proposed Solution

Our solution merges the best features of those that are available in literature, in that it provides network access to a centralized storage along with centralized printing capability. With regard to the storage solution that it provides, it emulates a directly-attached storage device by making use of iSCSI. It also extends the principle of iSCSI, which is typically understood to be for storage systems to print systems too [1].

The IXP425 board is the ideal environment to host an iSCSI target on. It provides major networking capabilities that are required for this purpose. For instance, it inherently supports several security features, its network processing engines export common networking functions to hardware and it provides a balance between efficiency and cost, which are critical for SOHOs.

A. *Target Software Features*

The iSCSI target software, which is housed on the network processor, must adhere to certain basic principles that we identified [3,6].

a) *Kernel Modification is to be kept at a bare minimum*. Modifying the source code of the Linux kernel, which is not specialized, may lead to better performance. This may however cause compatibility issues with newer versions of the kernel.

b) *Maintain interoperability with existing systems*. The success of a device like ours depends to a large extent on the interoperability that it provides with existing systems.

c) *Utilize existing device management features*. For instance, the storage solution that we provide must utilize disk management features like the logical volume manager (LVM), as is provided in the Linux kernel.

d) *Meet production environment performance standards*. The device should meet the workloads of a typical SOHO production environment. The target software serves a dual purpose. It interacts with both a printer subsystem and a storage subsystem. The target software is responsible for classifying incoming requests into these two major categories.

B. *Storage Subsystem*

While dealing with the storage subsystem, several design options have been identified. [3] With respect to the kernel interface with which the storage subsystem interacts we consider the following design alternatives (Figure 1).

a) Directly with the SCSI subsystem, meaning to work directly with the device.

b) At the block I/O layer, thereby providing to the initiator devices, blocks as logical units.

c) With the virtual file system and hence provide a regular file as the logical unit.

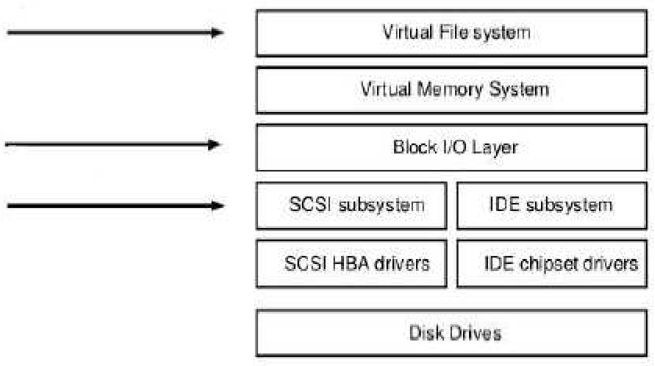

Fig. 1. Layers of accessibility in the kernel

Working with the SCSI subsystem or the block I/O layer would keep us from taking advantage of the rich functionality that the kernel could other wise provide us.

The virtual file system is chosen, to keep the design simple and to work with as many high level interfaces as possible [5]. It provides us with several advantages,

a) This method provides us the advantage of device virtualization. The Linux kernel provides a generic interface to various block devices, allowing homogeneous access to all block devices like SCSI, IDE, ATA disk drives.

b) Working at this level of abstraction also allows us to make use of the page cache, which minimizes disk I/O latency.

c) Also, it allows us to make use of flexible storage management, logical volume manager, and redundancy (Software RAID).

Write Durability is a feature that database programs and file systems rely on in order to maintain data integrity in case system failures. It means that when an initiator device receives a response from the target on a write, it expects that the data has been written to the physical disk drive.

C. *Printer Subsystem*

The printer subsystem is a far more simplified module. The aim of the device with respect to the service of printing was that the printer must appear to be directly attached, rather than a network-attached printer. The initiator concerns itself only with the task of packaging SCSI Command Descriptor Blocks (CDBs) within iSCSI PDUs for transfer over a network to the target. The target will forward printer commands to the printer subsystem. Thus the host machine assumes that the printer is directly connected, due to the fact that the initiator subsystem falls below the SCSI mid layer on the host machines.

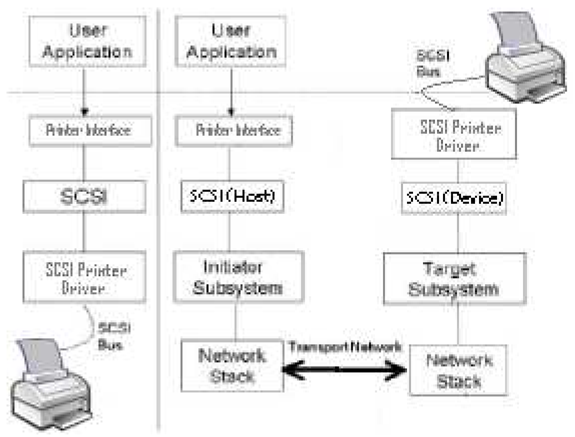

Fig. 2 Printer Architecture Comparison
The left pane shows a standard SCSI printer set up. The right pane shows the SCSI mid layer expanded to transparently include the network.

So the network is transparent to the host machines since it falls logically within the SCSI mid layer. To summarize, the target software will forward CDBs destined to the printer, to the printer subsystem which in turn will simply forward these to the SCSI mid layer at the target.

IV. Work in Progress

We are working with the Intel IXP425 Network Processor which is a versatile and highly integrated single-chip processor that is suited for high performance network-based applications [7]. The IXP425 combines an Intel XScale core with additional Network Processing Engines (NPEs) to achieve wire-speed packet processing performance. The network processing engines are used to offload computationally intensive data plane operations like IP header inspection and modification, packet filtering and error checking, checksum computation and flag insertion and removal. The NPEs have hardware elements each of which are specialized to increase a specific networking task.

The network processor will run a SnapGear Linux distribution operating on a Linux 2.6 kernel. SnapGear is an embedded Linux distribution that represents Linux technology for embedded microprocessors with or without MMUs.

The iSCSI target software that is housed in the network processor is divided into three major blocks.

a) *Read thread*: This thread will receive requests from the various initiator devices.

b) *Process block*: This block will process the requests, determine the nature of the requests, act on the request and formulate a response.

c) *Write thread*: This thread will respond to the appropriate initiator device.

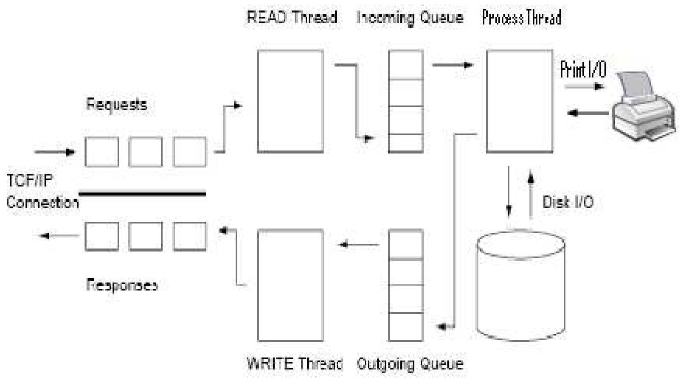

Fig. 3 Architectural Overview
Incoming requests are buffered in the ‘Incoming Queue’, which leads to the process thread. The process thread performs the necessary operation and writes out to the ‘Outgoing Queue’.

The read thread passes to the process block, the SCSI CDB that it has extracted from the received iSCSI PDU. Incoming requests are picked up by this thread and after stripping out the protocol headers is deposited on the ‘Incoming Queue’.

The process block has three logical units. The classifier classifies incoming packets into two flows; those which are destined to the storage device and those to the printer device. The flow that is intended for the printer device is led to the Printer I/O thread, and those for the storage device, to the Disk I/O thread, as shown in Figure 4.

The classifier based on the Operation Code of the SCSI CDB determines which flow a particular incoming packet falls into [2]. The classifier interacts with both the Disk I/O thread and the Print I/O thread.

We work with the virtual file system, so as to benefit from all the abstractions that the Linux kernel provides us. So in this scheme, a traditional file stored on the file system, is the logical unit (LU) that is projected to the initiator.

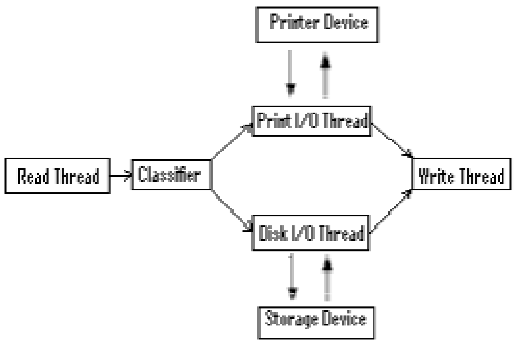

Fig. 4 Design Pipeline

This scheme gives us the freedom to use any kind of hardware for the storage device. This device independence is possible due to high level of abstraction that the virtual file system guarantees us.

The Printer I/O Thread will receive SCSI CDBs that it sends directly to the SCSI Mid Layer. By doing so it simply emulates a directly connected SCSI printer device. iSCSI gives us the benefit of extending it seamlessly over a network. Thus all hosts whose initiators are configured view the printer as a directly attached printer device. The printer I/O thread interacts with the SCSI Mid Layer of the kernel. This way we only interact with the kernel never actually modifying modules, thus preserving the ability to work with newer kernels. Maintaining interoperability with newer kernel versions was an important design goal. The device currently works with SCSI printers.

Another feature that the device provides is host independent printing. As mentioned earlier, this provides a direct printing service from a USB flash device, without the intervention of a host machine. The IXP425 is typically suited for such embedded applications. The IXP425 provides a USB 1.0 port, which along with the USB Mass Storage Class drivers that are provided by the Linux 2.6 kernel enables us to provide such a service. Here it must be noted that the Linux 2.6 kernel tree by it self supports several file systems, nearly 20 officially. This is important to such a device because it enables us to detect and mount several different kinds of file systems. USB Flash devices are typically formatted using the FAT 32 file system, but by using the kernel functionality we aren't limited by that.

As for host independent printing, we provide two options,

a) Either locate meta-data that is present on the flash device that provides information on the files on the flash device that are to be printed. We call this “selective” printing.

b) Or print all the files that are present on the flash device, we call this “dump” printing.

When the USB flash device is detected by the kernel, it spawns a process that handles this task. The process scans through the root directory of the file system of the flash disk, looking for the meta-data file.

V. Future Work

A. *Printer Subsystem*

The current design supports only SCSI printers as it only extracts the SCSI CDBs from the iSCSI PDUs and passes it on to the SCSI mid layer that interacts with the printer. A better and more generalized solution would be to extend this idea to make it work for other types of printers too. This would require the presence of another abstraction layer that will map the SCSI commands set to the appropriate target command set.

B. *Storage Subsystem*

The high level abstractions that are used and the rich functionality that the kernel provides makes the storage system very extensible. Some of the intended future work includes

a) Building an Object Store Drive [9]. An object store drive is a storage device that manages space in terms of objects and offsets within objects rather than logical block addresses(LBA). OSDs are primarily targeted towards shared storage and shared-something multiprocessors architectures. Such systems need high speed devices that work over the network and a network processor based implementation such as ours would be the best-suited for such scenarios.

b) Increasing reliability and bandwidth by accommodating RAID based storage solutions on the device.

VI. Conclusion

Our device holds much promise. Its high flexibility, portability, and simplicity make it a general solution to existing problems faced in small office/home office environments. The centralized storage and printing concept increases data management and device utilization which bring down costs. We have designed it to be extensible so that future improvements in storage and print technologies can be incorporated with minimal effort. Usage of the IXP425 Network Processor makes for a cost-effective and efficient implementation that provides high levels of performance. Host independent printing facilitates printer utilization without machine intervention.

References

[1] J. Satran, K. Meth, C. Sapuntzakis, M. Chadalapaka, and E. Zeidner, “Internet Small Computer Systems Interface (iSCSI),” April 2004, RFC 3720.

[2] M. Chadlapaka, R. Elliott, “Small Computer Systems Interface(SCSI) Command Ordering Considerations with iSCSI”, May 2004, RFC 3783

[3] F. Tomonori and O. Masanori, “Analysis of iSCSI Target Software”.

[4] P. Radkov, L. Yin, P. Goyal, and P. Sarkar, “A Performance Comparison of NFS and iSCSI for IP-Networked Storage,” in the *USENIX Conference on File and Storage Technologies*, San Francisco, CA, March 2004, pp. 101-114.

[5] A. Palekar, N. Ganapathy, A. Chadda, and R. D. Russell, “Design and implementation of a Linux SCSI target for storage area networks,” in the *5th Annual Linux Showcase & Conference.* Atlanta, GA: USENIX, November 2001.

[6] K. Z. Meth, J. Satran , “Design of the iSCSI Protocol,” in the *Mass Storage Systems and Technologies, 2003, proceedings 20th IEEE/11th NASA Goddard Conference*, April 2003, pp. 116-122.

[7] P. Barry, and G. Hartnett, “Designing Embedded Networking Applications,”, Intel Press, May 2005.

[8] R. Herriot, S. Butler, P. Moore, R. Turner, “Internet Printing Protocol/1.0; Encoding and Transport”., April 1999, RFC 2565

[9] A. Azagury, V. Dreizin, M. Factor, E. Henis, D. Naor , N. Rinetzky, O. Rodeh, J. Satran, A. Tavory, L. Yerushalmi, “Towards an Object Store,” in the *Mass Storage Systems and Technologies, 2003, proceedings 20th IEEE/11th NASA Goddard Conference*, April 2003, pp. 165-176.