# SCSI and Distributed I/O

Lansing Sloan

Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550-9900
USA

ljsloan@llnl.gov

Phone 1-510-422-4356
FAX 1-510-423-8715

Revision 1 incorporates some corrections and some of the comments made when revision 0 was presented to the SCSI working group on 95/01/11.

## Overview

We believe SCSI network-attached peripherals can enhance file and storage servers, particularly if more capability is added to SCSI.

We will discuss

1 goals, server requirements, and market possibilities
  1.1 reduce costs
  1.2 retain reliability and integrity
  1.3 scalability
  1.4 other remarks
2 possible SCSI approaches to meeting requirements
  2.1 servers control peripherals
  2.2 processor-and-peripheral transfers
    2.2.1 READ/WRITE
    2.2.2 COPY
    2.2.3 data exchange
  2.3 peripheral-to-peripheral transfers
    2.3.1 COPY and READ/WRITE
    2.3.2 data exchange
3 What LLNL is doing and what we need
4 Contacts for additional information

## LLNL Background

Lawrence Livermore National Laboratory (LLNL) has built and used large-scale file servers since the late 1960's.

Our technologies were transferred to industry (as "Unitree") in the 1980's.

We started, and participate in, the National Storage Laboratory (NSL).

The NSL is a collaboration of several national laboratories, vendors, and universities.

The NSL has developed NSL Unitree and transferred it to industry. NSL Unitree supports network-attached peripherals and dynamic storage hierarchies.

The NSL is developing a High Performance Storage System (HPSS), which adds more capability including scalable I/O and storage (striping).

LLNL's Scalable I/O Facility (SIOF) Project is extending HPSS to use lower-cost technologies (SCSI, Fibre Channel) and to support massively-parallel processors.

## 1. We want better file and storage servers.

File servers are widespread. They provide the ability to share files among many client systems.

There is an established marketplace for file servers.

We want
- reduced costs,
- reliability and integrity, and
- scalable data rates.

Our primary interest is transferring data between computers and peripherals.

- Transferring data among peripherals is less important, though also interesting.

### 1.1 We want to reduce costs.

Most file servers are attached to networks. Most connect directly to, and fully control, their "own" peripherals.

Such servers stage data through server buffers. This increases buffer and bandwidth costs of server processors (and adds transit delay).

If processors did not have to handle all of the data, probably their costs could be reduced, and each server could control more storage and I/O.

Therefore network-attached peripherals should improve cost effectiveness of servers by reducing processor costs.

> Note: If network-attached peripherals cost more than "ordinary" peripherals, they might be more cost effective only for larger server configurations, where processor costs can be reduced significantly.
>
> Note: If I/O requests are for small transfers, the amount of server buffering required is relatively low and cost benefits may be trivial or not exist.

The cost reduction is what makes network-attached peripherals interesting to a mass market.

### 1.2 We want reliability and integrity.

Most file servers are attached to networks. Most connect directly to, and fully control, their "own" peripherals. This helps make them reliable. We want to retain this reliability.

If network-attached peripherals can be commanded by any initiator on a network, the initiator could send commands that damage or steal data. File server vendors could not easily diagnose troubles if they could not rule out such possibilities.

We want a network-attached peripheral to exchange data with ports that can not control it. Today, SCSI standards do not discuss exchanging data without allowing control.

Networks with many, diverse initiators may be especially vulnerable.

- Systems from new vendors may have unknown bugs or weaknesses.
- Updates to operating systems often introduce new bugs.
- Configuration errors can be common in large networks.
- For small, static, or single-owner networks, full control may be unneeded.

### 1.3 We want high (scalable) data transfer rates.

We want many peripherals to be able to work in parallel on a single application I/O request, to provide high total data rates for the request.

We have a "massively-parallel processor" (MPP) and want it to transfer data with file servers fast.

If an MPP has many ports on a highly parallel network, we want many ports to be able to work in parallel on a single application I/O request, to provide high total data rates for the request. We will probably want specific ports to be used for specific data, to optimize memory access and assignment inside the MPP.

- LLNL's Scalable I/O Facility (SIOF) Project is working on these issues.
- We expect interconnect technologies like Fibre Channel and ATM to provide highly parallel networks.

Compared with other requirements, scalability may represent a relatively small marketplace.

### 1.4 Data must be identified properly.

A single application I/O request might be resolved into many transfers. These might occur out of order. Also, there might be concurrent application I/O requests. In effect, each peripheral transfer must be associated with the correct part of the correct application I/O request.

Example

1. An application in a client computer requests reading data from a file.
2. The client system sends the request to a file server.
3. The first part of the data is on one network-attached peripheral and the rest is on a second.
4. Each peripheral sends its part of the data.
5. Each part of the data goes to the correct part of the client buffer.

We will discuss some of the details that could make steps 4 and 5 possible with SCSI.

## We make some architectural assumptions.

HPSS and SIOF follow the IEEE Mass Storage System Reference Model.

In the Model, a peripheral's data transfers are controlled by a "Mover". A file or storage server will have Mover(s) for its peripherals. A processor has some Mover functions.

Movers may negotiate with one another to arrange and identify transfers.

Movers that control network-attached peripherals should make data transfer as directly as possible. When necessary, Movers stage data through buffers and use non-peripheral protocols to transfer data across a network.

## Summary of goals and requirements

Our goal is to use standard commercial file servers that are

- economical,
- reliable, and
- scalable.

We expect to reduce costs by using

- network-attached peripherals and
- SCSI.

We believe servers must effectively control their "own" peripherals, to achieve reliability and integrity in a manner vendors can support.

We want scalability, to allow very high transfer rates.

We think there is a market.

## We think there is a market.

We believe that network-attached peripherals can make servers less expensive, thus competitive.

We believe that there is little or no extra cost or complexity in individual network-attached peripherals to support scalable I/O rates.

We expect the first customers to be research laboratories and that these customers will be interested in high-end network-attached peripherals (such as RAID systems). That is, a file server would control multiple RAID systems.

## 2 Overview of SCSI Approaches

We look briefly at how well some ways to use SCSI meet the goals and requirements.

Approaches to let servers control their "own" peripherals are discussed.

For processor-to-peripheral applications, three approaches are discussed.

- Use conventional two-party SCSI commands (READ/WRITE).
- Use current three-party SCSI (COPY) implementations.
- Transfer data directly between processor and peripheral, but not commands .

For peripheral-to-peripheral applications, two approaches are apparent.

- Have one peripheral COPY with the other acting as a target.
- Transfer data directly between the peripherals, but not commands.

## 2.1 How can servers control their "own" peripherals?

Servers need sole control of their own SCSI network-attached peripherals. Other initiators must not have control.

There are several ways possible to help assure only servers control peripherals.

- Peripherals might have multiple ports, some on a network used for control. Other ports would prohibit control. Only servers and their peripherals would be on the control network.
- Peripherals might be configured to know which initiators are allowed to exercise control. They would check addresses and accept commands only from the configured addresses. The network would prevent address forgery.
- The network might support logical sub-nets (e.g., by letting a port have multiple aliases and restricting certain aliases for use only in certain sub-nets). Note that for this to work inter-operably, possibly both SCSI and (say) Fibre Channel standards would have to specify how such capabilities are provided. The network would prevent address forgery.

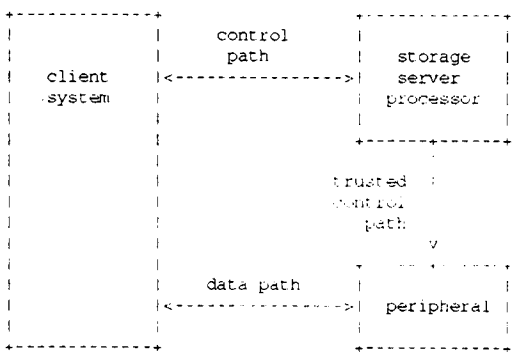
Peripherals must not accept configuration information from bad sources.

RESETs from non-controlling initiators must be acceptably harmless.

These considerations apply to all approaches for transferring data. Some of the approaches require additional considerations.

## 2.2 Processor-peripheral transfers use three major modules.

The diagram shows major modules and paths to enable data transfers directly between a client processor and a peripheral.




### 2.2.1 Processor-peripheral: how READ/WRITE works

A client processor sends a request to a file server.

The server alerts peripherals to allow specific access by specific client initiator(s) Details are on the next foil.

The server tells the client processor what to do.

The client initiator(s) send READ or WRITE commands accordingly.

The peripherals validate the commands before proceeding.

When done, the peripherals send status back to the client initiator.

The client sends the status to the server, so the server knows what happened to the peripherals.

The server alerts peripherals that specific access is no longer allowed.

The server sends final status to the client.

### Processor-peripheral: READ/WRITE server control

The best SCSI mechanisms for alerting a peripheral appear to be third-party RESERVE and RELEASE with extents.

The server uses third-party RESERVE/RELEASE commands.

(New rule) A client initiator can access an extent ONLY if it has a reservation.

- If a client initiator could access any unreserved extent, the server would have to ensure that no extent is ever unreserved. With current SCSI rules for superseding reservations, "reserve fragmentation" could result.
- If a client initiator could access an unreserved extent, it could have improper access after RESETs or power-up conditions.

The server must be able to RELEASE a reservation and know when a client can no longer access an extent, no matter how badly the client behaves. The server must also be able to determine or control the state of the peripheral.

Also see "2.1 How can servers control their "own" peripherals?"

### Processor-peripheral: READ/WRITE pros and cons

Summary: No show-stoppers are evident, thanks to discussion when these ideas were first presented to the SCSI Working Group.

Advantages are marked with "+", disadvantages with "-".

+ READs and WRITEs are mainstream SCSI.
+ TAGs identify requests; CDBs contain addresses and counts.

- A mechanism, such as logging, must be defined or specified so servers can know exactly what commands are sent by clients and what the status results are.
- The reliance on extents limits this approach to peripherals that can support extents. (What about tapes?)
- The dependence on (modified) RESERVE and RELEASE is undesirable.

Note: At the SCSI working group, interesting questions were asked about the possibility of group reservations, so a reservation could be used by any of several client initiators.

### Processor-peripheral: COPY pros and cons

Summary: No show-stoppers are evident.

Advantages are marked with "+", disadvantages with "-".

+ Servers directly control their own peripherals.
+ Extents and RESERVE/RELEASE are not needed to control client access.
+ This approach seems to require the fewest changes to the SCSI standard.

- The LUN is used to identify original requests. This is apt to require a large number of LUNs for each client processor. How do peripherals know when they can forget LUNs?
- Either the client processor must learn which peripheral ports will be initiators for a request, or else it must depend on the LUN as an unforgeable identifier.
- The blocks in the target (client processor) must align with blocks in the peripherals.
- COPY is not widely supported today.
- Target behavior by processors is probably not widely supported today.
- If LUNs are path-specific (see SCC documents) then they must be treated carefully in distributed I/O protocols.
- The client processor, not the peripheral, controls the flow of data.
- The peripherals rely on the target to say when a command completes.

### 2.2.2 Processor-peripheral: how COPY works

A client processor sends a request to a file server.

The client processor acts as a SCSI target. It probably accepts block commands. (Monia)

- The client probably creates a LUN to identify the original request.
- Block addresses map to client buffer addresses.
- Block counts map to client byte counts.

The server sends COPY commands to peripherals.

The peripherals send READ or WRITE commands accordingly.

The client checks command parameters before proceeding.

Upon completion, the peripherals send status back to the server.

The server sends final status to the client.

Also see "2.1 How can servers control their "own" peripherals?"

### 2.2.3 Processor-peripheral: how "data exchange" works

Key idea: "commands" are not sent on the data path.

A client processor sends a request to a file server.

Servers tell passive parties (probably clients) what to expect.

Servers tell active parties (probably peripherals) what data to transfer and how to identify the data (offset, length, transfer identifier).

Active parties send data identification and then either send data or prepare to receive data.

Passive parties receive and check identification and then either process incoming data or else send outgoing data.

Upon completion, servers receive status from their peripherals.

Servers send final status to clients.

Also see "2.1 How can servers control their "own" peripherals?"

## Processor-peripheral: "data-exchange" issues

Mechanisms for sending data identifiers and data must be specified. It may be best to set up a nexus in such a way that DATA IN and DATA OUT work for data. Several ideas have been suggested.

- Squeeze data identification into a command descriptor block.
- Link commands; send data identification with the first command and data with the second.
- Use SCSI messages to carry the data identification.
- Create protocol-specific extensions for the data identification.

  Do such approaches support the notion that "commands" are not used?

If a passive party does not know how many times active parties will establish a nexus, it is not obvious how the passive party discovers completion.

Formats must be specified for

- sending commands to peripherals,
- sending data and data identification, and
- sending status from peripherals.

If data identifiers include offsets and counts explicitly, Movers can segment requests with less negotiation. NSL protocols presently use offsets and counts. We prefer them. However, they lengthen data identifiers.

## Processor-peripheral: "data-exchange" pros and cons

Summary: Since we want to be industry standard, solving the first "con" is key.

Advantages are marked with "+", disadvantages with "-".

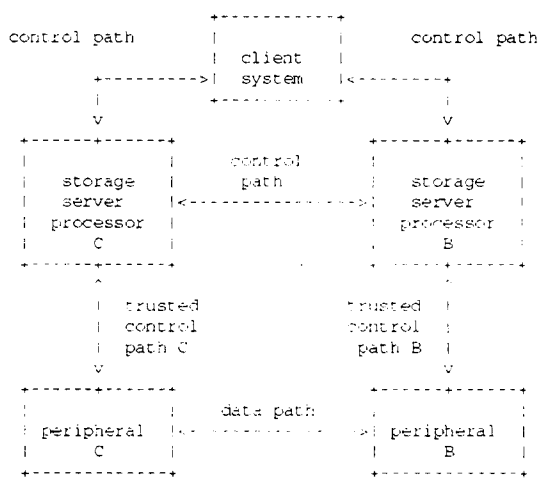
+ Since this approach is not yet specified, we can try to make it satisfy all needs.

- How can a new approach be defined and accepted by industry? How can a new approach remain accepted by industry?

If a specific approach is defined, presumably the lists of pros and cons will become longer and much more specific.

## 2.3 Peripheral-peripheral transfers use five major modules.

The diagram shows major modules and paths to enable data transfers directly between two network-attached peripherals controlled by separate servers.




## Peripheral-peripheral: distributed I/O transfers

Suppose a file is to be copied from one file server (provided by one vendor) to another file server (provided by a different vendor). Neither file server knows how data is arranged within the other server. Each server should control its own peripherals.

A client contacts one file server. The server finds where it has (or will have) the file and returns file parameters, data and control port identifiers, unique identifiers, etc. The client contacts the other file server, passing the information from the first server. The second server finds where it has (or will have) the file. The second server can find the source and sink peripherals and addresses for any part of the transfer. If necessary, (Mover modules in) the two servers can negotiate details. Each server determines final results and returns final status back to the client.

The processor-to-peripheral transfers discussed above can be viewed as a simplification of peripheral-to-peripheral transfers:

- One of the "peripherals" is a client processor memory buffer
- The client, one server, and "peripheral" are a single system.

### 2.3.1 Peripheral-peripheral: COPY and READ/WRITE

The basic idea is that one peripheral receives COPY commands and the other peripheral acts as a target, accepting READ and WRITE commands.

The disadvantage of this approach is roughly the union of the disadvantages of the COPY and "READ/WRITE" processor-peripheral approaches.

Also see "2.1 How can servers control their "own" peripherals?"

### 2.3.2 Peripheral-peripheral: "data-exchange"

If the "data-exchange" approach works for the processor-to-peripheral case, it should work for the peripheral-to-peripheral case, providing that behavior for both active and passive roles is specified. The passive role may be more difficult to implement, since the passive peripheral must interpret the data identifier and it is unclear how to determine when there will not be another nexus.

Also see "2.1 How can servers control their "own" peripherals?"

### Summary of SCSI Approaches

Servers must be able to control network-attached peripherals. This means peripherals must be able to distinguish server initiators from other SCSI ports and must reject inappropriate commands from the other ports. This is true for all approaches.

For processor-to-peripheral transfers, all three approaches are technically feasible. All three have problems to be solved.

For peripheral-to-peripheral transfers, both approaches are technically feasible. Both have problems to be solved.

For any approach, some changes are needed in SCSI standards for SCSI network-attached peripherals to be part of a large file-server market. (E.g., statements that all ports must be treated equally contradict the need for server control.)

### 3 What is LLNL doing?

LLNL transferred Unitree technology to industry. LLNL participates in the National Storage Laboratory (NSL), which transferred NSL Unitree to industry.

- Client software has been made generally available.
- Server software has been licensed to interested vendors.
- NSL Unitree supports network-attached peripherals and dynamic storage hierarchies.

The National Storage Laboratory is also working on a High Performance Storage System, which extends capabilities of NSL Unitree with support for parallel computers and workstation clusters, striping of both I/O and devices for scalable high bandwidth and storage.

- A Parallel I/O protocol specification is being written. It is consistent with the IEEE Mass Storage System Reference Model.

LLNL's Advanced Telecommunications Program has worked with Fibre Channel technology for about 5 years.

- LLNL has a Fibre Channel test bed.

### What is SIOF doing?

The Scalable I/O Facility (SIOF) project is extending HPSS to use lower-cost technologies (Fibre Channel and SCSI), to support massively parallel processors, and to eventually support wide-area access to data.

- This project is funded by the Gas and Oil National Information Infrastructure initiative (a collaboration among the gas and oil industry and some Department of Energy labs) and a Defense initiative.
- We invite industry participation.

LLNL's strengths include experience with high-speed local networks, experience with Fibre Channel, and experience with storage systems (including writing device drivers and modifying operating systems).

- Our experience with SCSI is relatively limited at present.
- We have little experience with creating computer peripherals.

SIOF is creating a test bed to experiment with distributed SCSI I/O protocols.

## What would we like from X3T10?

We want to be able to acquire standard commercial storage products that are more cost-effective, support high bandwidths scalably, and provide reliability and integrity.

We think SCSI, which is a high-volume, low-cost technology, can help.

We want X3T10 to see that SCSI needs more features if it is to enable servers to control network-attached peripherals while allowing data to be exchanged with other SCSI ports.

We'd like X3T10 to work on SCSI distributed I/O protocols.

We welcome ideas on how best to extend SCSI distributed I/O capabilities, and warnings when our ideas are bad.

We invite industry participation in the SIOF project.

## Contacts for additional information

National Storage Laboratory (Unitree and HPSS)

Dick Watson
510-422-9216
dwatson@llnl.gov

Parallel I/O Specification

Larry Berdahl
510-422-4217
berdahl@llnl.gov

Scalable I/O Facility Project

Kim Minuzzo
510-422-2141
minuzzo1@llnl.gov
or
Mark Seager
510-423-3141
seager@llnl.gov

World-wide-web URLs for further information:

http://www.llnl.gov/liv_comp/nsl/nsl.html for NSL
http://www.llnl.gov/liv_comp/nsl/hpss/hpss.html for HPSS
http://www.llnl.gov/liv_comp/siof.html for SIOF
http://www.llnl.gov/LLNL_Home.html for LLNL
http://www-atp.llnl.gov/atp for LLNL Gigabit Networking including Fibre Channel