X3T10/95-159R0

Distributed SCSI I/O: Status and new issue

Lansing Sloan

X3T10 SCSI Working Group
March 7-8, 1995

Mail Stop L-60
Lawrence Livermore National Laboratory
7000 East Avenue
Livermore, CA 94550-9900

ljsloan@llnl.gov
Phone: 1-510-422-4356
FAX: 1-510-423-8715

Other Contacts

National Storage Laboratory:
Dick Watson
510-422-9216
dwatson@llnl.gov

Scalable I/O Facility:
Kim Minuzzo
510-422-2141
minuzzo1@llnl.gov

Abstract

Several approaches to distributed SCSI I/O were presented in X3T10/95-112R1. Exploration of the READ/WRITE approach revealed an additional problem. Within a client system, other applications can access files belonging to the client application. Other approaches do not appear to have the problem. The problem, and two possible solutions, are discussed. We are continuing to analyze issues.

Overview

1 Background
2 The Problem
3 Solutions

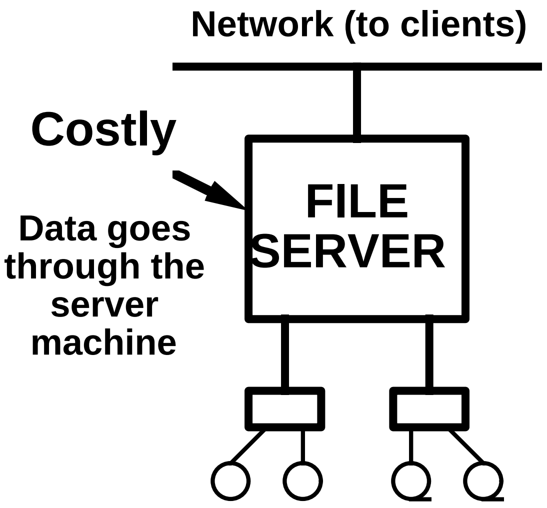
Background: current file servers are costly

Costs

- Large buffers
- I/O bandwidth
- Low capacity each
- Support costs

Slowness

- Latency in buffer



Network (to clients)

Costly

Data goes through the server machine

FILE SERVER

Net-attached peripherals reduce cost

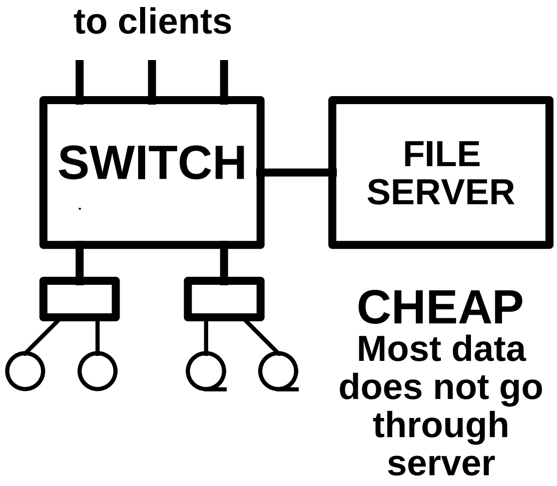
Cheaper server processors

- Data bypasses server processor
- reduced buffer, bandwidth

Fewer server processors

- Each handles more peripherals
- Fewer support personnel

Data flows more directly



to clients

SWITCH

FILE SERVER

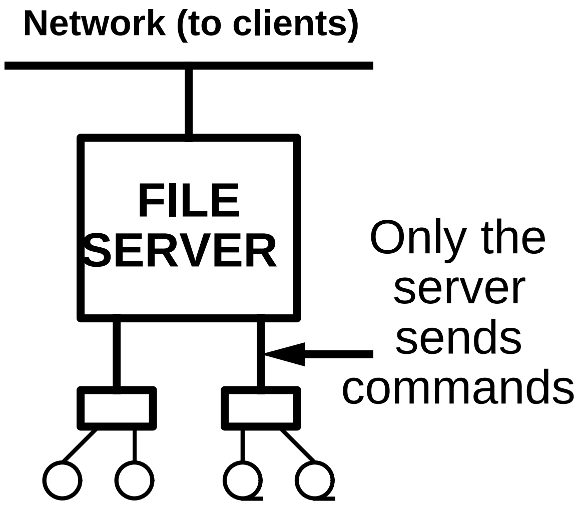
CHEAP

Most data does not go through server

Less latency
Prevents a bottleneck

However, need industry acceptance

Current file servers are reliable



Network (to clients)

FILE SERVER

Only the server sends commands

Check all requests

Sole access to peripherals

- Can log requests
- Can log results
- Cannot be bypassed

Need to achieve similar
control of network attached peripherals

How can file servers control NAPs?

The need is to enable file server ("trusted") initiators to command the network-attached peripherals (NAPs) that they own, while ensuring other initiators cannot exercise such control.

1. Configure NAP to know address of file server. Prevent address forgery.

   Ensure peripherals don't get configuration information from bad sources.

2. Alternatively, provide separate ports and fabrics for control.

Other ("untrusted") initiators must have no access to NAPs except as permitted by trusted initiators.

NAPs must distinguish permitted access and reject other access.

There are some other concerns (like RESET).

All of our Distributed SCSI approaches require such server control.

Problem: Assumptions and Constraints

We do not intend to modify operating system (UNIX) kernels.

We can write or modify drivers and daemons.
We want to minimize such changes, especially for production systems.
We expect heterogeneity.

Today, UNIX kernels don't act as untrusted initiators of NAPs.

Note: this problem may be temporary. If NAPs become accepted, industry may enhance kernels to support them.
For READ/WRITE we are thinking about using "raw device drivers" to bypass kernel and access NAPs.

We assume kernels, daemons, and drivers can be "trusted" to distinguish among applications within a client host.

The problem: Raw device driver access bypasses many kernel checks.

Processor-Peripheral: now four approaches

The first three are described in x3t10/95-122r1. The fourth (below) is new.

READ/WRITE.

An enhancement seems required. See below.

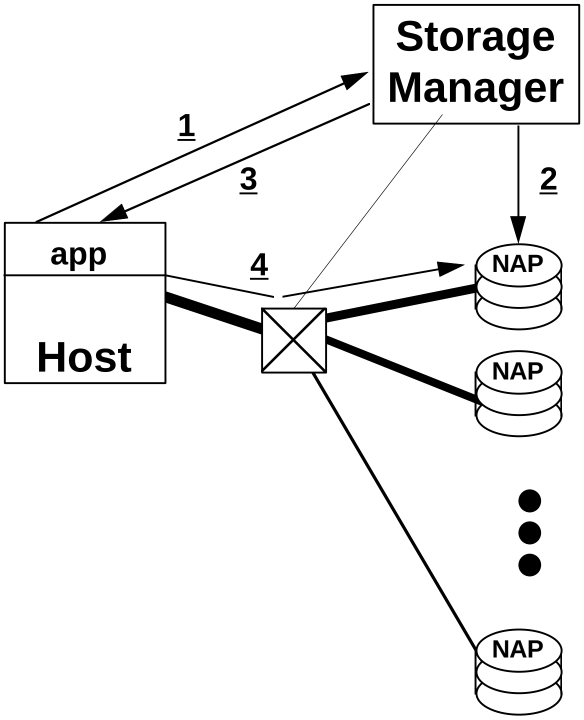
COPY.

"Data exchange."

Peripheral checks "tickets."

New. Similar to READ/WRITE. See below.

How READ/WRITE Works



1. Application makes request to storage manager (or file server).
2. Storage manager makes third-party extent reservation.
3. Storage manager notifies application to proceed and provides data location.
4. Application performs I/O operations.

NAP checks extents and client SCSI address.

Problem: Application process ID is never checked.

Note: These descriptions are quite simplified.

Other applications might gain access

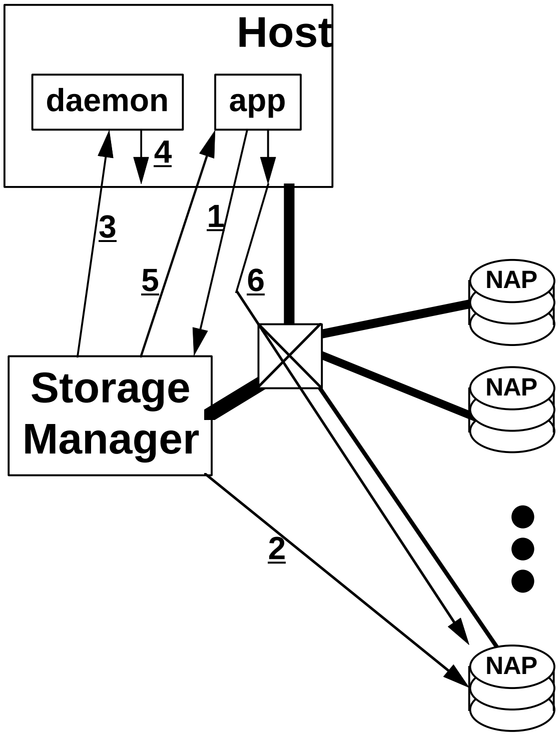
They have to guess data locations and the times when access is possible.

Data locations may be easy to guess. Example: learn location of world-readable files at any time.

Times may be easy to guess. Example: Check for well-known name of the application that can legitimately modify a world-readable file.

Today's commercial file servers do not seem to have this problem.

A fix: Driver checks for correct application



1. Application makes request to storage manager, provides process identifier (PID).

2. Storage manager makes third-party extent reservation.

3. Storage manager sends PID, NAP's SCSI address, and extents to daemon.

4. Daemon tells SCSI driver of authorized request (includes PID etc.).

5. Storage manager notifies application to proceed and provides data location.

6. Application performs I/O operations with associated PID.

SCSI driver checks PID and data location.

NAP checks extents and client SCSI address.

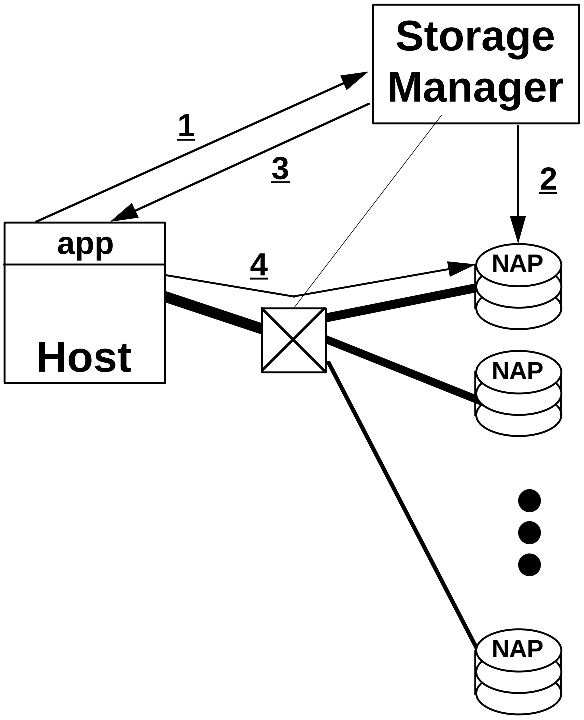
Possibly replace steps 3 and 4: Storage manager uses "AEN" to tell driver of authorization.

Second fix: a fourth approach ("tickets")

1. Application makes request to storage manager.
2. Storage manager creates a ticket and passes the ticket (and information on extents) to the NAP.
3. Storage manager passes ticket and data location to application.
4. Application performs I/O operations, includes ticket with commands.

NAP checks extents and ticket.

The "ticket" is an unguessable identifier, presumably hard to steal.



Summary and Status

Of the three January approaches, Livermore preferred READ/WRITE.

Others have not been examined as carefully.

We are conducting project reviews.

Justify continued funding
Refine requirements and plans

We have not yet decided whether any approaches are acceptable.

We again solicit comments and encourage you to consider working with us.

My strategy remains to use NAPs but stay as close to current file server paradigms as possible until we decide on our strategy.