A Method To Share Data Between Compute Clients With Network Attached Storage Implementation, Operation, and Performance Results

Ken Fallon
Bill Bullers

Impactdata
605 E. Huntington Dr.
Monrovia, CA, 91017
billb@impactdata.com
kenf@impactdata.com
+1-626-359-4491, +1-626-930-9405
Fax: +1-626-930-9478

Abstract: A Distributed Storage Node Architecture (DSNA) where multiple compute clients can concurrently share the same data through a high-bandwidth file system will be installed at the National Energy Research Scientific Computing Center (NERSC) and at the NASA Ames Research Center. A storage protocol defined by the DSNA, includes methods that produce applied, distributed, concurrent data management capabilities designed to allow multiple SGI and CRAY clients to reliably share data through network attached storage. This paper discusses the implementation, operation, and measured performance of the DSNA and the applied protocol. Implementation issues and initial test results are discussed. DSNA is an available, open standard.

A Network Peripheral Adapter (NPA) is embodied within the Distributed Storage Node Architecture of a network-centric computing and data storage enterprise. The NPA is an intelligent controller and optimized file server that empowers network-attached processors with DSNA protocol the means to unlock the potential of distributed data access while enjoying the benefits of centralized management and control. Optimized for high-performance and peer-level storage, the NPA is fabric-independent and capable of linking virtually any workstation, high-performance computer and/or network with any type of storage device.

By connecting high-performance disk arrays and tape drives to high-speed networks, the NPA creates network storage and allows computers to access and share data. First integrations target data sharing between multiple SGI, CRAY, and NT systems but will be extended to include SUN, HP, and AIX clients.

Introduction

The cutting edge of business, science, and industry, driving for computing, visualization, communications and information content are constantly creating challenges in data and information storage and retrieval processes. Context properties and intelligent formats for management agents are being applied to data. The ever increasing demand for managing larger quantities of data, and the growing requirement for rapid, distributed, global access is forcing the exploration of new approaches to handle and process data to be stored, searched, mined, and made available. This demand is sponsoring the evolution of data storage products that insulate users from storage location, media, and protective services, and is bringing about the concept of enterprise wide ***network computing***. Retrieval, archive, backup, hierarchical storage management applications, and distributed objects are evolving but, intelligence within the data storage devices and media has been slow to develop. As a result, there is an overwhelming demand for more sophisticated storage servers and intelligent storage media controllers.

This paper describes measured performance results using an architecture that provides intelligence at the storage device level and enables network attached processors to access and share data.

Distributed Storage Node Architecture

Distributed Storage Node Architecture (DSNA) was created to facilitate data management, data storage, data sharing, data archive, distributed objects, backup and retrieval. DSNA was developed with the flexibility to expand with the needs of high performance computing and high speed network users. Network file systems, like NFS, operate by copying data through multiple memory levels from the client to the file system, to cache, to user space, using small datagram block sizes. As a result they are inherently unable to provide high-performance network data-flows. These systems typically apply stateless protocols, with unmanaged concurrency, that requires the full intelligence including file serialization to be provided by the client. NFS clients must be 'smart' because NFS servers are 'dumb'. DSNA is designed with an entirely different structure that integrates file system intelligence into the server and operates with peer-level data flows to deliver large blocks and avoid multiple memory transfers. DSNA is a solution to the limitations of sending large files at high speeds. Applying DSNA protocol, a set of drivers is implemented with a Common Peripheral Interface (CPI), to utilize high speed networks effectively. There are no application level client drivers required.

DSNA is structured to be a complete storage system solution that provides several integrated benefits:

- A Coherent and Cohesive Network Storage Environment
- Intelligent Storage and Data Mining
- The Management of Data Stores
- Handling Increased Network Bandwidth and Response Times
- Storage Servers and Controllers that Provide for Rapid Growth and Change
- Support for Evolving Distributed Object Storage

The implementation of Distributed Storage Node architecture is based on standard components that support distinct network and storage device interfaces. These components are used to achieve fabric-independent integrations for high-performance and traditional networks. DSNA can accommodate interface components for connection to HIPPI, Fibre Channel, ATM OC-3/12, SCSI, Gigabit Ethernet, and even 10-BaseT and 100-BaseT. These components are integrated in a Network Peripheral Adapter (NPA) and are connected together through a PCI data bus. High-rate data transfer is achieved by bus-mastering the interface components. Taking advantage of memory buffers built into the interfaces, the NPA processor reacts to data transfer commands by initiating peer-level Direct Memory Access (DMA) data-flow between the components and eliminates multiple processor/cache read interactions with the data. By circumventing the processor, data-flows up to 100 MB/s can be achieved across the 32 bit, 132 MB/s PCI bus.

Figure 1 shows the internal structure of the NPA.

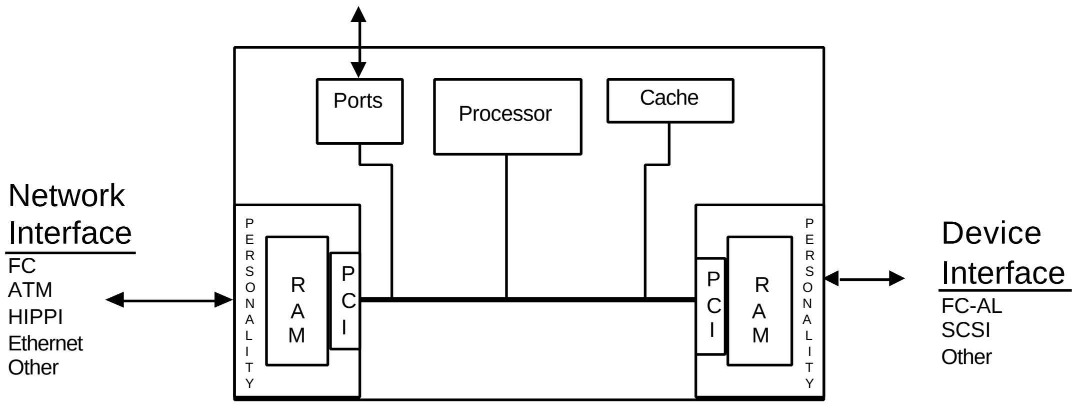

Figure 1

The PCI bus-mastering technique achieves high-rate data-flow while significantly freeing the burden on the processor and operating system, and allows a low cost NPA implementation with ‘Wintel’ technology. The use of standard component modules achieves significant flexibility for the DSNA to provide data storage in an open system context. The physical level operation and structure embodied in the NPA is a significant building block, but is only one of several major elements that enable DSNA to provide high-performance distributed data storage.

Elements of DSNA

DSNA creates an environment for multiple network attached storage nodes each with a Network Peripheral Adapters (NPA) and attached storage devices as shown in the next figure.

DSNA supports the notion of distributed file management across NPAs and provides the capability for true managed file-extent concurrency across clients. Files can be created on one client with extents of the file opened, concurrently modified, or locked for pristine manipulation by other clients, even using different operating systems. Each storage node can be setup with DSNA applications that provide Hierarchical Storage Management, data archive, backup, and disaster recovery processes. DSNA provides integrated storage management as part of a ‘Distributed Management’ methodology in which any processor on the network can assert management functions. Each processor maintains its own management data, such as operational status, configuration records, file and media level storage statistics, alarms, accounting and billing, and software update facilities. Each processor can query every other processor and display this management information for any storage node. Any NPA can also be setup to act as a security server to manage the security resources across a DSNA network. Operator access requires ID and password authentication with data access restrictions by ID. C2 security and POSIX compliance are provided in all processors.

A DSNA storage node implementation can be as simple as a single NPA and storage device that provides intelligent network connected shared data storage. It can be a diverse hierarchical arrangement of disks, removable media drives, and libraries.

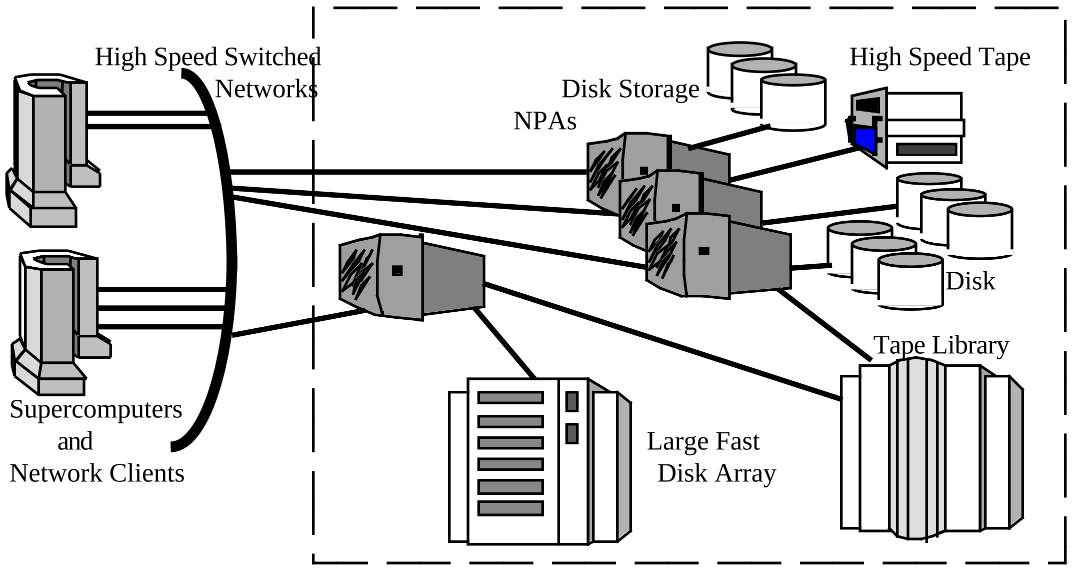

Figure 2: Typical DSNA Storage Nodes

Personality Modules defined to support standard application familiar protocols such as NFS and FTP are embodied within DSNA. Personality Modules include distributed object services with support for Object Request Brokers as defined through CORBA and ActiveX to promote open access to intelligent storage.

The Common Peripheral Interface is middleware that enables UNIX (and Windows NT) clients to use and share network storage as if it were locally attached. Used with any application that reads and/or writes to locally attached devices, the CPI converts Input/Output (I/O) operations to DSNA network commands with DSNA protocol. Users can take advantage of either a ‘transparent’ access to DSNA storage nodes or an Application Programmers Interface (API), through which they can apply a rich set of DSNA features including metadata and local file management. The DSNA protocol has as its foundation the proven IPI-3 storage protocol over which several extensions are specifically designed to accomplish high-performance network shared data storage. Device drivers are provided for targeted systems that transform standard file I/O requests into DSNA commands and messages. These CPI drivers provide a local file system connection through defined mount points to DSNA storage nodes with disk and tape media. Host application programs only require modifications if the rich DSNA feature set is to be fully exploited. The DSNA protocol and the CPI integrate a metadata facility that collects useful information about the files, the storage devices, and the media. The DSNA Metadata is a vital component of the Storage Node File System (SNFS). The information is collected in real-time and is available to DSNA utilities that use the information to locate and manage directories and files, support file access at the block level, maintain file security and locking, and provide operational statistics. Other utility agents link this Metadata to the HSM and Archive Manager to place and locate files.

Common Peripheral Interface Operation

CPI drivers support two modes of operation, Transparent Mode, and API mode. Transparent Mode allows existing applications to apply DSNA without modifications and achieve high-performance results. Transparent Mode does not give user applications the full range of DSNA features. API mode makes the full feature set available to the user. Applications that use Transparent Mode issue OPEN, CLOSE, IOCTL, READ and WRITE calls to a mount point. The mount point configuration/access table insures that the call is routed to the SNFS interface where it is converted to CPI commands. Two classes of storage device are supported, tape and disk. The API mode is a programmatic interface to the CPI Driver that uses calls to OPEN, CLOSE, READ, WRITE, IOCTL and CPICMD. CPICMD makes Metadata and other DSNA capabilities available to the application. These capabilities include access to file metadata to exploit a broad set of execution, and administrative content:

- Formatting and Configuration
- File Attribute Reports and Control
- Metadata Reports and Controls
- Operation and Execution Status
- Abort Execution Methods
- File Mark and Position Control
- Diagnostics and Error Logs
- Session Control
- Data Delivery Times

Figure 3 shows the difference between how CPI Transparent Mode and API Mode operate within User Space (Application level) and Kernel space (CPI level).

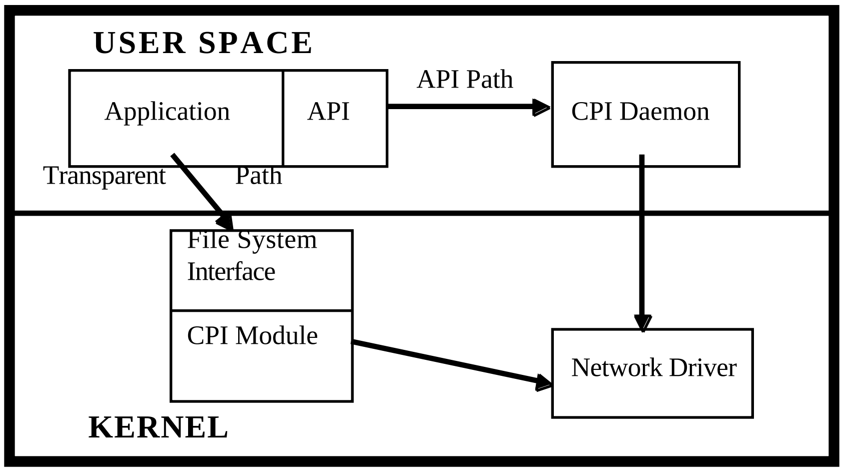

Figure 3: Common Peripheral Interface Operation

API Mode

Details of the API Mode operation are presented in the next diagram. The CPI daemon process and the user application that calls the API are independent UNIX processes. The API code runs as part of the application and communicates with the CPI daemon through UNIX inter-process communication facilities. The CPI daemon can service multiple applications allowing several simultaneous accesses to the same NPA storage device. The API is a socket connection interface through which the CPI daemon and application communicate. The CPI daemon spawns a separate network 'child' process for each socket connection and returns the results to the application. The socket connection is used to communicate the API commands and responses. The transfer of data, defined in the API call, takes place through shared memory. The socket connection alerts the CPI daemon to associate an area of shared memory with the application that is used for the data transfer.

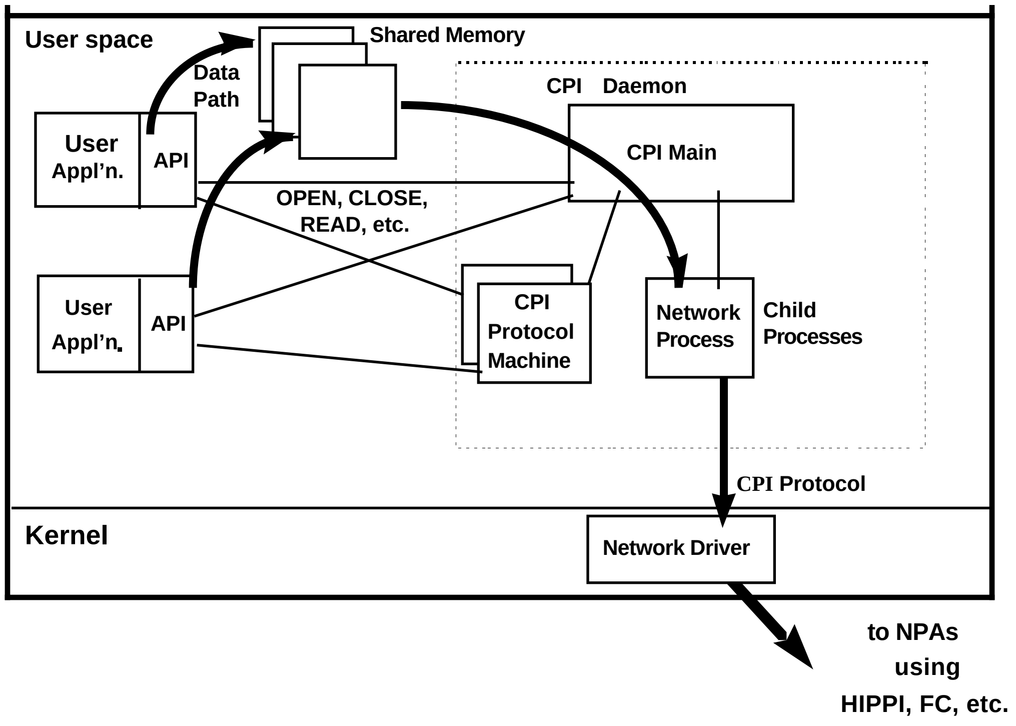

Figure 4

Transparent Mode

The file system appears to be local to the UNIX client in the CPI Transparent Mode. UNIX commands operate normally even though files are maintained on the NPA and available through the network. The CPI Driver uses the mount point identifier to route the file request to the Storage Node File System (SNFS). Through the SNFS, requests are converted to CPI/DSNA protocol and issued to the network driver which then delivers them to the NPA. The NPA processes the request and delivers the data to the destination device through the bus-mastered interface. The SNFS operates as a Virtual File System within UNIX and supports all the attributes of a local file system. The Local UNIX file system processes client application requests and routes the requests through the UNIX Vnode Interface to SNFS as shown in the diagram that follows. The SNFS is architected into UNIX in the same way other file systems like NFS integrate.

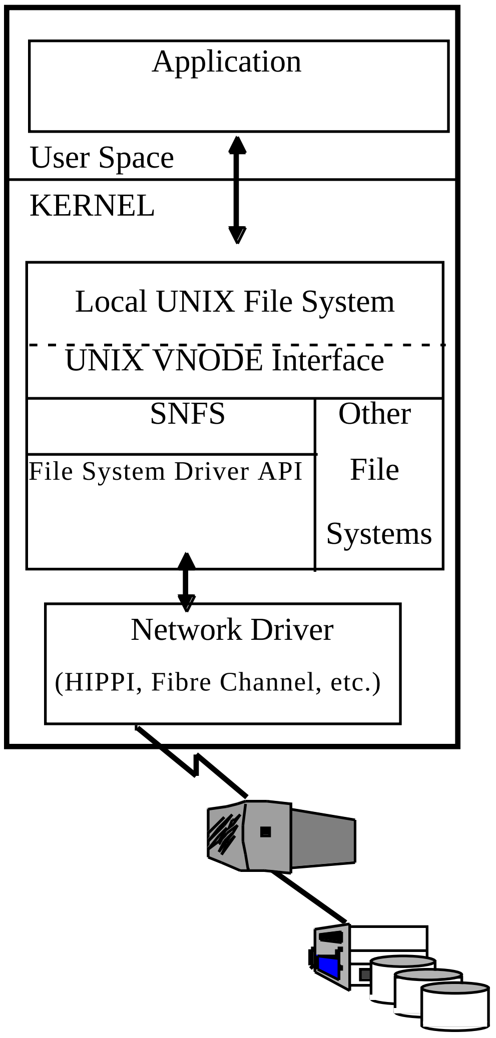

Figure 5: File System Interface

The SNFS supports features that are common to the Local File System. A configuration/access file is used to enforce system security through encrypted NPA address IDs, license keys, and passwords. Metadata maintained at the NPA, controls file access modes including locking, and maintains user ownership, group ownership, authorization, and privilege information that manages user files. Files that are shared by multiple clients can be locked at different levels including NO locking, READ/WRITE locking, and WRITE locking. The default is lock on WRITE.

CPI supports multiple concurrent accesses from different network attached clients but it is envisioned to support multiple accesses from a single client. The initial implementation of the architecture is limited to a single shared file handle per client but future releases will enable separate file handles to be created for each of multiple file accesses made by a client. This capability will permit concurrent parallel processing applications to be performed on distinct extents of a file by a single client.

Other File Systems

DSNA is structured to operate within several different file systems based on the mount point that is selected. The simplest format for operating with multiple file systems like NFS, and SGI's BDS, is to define and allocate specific disk partitions and tape volumes to each system. Other architectures have been devised that reformat files and transfer their data from one file system to another. DSNA implements a completely open solution that eliminates the need for separate partitions and allows data to be concurrently shared across processes and across file systems. The Windows NT operating system provides an NT Redirector that allows Personality Module drivers to be written and installed that 'redirect' file accesses for selected mount points from the 'other file system' to CPI. A client application that chooses to operate through NFS, for example, is able to do that by defining specific mount points for NFS files to be managed. The NT Redirector diverts the file access control from NFS through CPI to the NPA and allows the file to be opened at the NPA as an SNFS file and at the client application as an NFS file. Clients attached to networks that support TCP/IP such as Gigabit Ethernet and future releases of the Essential Communications HIPPI Card can apply the network performance benefits of CPI to standard current applications.

DSNA Integration and Benchmarks

DSNA was first integrated as a high-performance network storage node in a High Performance Storage System (HPSS) at The Caltech Center for Advanced Computing Research by connecting a 72 GB FibreRAID through an NPA to the HIPPI network. The HPSS server provides native support for third-party IPI-3 disk which allows direct data transfers to be setup and executed from the HPSS server to create a data flow path directly between the disk and the client application. This avoids the performance lag caused by staging the data through multiple memory read operations. Several performance measurements and benchmarks have been made using files of 64MB and smaller. Performance using large files (to 2GB) will be evaluated in early 1998 to compare transfer rates between local disk and network attached disk. A similar configuration will be installed at the Lawrence Livermore National Laboratory where additional benchmark testing is planned. These first integrations do not explore the full capabilities of the DSNA because they are limited to direct IPI-3 connection and do not require the data sharing capability of the Storage Node File System.

Performance

The use of standard, commercially available network and device interface components to structure a cost efficient design has required close cooperation between Impactdata and partner suppliers to realize the performance objectives of the DSNA. A Pentium Pro processor card in a passive backplane system with a standard 32 bit, 33 MHz PCI bus is used in all the testing. HIPPI network connectivity is accomplished with the Essential Communications PCI adapter using an Impactdata developed peer-level driver that includes a DMA stream engine. For disk array attachment with SCSI protocol on Fibre Channel, Emulex and Systran Adapters have been tested. Both use Impactdata developed drivers optimized for bus mastered peer-level transfers. Current test results show 35 Mbytes per second data transfers with small transfer sizes of 4MB. This rate was achieved by adding a 16MB transfer memory buffer between the network and device cards. Impactdata expects to achieve 50 Mbytes per second by expanding the memory buffer size to 128MB and increasing the transfer size to 32MB. Additional driver optimization will be possible with firmware changes in work at Essential and the Fibre Channel card vendors. These changes coupled with data transfers of 128 Mbytes or greater are expected to achieve the objective 65 Mbyte per second transfer rate.

Conclusions and Future Work

The cost efficient implementation achieved through the network peripheral adapter coupled with high data transfer and managed file sharing make DSNA a very practical architecture to embrace. The first applications to use true file sharing properties achieved through CPI are with SGI Origin 2000 and CRAY client systems in an integration at the National Energy Research Scientific Computing Center (NERSC). The NERSC integration is the first opportunity to move DSNA file sharing out of the Impactdata laboratory into a computer center test configuration. An installation is also scheduled at NASA Ames to operate with CPI in the Transparent Mode and enable multiple clients to share files. Additional validation tests to collect file transfer statistics and performance benchmarks are planned at Ames, along with stress tests of concurrency, file locking, and hierarchical storage management. Development is in process for similar CPI integrations with HP Convex, and SUN Systems.