Configurable Data Manipulation in an Attached Multiprocessor

Marc F. Pucci Bellcore

ABSTRACT: The ION Data Engine is a multiprocessor tasking system that provides data manipulation services for collections of workstations or other conventional computers. It is a back-end system, connecting to a workstation via the Small Computer Systems Interface (SCSI) disk interface. ION appears to the workstation as a large, high speed disk device, but with user extensible characteristics. By mapping an application's functionality into simple disk read and write accesses, ION achieves a high degree of application portability, while providing enhanced performance via dedicated processors closely positioned to I/O devices and a streamlined tasking system for device control.

The programming model for ION supports the notion of separation of control function from data transmission. Typically, a small list of data manipulation directives is transmitted from the workstation to the ION node, where data filtering or other forms of processing occur. Only results, as opposed to all data, need be returned to the workstation. In the extreme case, the ION system can acquire all input data and generate all output data, without any processing occurring in the workstation. An example application uses a simple set of directives to capture and digitize high quality stereo audio, mix it to monaural, rate adjust the digitized samples to ISDN rates, convert

from binary to mulaw encoding, and transmit the result to a workstation.

ION is being used as an experimental platform for voice mail services in a userprogrammable telephone switch prototype, and as a tool for measuring the I/O performance of computer-disk interfaces. Applications under development include an automated camera positioning system and an object repository.

1. Introduction

The workstations that exploit the rapidly advancing state-of-the-art in processor technology can often be a bane to developers of applications that utilize dedicated special purpose hardware or that impose strict access requirements on conventional hardware. Such evolving systems can suffer from:

- Constantly porting hardware dependent components to new hardware.
- Being locked into a particular vendor to avoid major hardware disruptions.
- Forcing the use of high-end stations because entry-level stations are not easily expandable.
- Constantly upgrading local workstation based device drivers to coexist with operating system releases.
- Relying upon an operating system that is not appropriate for the system's functionality.
- Insufficient workstation capacity to support the hardware requirements of the application.

Applications tied to obsolete processor technology will soon suffer from comparative performance problems as newer workstation technology passes it by. However, interfacing new workstations to an existing hardware base is not simple. Initial workstation offerings often possess

meager expansion characteristics, typically just a disk and network connection, so achieving even the electrical connection can be difficult.

Ideally, utilizing a new workstation should entail only simple recompilation of the application code; however, machine dependencies that result from the use of special purpose hardware complicate a code port. Workstation hardware may not be portable to different manufacturer's stations or even across a line of workstations from the same vendor. This can lead to the loss of a significant hardware investment as working components must be redesigned. Supporting multiple versions of hardware in order to preserve customer satisfaction with older configurations can also be expensive. Even hardware common to multiple stations, which is currently possible since many stations now offer VME bus interfaces, may still require device driver changes and must also track operating system variations from release to release.

An additional problem of using special purpose hardware on a conventional workstation is that the internal structure of the host operating system may not be conducive to the requirements of the hardware. It may be preferable to model an application into subtasks, each with its own flow of control; however, the relatively expensive context switch time for a general purpose operating system may make such an implementation infeasible for performance reasons. Also, the data rates generated by some hardware may have a detrimental effect on other functions in the workstation. In general, it is best to place compute power as close as possible to the source of data, passing only results or preprocessed information on to higher levels in the system. In this manner, devices requiring rapid response need not interfere with time-sharing operations.

ION addresses these problems by partitioning an application into hardware dependent and independent components, and providing a vendor independent interface between the two. The hardware independent components reside in the workstation, and are therefore easily ported to new architectures. The hardware dependent components are situated within a separate backplane-based environment, which is portable in its entirety across workstation changes. The low level connection between these components is the Small Computer Systems Interface (SCSI) disk interface, ANSI X3.131. Since each workstation accesses ION using its local disk system, which is a stable, well-defined interface, there is no need to change vendor supplied host sys-

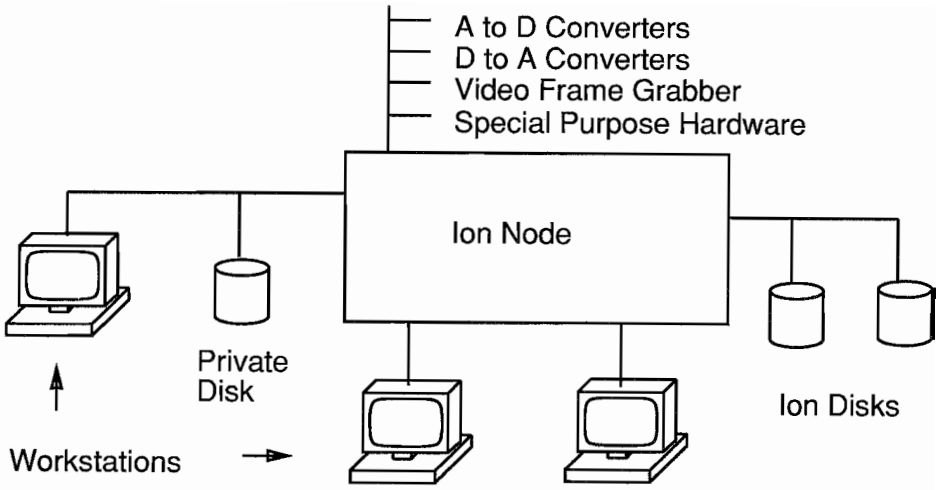

Figure 1. An ION system. Multiple workstations connect to an ION node, which contains single board computers and other peripheral interfaces and devices. Each workstation views its ION connection as though it were a large conventional disk drive.

tem software. Current SCSI performance capabilities also provide a respectable (5 megabyte per second) data access rate.

ION configurations are expandable and sharable as needs dictate. Additional single board computers (SBC's) in a backplane can connect multiple workstations to the same set of hardware resources, or provide extra CPU cycles for I/O devices or applications that require it. Further expansion is possible by using bus repeaters and local area networks to interconnect multiple ION nodes together. The basic structure of an ION system is shown in Figure 1.

2. *The ION Interface*

A workstation sees ION as though it were physically a local disk drive (an ION drive) with a data capacity of 2 terabytes (the SCSI limit). Software running within the ION system mimics the behavior of a conventional device, providing the workstation with a peripheral that it knows how to deal with. The "data" contained in this pseudo-disk

device can be random read/write data, traditional file system data, or more complex objects for a variety of applications managed by tasks running within the ION system. The latter is implemented by defining application specific functions, called *actions,* that are enabled by reading or writing specific disk block addresses within the ION drive.

For example, ION supports an analog to digital (A-to-D) conversion application that provides voice messaging service for a prototype telephone switch. The bulk of the application resides in a conventional workstation, while the peripheral devices are located within ION. The application's interface to the A-to-D converters is implemented as an action defined on a set of 5 disk block addresses, each corresponding to 1 of the 5 analog channels. The controlling program within the workstation merely reads from one of these designated disk block addresses to obtain the converted data (**lseek()** followed by **read()** in the Unix domain). By defining such interactions in terms of standard disk read and write accesses, the application remains portable across workstation changes, operating system releases, and to a large degree, complete operating system changes (e.g., Unix to VMS), while preserving any existing special purpose hardware investments.

A further advantage of the disk-like interface of ION is its robustness in the face of application failure. Since ION mimics a local disk drive, the worst case scenario for failure merely results in the apparent symptom that the ION drive has gone into an *off-line* condition equivalent to a real drive losing power or spinning down. This should not have any long lasting effect on the workstation and is remedied by rebooting the ION system.

3. System Architecture

The hardware configuration of an ION node is shown in Figure 2. The current ION configuration uses high speed Motorola 68030 microprocessor based single board computers (SBC's). A port to an Intel 960 based product is underway, although the current system only deals with homogeneous processors. These processors offer sufficient power for the current set of ION I/O devices and will be upgraded to faster processors when more demanding peripherals are in use.

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.