RAID-II: A High-Bandwidth Network File Server

Ann L. Drapeau Ken W. Shirriff John H. Hartman Ethan L. Miller
Srinivasan Seshan Randy H. Katz Ken Lutz David A. Patterson[1]

Edward K. Lee[2] Peter M. Chen[3] Garth A. Gibson[4]

Abstract

In 1989, the RAID (Redundant Arrays of Inexpensive Disks) group at U. C. Berkeley built a prototype disk array called RAID-I. The bandwidth delivered to clients by RAID-I was severely limited by the memory system bandwidth of the disk array's host workstation. We designed our second prototype, RAID-II, to deliver more of the disk array bandwidth to file server clients. A custom-built crossbar memory system called the XBUS board connects the disks directly to the high-speed network, allowing data for large requests to bypass the server workstation. RAID-II runs Log-Structured File System (LFS) software to optimize performance for bandwidth-intensive applications.

The RAID-II hardware with a single XBUS controller board delivers 20 megabytes/second for large, random read operations and up to 31 megabytes/second for sequential read operations. A preliminary implementation of LFS on RAID-II delivers 21 megabytes/second on large read requests and 15 megabytes/second on large write operations.

1 Introduction

It is essential for future file servers to provide high-bandwidth I/O because of a trend toward bandwidth-intensive applications like multi-media, CAD, object-oriented databases and scientific visualization. Even in well-established application areas such as scientific computing, the size of data sets is growing rapidly due to reductions in the cost of secondary storage and the introduction of faster supercomputers. These developments require faster I/O systems to transfer the increasing volume of data.

High performance file servers will increasingly incorporate disk arrays to provide greater disk bandwidth. RAIDs, or Redundant Arrays of Inexpensive Disks [13], [5], use a collection of relatively small, inexpensive disks to achieve high performance and high reliability in a secondary storage system. RAIDs provide greater disk bandwidth to a file by striping or interleaving the data from a single file across a group of disk drives, allowing multiple disk transfers to occur in parallel. RAIDs ensure reliability by calculating error correcting codes across a group of disks; this redundancy information can be used to reconstruct the data on disks that fail. In this paper, we examine the efficient delivery of bandwidth from a file server that includes a disk array.

In 1989, the RAID group at U.C. Berkeley built an initial RAID prototype, called RAID-I [2]. The prototype was constructed using a Sun 4/280 workstation with 128 megabytes of memory, four dual-string SCSI controllers, 28 5.25-inch SCSI disks and specialized disk striping software. We were interested in the performance of a disk array constructed of commercially-available components on two workloads: small, random operations typical of file servers in a workstation environment and large transfers typical of bandwidth-intensive applications. We anticipated that there would be both hardware and software bottlenecks that would restrict the bandwidth achievable by RAID-I. The development of RAID-I was motivated by our desire to gain experience with disk array software, disk controllers and the SCSI protocol.

Experiments with RAID-I show that it performs well when processing small, random I/O's, achieving approximately 275 four-kilobyte random I/Os per second. However, RAID-I proved woefully inadequate at providing high-bandwidth I/O, sustaining at best 2.3 megabytes/second to a user-level application on RAID-I. By comparison, a single disk on RAID-I can sustain 1.3 megabytes/second. The bandwidth of nearly 26 of the 28 disks in the array is effectively wasted because it cannot be delivered to clients.

There are several reasons why RAID-I is ill-suited for high-bandwidth I/O. The most serious is the memory contention experienced on the Sun 4/280 workstation during I/O operations. The copy operations that move data between kernel DMA buffers and buffers in user space saturate the memory system when I/O bandwidth reaches 2.3 megabytes/second. Second, because all I/O on the Sun 4/280 goes through the CPU's virtually addressed cache, data transfers experience interference from cache flushes. Finally, disregarding the workstation's memory bandwidth limitation, high-bandwidth performance is also restricted by the low backplane bandwidth of the Sun 4/280's VME system bus, which becomes saturated at 9 megabytes/second.

The problems RAID-I experienced are typical of many workstations that are designed to exploit large, fast caches for good processor performance, but fail to support adequate primary memory or I/O bandwidth [8], [12]. In such

[1]Computer Science Division, University of California, Berkeley, CA 94720
[2]DEC Systems Research Center, Palo Alto, CA 94301-1044
[3]Computer Science and Engineering Division, University of Michigan, Ann Arbor, MI 48109-2122
[4]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA 15213-3891

1063-6897/94 $03.00 © 1994 IEEE

workstations, the memory system is designed so that only the CPU has a fast, high-bandwidth path to memory. For busses or backplanes farther away from the CPU, available memory bandwidth drops quickly. Thus, file servers incorporating such workstations perform poorly on bandwidth-intensive applications.

The design of hardware and software for our second prototype, RAID-II [1], was motivated by a desire to deliver more of the disk array's bandwidth to the file server's clients. Toward this end, the RAID-II hardware contains two data paths: a high-bandwidth path that handles large transfers efficiently using a custom-built crossbar interconnect between the disk system and the high-bandwidth network, and a low-bandwidth path used for control operations and small data transfers. The software for RAID-II was also designed to deliver disk array bandwidth. RAID-II runs LFS, the Log-Structured File System [14], developed by the Sprite operating systems group at Berkeley. LFS treats the disk array as a log, combining small accesses together and writing large blocks of data sequentially to avoid inefficient small write operations. A single XBUS board with 24 attached disks and no file system delivers up to 20 megabytes/second for random read operations and 31 megabytes/second for sequential read operations. With LFS software, the board delivers 21 megabytes/second on sequential read operations and 15 megabytes/second on sequential write operations. Not all of this data is currently delivered to clients because of client and network limitations described in Section 3.4.

This paper describes the design, implementation and performance of the RAID-II prototype. Section 2 describes the RAID-II hardware, including the design choices made to deliver bandwidth, architecture and implementation details, and hardware performance measurements. Section 3 discusses the implementation and performance of LFS running on RAID-II. Section 4 compares other high performance I/O systems to RAID-II, and Section 5 discusses future directions. Finally, we summarize the contributions of the RAID-II prototype.

2 RAID-II Hardware

Figure 1 illustrates the RAID-II storage architecture. The RAID-II storage server prototype spans three racks, with the two outer racks containing disk drives and the center rack containing custom-designed crossbar controller boards and commercial disk controller boards. The center rack also contains a workstation, called the host, which is shown in the figure as logically separate. Each XBUS controller board has a HIPPI connection to a high-speed Ultra Network Technologies ring network that may also connect supercomputers and client workstations. The host workstation has an Ethernet interface that allows transfers between the disk array and clients connected to the Ethernet network.

In this section, we discuss the prototype hardware. First, we describe the design decisions that were made to deliver disk array bandwidth. Next, we describe the architecture and implementation, followed by microbenchmark measurements of hardware performance.

2.1 Delivering Disk Array Bandwidth

2.1.1 High and Low Bandwidth Data Paths

Disk array bandwidth on our first prototype was limited by low memory system bandwidth on the host workstation. RAID-II was designed to avoid similar performance limitations. It uses a custom-built memory system called the XBUS controller to create a high-bandwidth data path. This data path allows RAID-II to transfer data directly between the disk array and the high-speed, 100 megabytes/second HIPPI network without sending data through the host workstation memory, as occurs in traditional disk array storage servers like RAID-I. Rather, data are temporarily stored in memory on the XBUS board and transferred using a high-bandwidth crossbar interconnect between disks, the HIPPI network, the parity computation engine and memory.

There is also a low-bandwidth data path in RAID-II that is similar to the data path in RAID-I. This path transfers data across the workstation's backplane between the XBUS board and host memory. The data sent along this path include file metadata (data associated with a file other than its contents) and small data transfers. Metadata are needed by file system software for file management, name translation and for maintaining consistency between file caches on the host workstation and the XBUS board. The low-bandwidth data path is also used for servicing data requests from clients on the 10 megabits/second Ethernet network attached to the host workstation.

We refer to the two access modes on RAID-II as the *high-bandwidth mode* for accesses that use the high-performance data path and *standard mode* for requests serviced by the low-bandwidth data path. Any client request can be serviced using either access mode, but we maximize utilization and performance of the high-bandwidth data path if smaller requests use the Ethernet network and larger requests use the HIPPI network.

2.1.2 Scaling to Provide Greater Bandwidth

The bandwidth of the RAID-II storage server can be scaled by adding XBUS controller boards to a host workstation. To some extent, adding XBUS boards is like adding disks to a conventional file server. An important distinction is that adding an XBUS board to RAID-II increases the I/O bandwidth available to the *network*, whereas adding a disk to a conventional file server only increases the I/O bandwidth available to that particular file server. The latter is less effective, since the file server's memory system and backplane will soon saturate if it is a typical workstation.

Eventually, adding XBUS controllers to a host workstation will saturate the host's CPU, since the host manages all disk and network transfers. However, since the high-bandwidth data path of RAID-II is independent of the host workstation, we can use a more powerful host to continue scaling storage server bandwidth.

2.2 Architecture and Implementation of RAID-II

Figure 2 illustrates the architecture of the RAID-II file server. The file server's backplane consists of two high-bandwidth (HIPPI) data busses and a low-latency (VME) control bus. The backplane connects the high-bandwidth network interfaces, several XBUS controllers and a host workstation that controls the operation of the XBUS controller boards. Each XBUS board contains interfaces to the

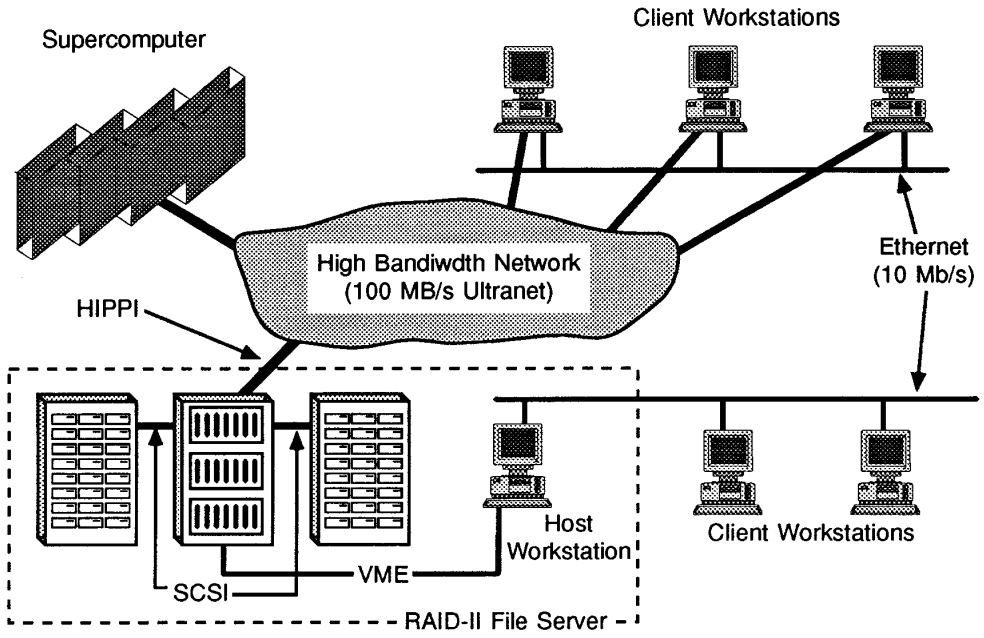

Figure 1: The RAID-II File Server and its Clients. RAID-II, shown by dotted lines, is composed of three racks. The host workstation is connected to one or more XBUS controller boards (contained in the center rack of RAID-II) over the VME backplane. Each XBUS controller has a HIPPI network connection that connects to the Ultranet high-speed ring network; the Ultranet also provides connections to client workstations and supercomputers. The XBUS boards also have connections to SCSI disk controller boards. In addition, the host workstation has an Ethernet network interface that connects it to client workstations.

HIPPI backplane and VME control bus, memory, a parity computation engine and four VME interfaces that connect to disk controller boards.

Figure 3 illustrates the physical packaging of the RAID-II file server, which spans three racks. To minimize the design effort, we used commercially available components whenever possible. Thinking Machines Corporation (TMC) provided a board set for the HIPPI interface to the Ultranet ring network; Interphase Corporation provided VME-based, dual-SCSI, Cougar disk controllers; Sun Microsystems provided the Sun 4/280 file server; and IBM donated disk drives and DRAM. The center rack is composed of three chassis. On top is a VME chassis containing eight Interphase Cougar disk controllers. The center chassis was provided by TMC and contains the HIPPI interfaces and our custom XBUS controller boards. The bottom VME chassis contains the Sun4/280 host workstation. Two outer racks each contain 72 3.5-inch, 320 megabyte IBM SCSI disks and their power supplies. Each of these racks contains eight shelves of nine disks. RAID-II has a total capacity of 46 gigabytes, although the performance results in the next section are for a single XBUS board controlling 24 disks. We will achieve full array connectivity by adding XBUS boards and increasing the number of disks per SCSI string.

Figure 4 is a block diagram of the XBUS disk array controller board, which implements a 4x8, 32-bit wide crossbar interconnect called the XBUS. The crossbar connects four memory modules to eight system components called *ports*. The XBUS ports include two interfaces to HIPPI network boards, four VME interfaces to Interphase Cougar disk controller boards, a parity computation engine, and a VME interface to the host workstation. Each XBUS transfer involves one of the four memory modules and one of the eight XBUS ports. Each port was intended to support 40 megabytes/second of data transfer for a total of 160 megabytes/second of sustainable XBUS bandwidth.

The XBUS is a synchronous, multiplexed (address/data), crossbar-based interconnect that uses a centralized, priority-based arbitration scheme. Each of the eight 32-bit XBUS ports operates at a cycle time of 80 nanoseconds. The memory is interleaved in sixteen-word blocks. The XBUS supports only two types of bus transactions: reads and writes. Our implementation of the crossbar interconnect was fairly expensive, using 192 16-bit transceivers. Using surface mount packaging, the implementation required 120 square inches or approximately 20% of the XBUS controller's board area. An advantage of the implementation is that the 32-bit XBUS ports are relatively inexpensive.

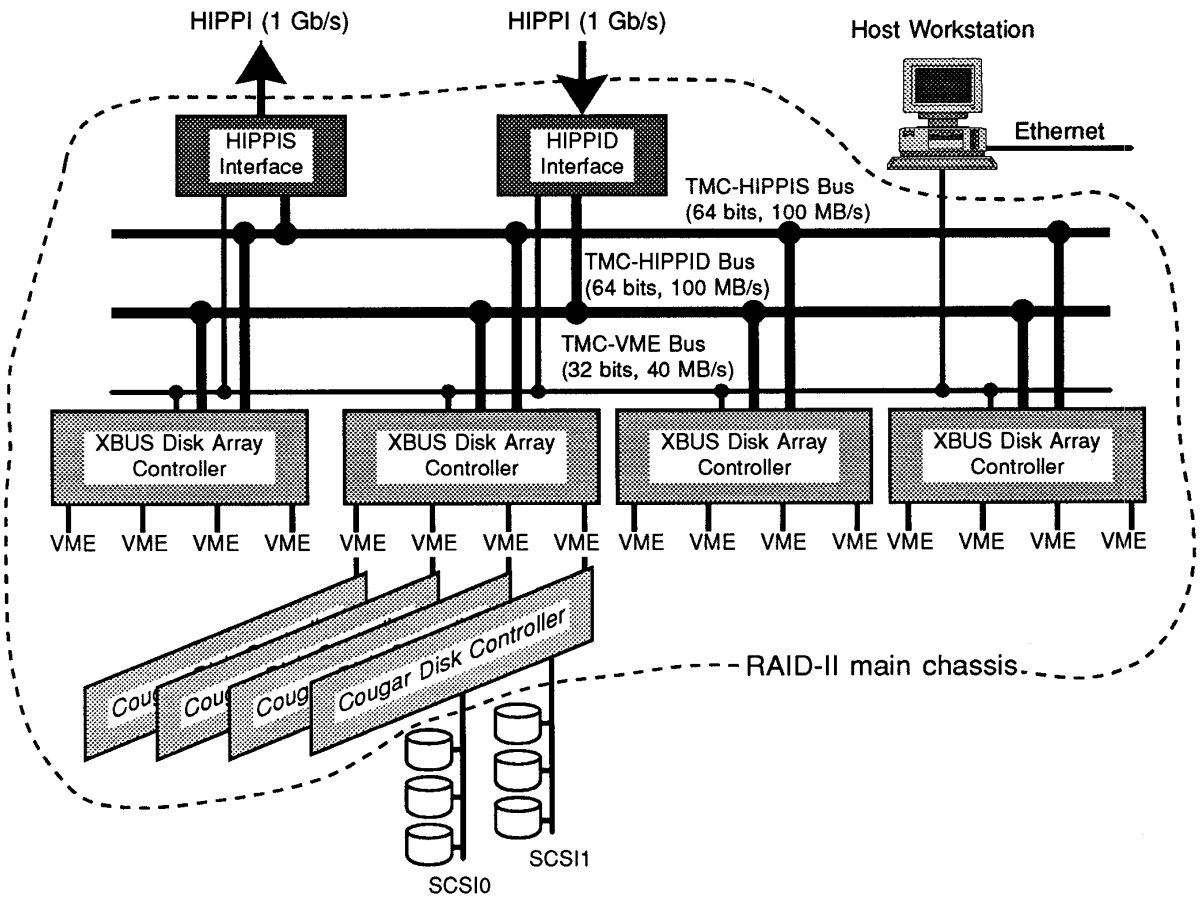

Figure 2: Architecture of RAID-II File Server. The host workstation may have several XBUS controller boards attached to its VME backplane. Each XBUS controller board contains interfaces to HIPPI network source and destination boards, buffer memory, a high-bandwidth crossbar, a parity engine, and interfaces to four SCSI disk controller boards.

Two XBUS ports implement an interface between the XBUS and the TMC HIPPI boards. Each port is unidirectional, designed to sustain 40 megabytes/second of data transfer and bursts of 100 megabytes/second into 32 kilobyte FIFO interfaces.

Four of the XBUS ports are used to connect the XBUS board to four VME busses, each of which may connect to one or two dual-string Interphase Cougar disk controllers. In our current configuration, we connect three disks to each SCSI string, two strings to each Cougar controller, and one Cougar controller to each XBUS VME interface for a total of 24 disks per XBUS board. The Cougar disk controllers can transfer data at 8 megabytes/second, for a maximum disk bandwidth to a single XBUS board of 32 megabytes/second in our present configuration.

Of the remaining two XBUS ports, one interfaces to a parity computation engine. The last port is the VME control interface linking the XBUS board to the host workstation. It provides the host with access to the XBUS board's memory as well as its control registers. This makes it possible for file server software running on the host to access network headers, file data and metadata in the XBUS memory.

2.3 Hardware Performance

In this section, we present raw performance of the RAID-II [1] hardware, that is, the performance of the hardware without the overhead of a file system. In Section 3.4, we show how much of this raw hardware performance can be delivered by the file system to clients.

Figure 5 shows RAID-II performance for random reads and writes. We refer to these performance measurements as hardware system level experiments, since they involve all the components of the system from the disks to the HIPPI network. For these experiments, the disk system is

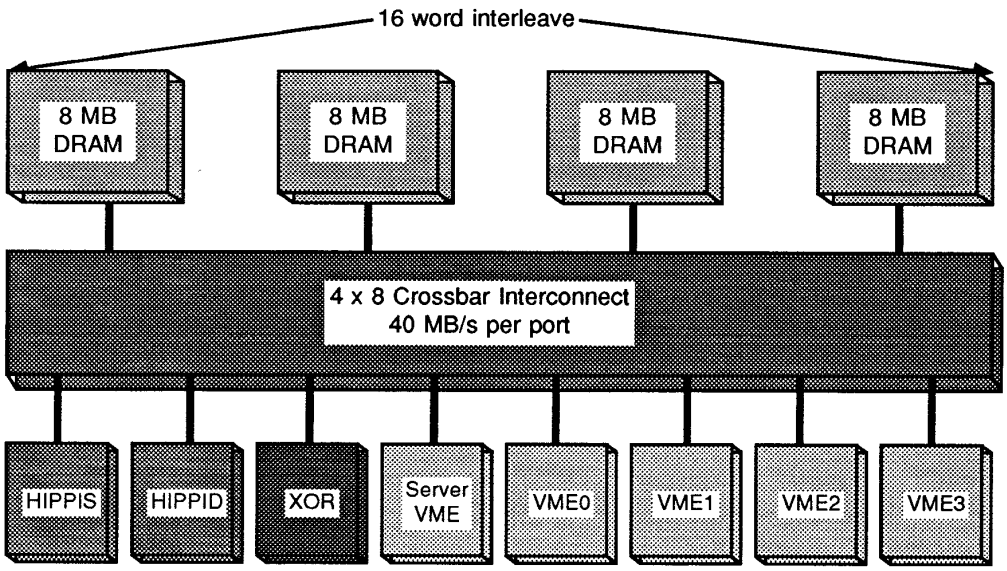

Figure 4: Structure of XBUS controller board. The board contains four memory modules connected by a 4x8 crossbar interconnect to eight XBUS ports. Two of these XBUS ports are interfaces to the HIPPI network. Another is a parity computation engine. One is a VME network interface for sending control information, and the remaining four are VME interfaces connecting to commercial SCSI disk controller boards.

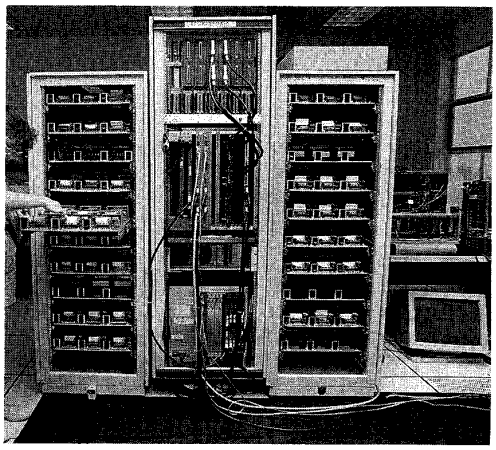

Figure 3: The physical packaging of the RAID-II File Server. Two outer racks contain 144 disks and their power supplies. The center rack contains three chassis: the top chassis holds VME disk controller boards; the center chassis contains XBUS controller boards and HIPPI interface boards, and the bottom VME chassis contains the Sun4/280 workstation.

configured as a RAID Level 5 [13] with one parity group of 24 disks. (This scheme delivers high bandwidth but exposes the array to data loss during dependent failure modes such as a SCSI controller failure. Techniques for maximizing reliability are beyond the scope of this paper [4], [16], [6].)

For reads, data are read from the disk array into the memory on the XBUS board; from there, data are sent over HIPPI, back to the XBUS board, and into XBUS memory. For writes, data originate in XBUS memory, are sent over the HIPPI and then back to the XBUS board to XBUS memory; parity is computed, and then both data and parity are written to the disk array. For both tests, the system is configured with four Interphase Cougar disk controllers, each with two strings of three disks. For both reads and writes, subsequent fixed size operations are at random locations. Figure 5 shows that, for large requests, hardware system level read and write performance reaches about 20 megabytes/second. The dip in read performance for requests of size 768 kilobytes occurs because at that size the striping scheme in this experiment involves a second string on one of the controllers; there is some contention on the controller that results in lower performance when both strings are used. Writes are slower than reads due to the increased disk and memory activity associated with computing and writing parity. While an order of magnitude faster than our previous prototype, RAID-I, this is still well below our target bandwidth of 40 megabytes/second. Below, we show that system performance is limited by that of the commercial disk controller boards and our disk interfaces.

Table 1 shows peak performance of the system when

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.