

# Logistical Storage in Active Networking: a promising framework for network services

Alessandro Bassi, Micah Beck  
LoCI Laboratory - University of Tennessee  
203 Claxton Building - 37996-3450 Knoxville, TN, USA  
abassi@cs.utk.edu, mbeck@cs.utk.edu

Jean-Patrick Gelas, Laurent Lefèvre  
RESAM Laboratory - INRIA RESO - Ecole Normale Supérieure de Lyon  
46, allée d'Italie - 69364 LYON Cedex 07 - FRANCE  
Jean-Patrick.Gelas@ens-lyon.fr, Laurent.Lefevre@inria.fr

## Abstract

*Active networks are a promising way to develop new services for data transport. But active routers could also store “on the fly” streams of data to propose high level services : web cache, reliable multicast... In this paper, we describe the merging of Active networking with Distributed storage solutions. We focus our approach on two dedicated frameworks : the IBP suite<sup>1</sup> and the Tamanoir execution environment<sup>2</sup>. We describe the overall architecture of the Active Logistical Storage suite and present first experimental results.*

*Keywords:* Active networks, logistical storage, Tamanoir, IBP

## 1 Introduction

Internet, the most explosive phenomenon of the recent years, has often been associated with highways, because of the fast and straightforward way data are delivered, giving an image of transparency of the underlying structure

---

<sup>1</sup>This work is supported by the National Science Foundation under Grants ACI-9876895, EIA-9975015, EIA-9972889, ANI-9980203, the Department of Energy under the SciDAC/ASCR program, and the University of Tennessee Center for Information Technology Research

<sup>2</sup>This work is supported by the RNTL Etoile and ANVAR Active Networking Platform.

and making people believe that between their workstations and the server containing the desired informations there is a single stretch of wire. But if we give a more attentive look to this metaphor, we can observe that in the transportation field warehouses and depots made for holding goods in transit for a limited amount of time play a role which is as important as the highways themselves, as well because of the important local decision that are taken at those points.

Together with the above considerations, as the extraordinary success of the Internet showed that the sharing of the “highways”, specifically resources such as wires and routers, makes it possible to achieve exponential growth rates, what we believe is that a similar approach to storage resources, exposing them and sharing them in an Internet-style way, can lead to a similar growth.

As the “old Internet” of static HTML pages is already giving place to a more complex model - we think of all the initiatives in the area of Web Services, for instance - we believe that Logistical Networking[1], a term introduced for the analogy with military and industrial use of warehouses and depots, and Active Networks[2] can be the base concepts of a new wave of Internet services, which might keep the growth of the movement as steep as

it has been in the recent past.

Our approach is based on active networks where a new generation of network equipments (such as routers, proxies, and gateways) apply services on the fly on data packets. These services may operate at low level (packets marking, selective drops ...) or high level (QoS, cryptography, compression on the fly ...). As active networks face various problems, especially with regards to security and high performance, we have developed an high performance execution environment called Tamanoir to deal efficiently with multi-streams active transport and dynamic services deployment in the net.

While the above results lie in the active networking field, and can be seen as an optimization of existing ideas, taking benefit of on-the-fly storage in the network is a original and unexplored approach, which leads to extremely interesting experimental results.

This paper is organized as follows: section 2 describes the involved frameworks (the Logistical Networking suite and Tamanoir). Section 3 focuses on the implementation of a logistical storage service in an active router; section 4 describes new network services, while section 5 presents our first performance results. We briefly describe related projects in section 6. In the last section, we present our future works.

## 2 Frameworks description

This section aims to describe the two projects. The Logistical Networking Infrastructure, in which the Internet Backplane Protocol plays a major part, and the Tamanoir active networking execution environment.

### 2.1 The Logistical Networking Infrastructure

#### 2.1.1 The Internet Backplane Protocol

IBP[1] is a Middleware whose purpose is to allow the sharing of distributed storage resources of any size in any network size. (See Fig. 3) What makes IBP unique is the way of *exposing* those resources: any application can allo-

cate a predictable amount of space for a predictable amount of time on a given server. This key aspect of the IBP storage model, the capacity of allocating space on a shared network resource, can be seen as doing a C-like malloc on an Internet resource, with some outstanding differences, such, for instance, time-limitation to prevent Denial-of-Use attacks.

From a networking point of view, IBP design matches in many ways the design of the Internet Protocol (IP); as IP is a more abstract service based on link-layer datagram delivery, IBP is a more abstract service based on blocks of data, managed as "byte arrays".

The IBP software, developed in C (for the server side) and C and Java (for the client library), has been successfully tested for different OSes and hardware architectures (Linux on i686 and ia64, Solaris, Windows 2000, AIX, DEC alpha, OS X).

#### 2.1.2 The exNode

As in the networking field the IP protocol alone does not guarantee many highly desirable characteristics, and needs to be complemented by a transport protocol such as TCP, in a similar way many application need stronger allocation properties, such as reliability and arbitrary, than the ones IBP is able to offer them.

The *exNode*, or *external Node* follows the example of the Unix inode: but rather than aggregating blocks on a single disk volume, the exNode is a set of storage allocations on the Internet, and the exposed nature of IBP allocations makes IBP byte arrays perfectly fit for such aggregations.

The exNode library (developed in C and Java) allows to serialize its contents using XML, to open the exNode use to XML-based tools and inter-operability with XML systems and protocols.

#### 2.1.3 The L-Bone

The *Logistical Backbone*, also known as *L-Bone*, is a distributed set of facilities that aim to provide high-performance, location- and

application-independent access to storage for network and Grid applications of all kind. In its current stage, it is based on a directory of IBP depots, and informations such as network proximity to any Internet access point can be calculated in real-time (using tools such as NWS[3]). The L-Bone client library allows users to query for depots that satisfy user-specified criteria, creating a layer of services on top of IBP.

## 2.2 Tamanoir : High Performance Active Networking

The integration of new and standard technologies into the shared network infrastructure has become a challenging task, and the growing interest in the active networking field[2] might be seen as a natural consequence. In “active” networking vision, routers or any network equipments (like gateway or proxy) within the network can perform computations on user data in transit, and end users can modify the behavior of the network by supplying programs, called *services*, that perform these computations. This kind of routers are called *active nodes* (or *active routers*) and show a greater flexibility towards the deployment of new functionalities, more adapted to architecture, users and service providers’ requirements.

The Tamanoir[4] architecture design does not interfere with the core network, mainly to guarantee higher performance results, and it’s deployed only on the network periphery (Fig. 2).

The injection of new functionalities, called services, is independent of data stream: services are deployed on demand when streams reach an active node (see Figure 1) which is not able to perform the expected data process.

Tamanoir Active Nodes (TAN) provide persistent active routers which are able to handle different applications and various data stream (such as audio/video) at the same time. The two main transport protocol (TCP and UDP) are supported by the TAN for carrying data. We use the ANEP (Active Network Encapsulated Protocol)[5] format to send data over ac-

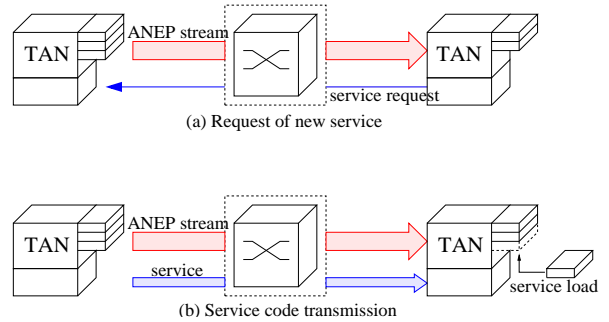


Figure 1: Service deployment between two Active Routers. When a stream reaches an active routers who doesn’t hold the required service, it sends a request to the last active router crossed by the stream. Then, the service is sent, uploaded, installed in memory and ready to process the stream.

tive networks.

For the implementation process we choose to use a common and portable language to let active networks users be able to define and write their own services. For this reason the Tamanoir execution environment has been written entirely in JAVA, because this language provides great flexibility typical of an Object-Oriented language and is shipped with standard library.

Unfortunately, the execution environment provided by the JVM (*Java Virtual Machine*) gives a very high level of abstraction, through which applications have some difficulties to reach the guts of a system. In the context of high performance this abstraction can raise a huge number of optimizations problems. Luckily, recent JVM releases ( $\geq 1.3.x$ ) provided by companies like *SUN*, *IBM* or by project like *Blackdown*[6] give excellent performance for the mainstream hardware architecture (i.e., x86). This is mainly due to the improvements in Just-In-Time compilation techniques. It is important to notice that the *GNU Compiler for the Java Programming Language*, called GCJ[7], can be used to run a TAN in native mode.

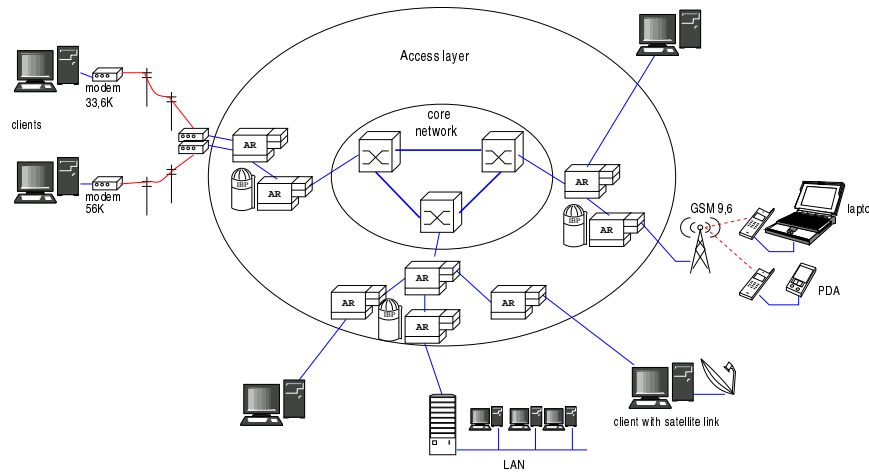


Figure 2: Tamanoir active network topology. Active Routers are edge routers. IBP depots are distributed close to the Tamanoir active nodes

### 3 Implementation : Design of IBP Services for Tamanoir

A Tamanoir Active Node (TAN) is an autonomous process running on a node (PC) waiting for processing new active streams. An IBP depot is an autonomous process which can run on the same or another local node. As shown in figure 3, Tamanoir and its services run in the application layer (the higher rectangle) and access directly the IBP layer.

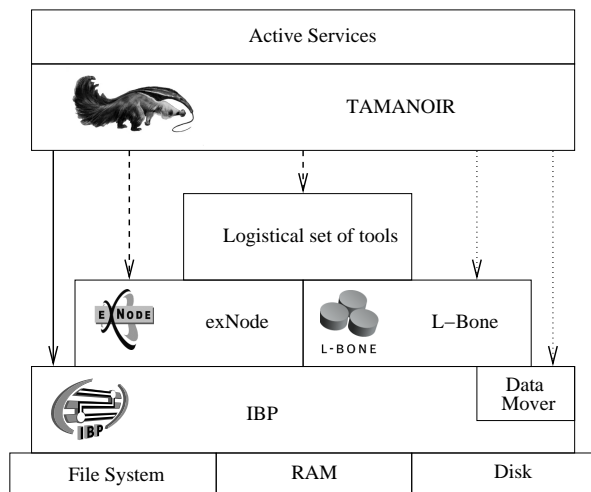


Figure 3: Tamanoir - IBP architecture

A Tamanoir service communicates with an IBP depot through a socket over a reliable transport protocol (TCP/IP). It is worth noticing that it is not mandatory for an IBP depot to run on the same node, because communications between depot and active service are done through the intermediary of sockets; but, of course, this architecture might show a better performance. The original Java client for IBP was re-used to implement our code, as a Tamanoir service is considered as a client from an IBP depot point of view. We have to instantiate IBP client classes in each service requiring to communicate with an IBP depot. These classes provide constructors and methods to create *Capabilities* (pointers to IBP allocations) on any IBP depot, with whom the Tamanoir service can write, read and manage data remotely on any IBP depot.

Figure 4 presents a very basic service for the Tamanoir Active Node. This service is designed to cache data in a IBP depot and forward them immediately. Each service is inherited from a generic Service which is a *Thread*. When this service is instantiated, the constructor `Ibp_testS()` (in our example) is called. Its goal is to allocate a storage area in the chosen IBP depot, and returns the pointers (or capabilities) to this area. When this is done, the main loop `run()` copy the ANEP payload to

the IBP depot and forward the entire ANEP packet to the receiver.

```

class Ibp_testS extends Service
{
    static final String IBP_DEPOT_NAME = "ibp01.ens-lyon.fr";
    static final int IBP_DEPOT_PORT = 6716;
    static final int CAP_SIZE = 1000 ; // KBytes
    IBPDepot dpt ;
    CapAttribute attr ;
    IBPCaps caps ;
    WriteCap wc ;
    ReadCap rc ;

    Ibp_testS()
    {
        try{
            dpt = new IBPDepot( IBP_DEPOT_NAME ,IBP_DEPOT_PORT );
            attr = new CapAttribute(0,IBPProtocol.IBP_STABLE,
                IBPProtocol.IBP_BYTEARRAY);
            caps = new IBPCaps(dpt,CAP_SIZE*1024,attr,0,0);
            wc = caps.getWriteCap();
            rc = caps.getReadCap();
        } catch (UnknownHostException e) {
        } catch (IBPException e) {...}
        } catch (IOException e) {...}
    }
    public void run()
    {
        long ret1 = 0;
        while(!FINISHED) {
            try {
                // 1/ cache payload in the local ibp depot
                ret1 = wc.write( getANEPkt().payload,
                    getANEPkt().payload.length );
                System.err.println("Ibp_testS: write in " + IBP_DEPOT_NAME +
                    " depot : " + ret1 + " bytes");
                // 2/ send cached data immediately
                forward();
            } catch (IOException e) {
                System.err.println("Ibp_testS 1: " + e);
            } catch (IBPException e) {
                System.err.println("Ibp_testS 2: " + e);
            }
            // 3/ wait for the next ANEP packet
            rcv();
        } // while(!FINISHED)
    } // run()
    public void process() {}
} // class Ibp_testS

```

Figure 4: IBP service in TAN

## 4 Network services

In this section, we illustrate the current services and discuss about the future ones, which can be used by applications willing to take advantage of the mixture of the IBP storage capabilities and Tamanoir dynamic behavior alteration capabilities.

### 4.1 Data caching

As the Tamanoir project was planned essentially as a high performance active networking environment for the transmission of multimedia streams, using some unreliable protocol like

UDP for applications like VoD (Video on Demand), the packet loss was tolerated. On the other hand, what was not acceptable was losing packets because the service that should have taken care of the data was not present. Therefore, when a Tamanoir Active Node (TAN) received a stream without holding the service needed, the first packets were simply discarded until the service was loaded and installed in the TAN memory. To correct this behaviour, we run an IBP depot on the local node (where the TAN is running) in order to cache data (see Fig.5).

This first new service, called *IBPService* uses a *IBP\_store* operation to redirect the data stream beginning towards the IBP depot. The IBP service checks as well the presence of the required service each time that a new packet arrive, and if so, a *IBP\_load* operation is done to redirect all the data cached in the IBP depot towards the service able to process, route and forward efficiently these data. The only difference between the *IBPService* and any other service lies in the load-time, which is done at boot time for the *IBPService*, in order to be able to cache data immediately.

This service is also useful for a reliable transport protocol such as TCP. As we said above, until the service was loaded and ready to operate, all the packets were dropped. In case of a reliable connection, as we don't have the "right" to loose data, the TAN should send a message to the server to inform it about the loss of packets and ask him to retransmit. As above, with an IBP depot this is not necessary because data are cached in the local depot until the TAN is able to process and transmit them efficiently. With a FIFO spirit, cached data are sent first, and if the capability gets empty, the remaining data are forwarded to the service without passing through the local IBP depot.

### 4.2 Reliable Multicast

There are three benefits of performing storage in the network for a reliable multicast application. First, we can look at the TAN with its depot as a kind of mirror for data distri-

bution, to download them from the geographically (or network) closest point to the consumers. Next, clients can consume data with their own processing speed capabilities without disturbing the server where data come from, and finally, a TAN can retransmit lost data without uselessly overloading the network between the server and the TAN.

The above considerations led us to design a second service, called *XcastIbpService* and it is targeted for multicast applications. Figure 5 shows a view of multicast topology with on the left a data server (e.g push server, ASP server) and on the right heterogeneous clients in terms of connections speed and processing capabilities. Because clients don't ask the same data at the same time and don't consume them at the same speed we put an IBP depot in (or close to) a TAN to cache data. For this application, a TAN can be seen has a multicast node with caching capabilities. Server have just to send only once data towards the closest active node to the clients. Then, clients can consume data at their own speed.

For the first experiments, we implemented a very simple multicast algorithm, but we are currently implementing a more complex algorithm of reliable multicast called DyRAM (Dynamic Replier Active Reliable Multicast)[8].

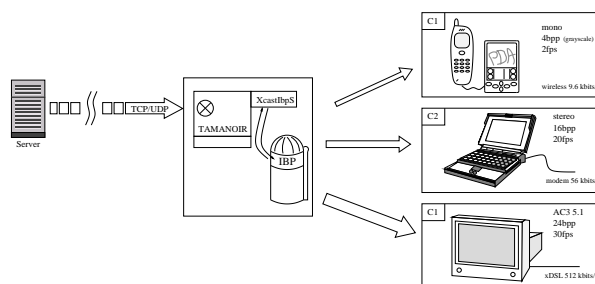


Figure 5: Reliable multicast with Tamanoir and IBP depot

### 4.3 IBP-mail done with Tamanoir

IBP-mail [9] allows a user to send an attachment of whatever size without overloading the mail server by making use of the network storage available through a set of IBP storage depots. The file is stored in a IBP depot for the duration specified. The receiver can download the file at its convenience, using a tool developed for this purpose.

For this application we would like to leverage the processing capability of a TAN for managing the movement from an IBP depot close to the sender to an IBP depot close to the receiver in order to take advantage of proximity (i.e high throughput and low latency) of the attachments, which are usually big.

This kind of application involves the use of processing capabilities in network to route attachment towards the nearest IBP depot to the receiver. The notion of proximity in computer networks is still an active open problem [3] and can be measured with different metrics (round-trip time, latency, throughput, space, hops numbers...)

Since the need to move very large digital objects around within the science, engineering, financial and even graphic multimedia community is getting stringent, we expect that the capacity to manage extremely large attachments will be fundamental for the e-community.

## 5 Raw performance results

In this section we measure the overhead introduced by the caching action in an IBP depot close to the Tamanoir Active Node. For these results we ran an IBP depot on the same node than the TAN, with the *XcastIbpService* (described in section 4.2).

Our testbed, disconnected from the production network, is set up with one active Tamanoir node which is a PC (Dual-Pentium III, 1 Ghz, 256MB RAM) shipped with several *Ethernet 100Mbits* network interface cards and a standard IDE hard-drive. We link to this PC different other PC who have the role to either feed the network or receive data. These

PCs run under GNU/Linux with a 2.4.16 kernel (distribution *Debian 2.2r3*).

When an ANEP packet reaches the TAN, its payload is extracted and sent towards the required service. At the beginning, the service copies the payload in the local IBP depot. For the measures we have set up only one client which consume data as fast as possible. Next, the payload is read in the cache and forwarded to the client immediately. For each packet, we measure the time to cross a TAN (latency) with and without caching action. A chronometer is located as low as possible in the kernel. We use *NetFilter* [10] which is a package for filtering in the Linux kernel. NetFilter allows to introduce code just behind the network interface card. When a packet reach the TAN, our NetFilter module analyses the packet, if it is an ANEP packet a chronometer is started. When the same ANEP packet leave the TAN, the chronometer is stopped.

In figure 1 we present the latency of a TAN by an ANEP packet. In the first case (lower curve) packets are immediately forwarded and not cached in the IBP depot, in the second case packets are cached and next forwarded.

As shown, caching doesn't introduce a very important overhead. For any packet size, overhead remains constant. For small packets size, performances are weak due too the policy of transmission of Linux kernel TCP implementation which tries to aggregate small packets before being transmitted. So, small packets are released just after a timeout.

Payload size (bytes)	200	5K	10K	20K	25K
Forward	15.2	4.4	11.0	21.3	26.9
IBP+Fwd	17.9	5.3	12.7	25.2	31.8
Overhead %	17	20	15.4	18.3	18.2

Table 1: Overhead (ms) introduced by caching data in a local IBP depot

## 6 Related works

Despite intensive search, we do not find any related projects directly merging distributed logistical storage with active networking.

IBP occupies an architectural niche similar to that of network file systems such as AFS[11] and Network Attached Storage appliances , but its model of storage is more primitive, making it similar in some ways to Storage Area Networking (SAN) technologies developed for local networks. In the Grid community, projects such as GASS [12] and the SDSC Storage Resource Broker [13] are file system overlays that implement a uniform file access interface and also impose uniform directory, authentication and access control frameworks on their users.

Active networks projects using storage facilities are not intensively addressed. Most related projects are Active Disks [14] or active-network-storage [15] which explore the increasing of intelligence in Network Attached Storage.

## 7 Conclusion

By merging two environments like IBP and Tamanoir, we allow users and network operators to provide new kind of high level services.

The benefits for the frameworks are double. Tamanoir can use storage distributed facilities (IBP depots) for service deployment and temporary storage of data streams. It is now easy to transform an active router in an active proxy with on the fly storage capabilities. The latency added by a dynamic storage is small. IBP benefits from Tamanoir dynamic service deployment infrastructure.

As adding storage in the network could also mean to add disks in the routers, which is an unusual approach for networks operators (low MTBF, failure...), a RAM-based version of IBP is currently under design by LoCI Laboratory.

Our next step will consist in the design of high level network services dedicated to Grid computing Middleware requirements. More information on both frameworks can be found on

the following sites :

- Tamanoir : [http://www.ens-lyon.fr/LIP/RESAM/index\\_activenetwork.html](http://www.ens-lyon.fr/LIP/RESAM/index_activenetwork.html)
- IBP : <http://loci.cs.utk.edu/ibp>

## References

- [1] J. Plank, A. Bassi, M. Beck, T. Moore, M. Swamy, and R. Wolski. Managing data storage in the network. *IEEE Internet Computing*, 5(5), september 2001.
- [2] David Tennenhouse and David Wetherall. Towards an active network architecture. *Computer Communications Review*, 26(2):5–18, April 1996.
- [3] R. Wolski. Forecasting network performance to support dynamic scheduling using the network weather service. In IEEE Press, editor, *6th IEEE Symp. on High Performance Distributed Computing, Portland, Oregon*, 1997.
- [4] Jean-Patrick Gelas and Laurent Lefèvre. Tamanoir: A high performance active network framework. In C. S. Raghavendra S. Hariri, C. A. Lee, editor, *Active Middleware Services, Ninth IEEE International Symposium on High Performance Distributed Computing*, pages 105–114, Pittsburgh, Pennsylvania, USA, August 2000. Kluwer Academic Publishers. ISBN 0-7923-7973-X.
- [5] S. Alexander, B. Braden, C. Gunter, A. Jackson, A. Keromytis, G. Minden, and D. Wetherall. Active network encapsulation protocol (anep). RFC Draft, Category : Experimental, July 1997.
- [6] Blackdown. Blackdown java development kit. <http://www.blackdown.org/>.
- [7] GCJ. The GNU Compiler for the Java Programming Language. <http://sourceware.cygnus.com/java/>.
- [8] M. Maimour and C. Pham. A throughput analysis of reliable multicast protocols in an active networking environment. In *Proceedings of the Sixth IEEE Symposium on Computers and Communications (ISCC 2001)*, pages 151–158, July 2001.
- [9] Wael Elwasif, James Plank, Micah Beck, and Rich Wolski. Ibp-mail: Controlled delivery of large mail files. In *Net-Store'99: Network Storage Symposium*, Seattle, WA, October 1999.
- [10] Rusty Russell. Linux Filter Hacking HOWTO. july 2000.
- [11] J.H. Morris, M. Satyanarayan, M.H. Conner, J.H. Howard, D.S.H. Rosenthal, and F.D. Smith. Andrew: A Distributed Personal Computing Environment. *Communication of the ACM*, 29(3):184–201, 1986.
- [12] J. Bester, I. Foster, C. Kesselman, J. Tedesco, and S. Tuecke. Gass: A data movement and access service for wide area computing systems. In *Sixth Workshop on I/O in Parallel and Distributed Systems*, may 1999.
- [13] C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC Storage Ressource Broker. In *CASCON'98*, Toronto, Canada, 1998.
- [14] Erik Riedel. Active disks - remote execution for network-attached storage. Technical Report CMU-CS-99-177, Electrical and Computer Engineering Carnegie Mellon University, Pittsburgh, PA 15213, Nov. 1999.
- [15] A. Tomita, Y. Takamoto, S. Inohara, H. Odawara, F. Maciel, M. Sugie, and N. Watanabe. A Scalable, Cost-Effective, and Flexible Disk System Using High-Performance Embedded-Processors. In *IEEE International Conference on Parallel Processing*, 2000.