RAID-x: A New Distributed Disk Array for I/O-Centric Cluster Computing

Kai Hwang[1], Hai Jin[1,2], and Roy Ho[2]
University of Southern California[1]
The University of Hong Kong[2]
Email: {kaihwang, hjin}@ceng.usc.edu

Abstract

A new RAID-x (*redundant array of inexpensive disks at level x*) architecture is presented for distributed I/O processing on a serverless cluster of computers. The RAID-x architecture is based on a new concept of *orthogonal striping and mirroring* (OSM) across all distributed disks in the cluster. The primary advantages of this OSM approach lie in: (1) a significant improvement in parallel I/O bandwidth, (2) hiding disk mirroring overhead in the background, and (3) greatly enhanced scalability and reliability in cluster computing applications. All claimed advantages are substantiated with benchmark performance results on the Trojans cluster built at USC in 1999. Throughout the paper, we discuss the issues of scalable I/O performance, enhanced system reliability, and striped checkpointing on distributed RAID-x in a serverless cluster environment.

1. Introduction

In a serverless cluster of computers, no central server is used. The conventional client-server architecture does not apply in this class of multicomputer clusters. Instead, all client hosts in the cluster divide the server functions in a distributed manner. Often, a serverless cluster applies a distributed RAID architecture, which is built over dispersed disks physically attached to client hosts. This paper presents a new RAID-x architecture for high-bandwidth, distributed I/O processing on serverless clusters. In early RAID architecture [5], all disks are densely packaged in the same rack with a centralized control. Centralized RAIDs are often attached to a storage server or as a network-attached I/O subsystem.

The TickerTAIP [4] project offered a parallel RAID architecture for supporting parallel disk I/O with multiple controllers. Our paper deals with *distributed* RAID architectures. Distributed RAID concept was explored by Stonebraker and Schloss [27]. Prototyping of distributed RAIDs started with the Petal project [19] at Digital Lab and the Tertiary Disk project [28] at UC Berkeley. This paper reports the architecture and performance of a new distributed RAID-x system, built at the USC Trojans cluster project. The level x in RAID-x is yet to be assigned by the RAID Advisory Board [24].

To build a distributed RAID, one must establish three capabilities: (i) a single I/O space (SIOS) for all disks in the cluster, (ii) high scalability, availability, and compatibility with current cluster architectures and applications, and (iii) local and remote disk I/O operations performed with comparable latency. These requirements

Table 1 Research Projects on Parallel and Distributed RAIDs

System Attributes	USC RAID-x [15]	Princeton TickerTAIP [4]	Digital Petal [19][29]	Berkeley Tertiary Disk [28]
RAID architecture environment	Orthogonal striping and mirroring over a Linux cluster	RAID-5 with multiple controllers as a centralized subsystem	Chained declustering in an Unix cluster	A RAID-5 built with a Solaris PC cluster
Enabling mechanism for SIOS	Cooperative device drivers in Linux kernel	Single server implements the SIOS	Petal device drivers at user level	xFS storage servers at file system level
Data consistency Maintenance	Locks at device driver level	Sequencing of user requests	Frangipani file system	Locks in the xFS file system
Reliability Implementation	OSM and striped checkpointing	Parity checks in RAID-5	Striped mirroring	SCSI disks with parity checks

1082-8907/00 $10.00 © 2000 IEEE

imply a total transparency to the users, who can utilize all disks without knowing the physical locations of the data blocks. The development of the RAID-x architecture was inspired by several projects summarized in Table 1.

The enabling mechanisms for SIOS are quite different among the above distributed RAID architectures. Petal achieves the SIOS with user-level device drivers. The Tertiary Disk uses the xFS file system [1] to yield the SIOS. We realize the SIOS with cooperative device drivers at the Linux kernel level. The four RAID architectures differ also in their handling of the data consistency problem in order to establish a global file hierarchy. Locks are implemented at our device drivers at Linux kernel level. Furthermore, the reliability is also supported different mechanisms in the four I/O subsystems.

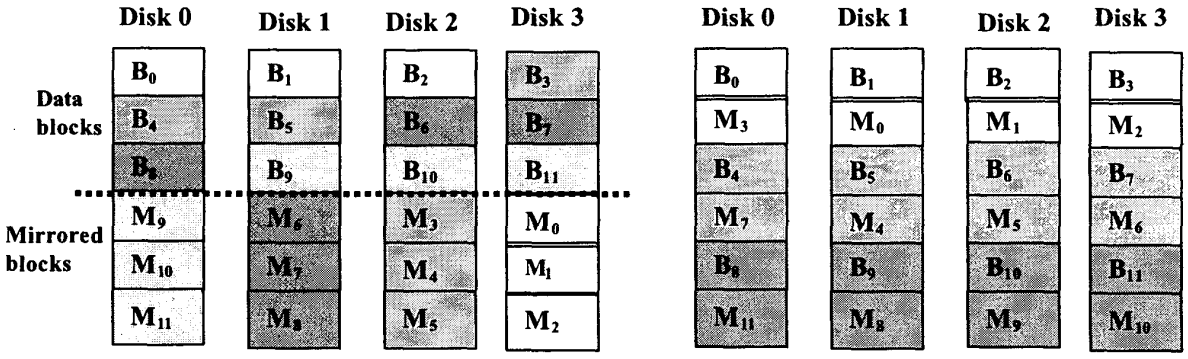
2. Orthogonal Striping and Mirroring

In addition to those projects listed in Table 1, our new RAID-x architecture is influenced by the following RAID projects: The AFRAID [26], AutoRAID [31], and the chained declustering. Our RAID-x differs from the above distributed RAID architectures in mainly two aspects. First, the RAID-x is built with a new disk mirroring technique, called *orthogonal striping and mirroring* (OSM). The small write problem associated with RAID-5 is completely eliminated in this OSM approach. Second, we have developed *cooperative disk drivers* (CDD) to implement the OSM at the kernel level. The CDD maintains data consistency directly without using NFS or inter-space Unix system calls.

Figure 1 shows the architecture of RAID-x (Fig.1a) along with the chained declustering RAID (Fig.1b). The original *data blocks* are denoted as B_i. The corresponding *mirrored blocks* are labeled as M_i. With 4 disks in Fig.1a, each *mirroring group* involves 3 consecutive disk blocks, such as (M_0, M_1, M_2) for data blocks (B_0, B_1, B_2). Different mirroring groups are distinguished with different shadings in the disk blocks. Data blocks in RAID-x (Fig.1a) are striped across all disks on the top half of the disk array, behaving like a RAID-0. The image blocks (such as M_0, M_1, M_2) are "clustered" in the same disk (Disk 3) vertically. For a stripe group of 4 blocks, the image blocks are saved in exactly two disks, occupying the lower half of the disk array. The clustered images in a mirroring group are simultaneously updated at the background. This results in a lower latency and higher bandwidth in using the RAID-x.

The new OSM concept has the advantages of both RAID-1 and chained declustering. The RAID-x completely avoids the small write problem associated with the RAID-5. The orthogonal mapping implies that no data block and its image should be mapped in the same disk. For large write, the data blocks are written in parallel to all disks in the stripe simultaneously. The image blocks are gathered as a long block written into the same disk. Table 2 presents the expected peak performance of four RAID architectures.

Let n be the number of disks in RAID subsystem, B be the maximum bandwidth per disk, and m be the number of blocks in a file. R and W refer to average block read and write time, respectively. RAID-x shows the same bandwidth potential as RAID-0 and chained declustering. It can tolerate single-disk failures, same as RAID-5. RAID-x improves from the chained declustering RAID mainly in parallel write operations. For large array size, the improvement factor approaches two. Thus the RAID-x performs much better in write operations than all other three RAID architectures in Table 2.



(a) Orthogonal striping and mirroring (OSM) in the RAID-x

(b) Skewed mirroring in a chained declustering RAID

Figure 1. Disk mirroring schemes in the RAID-x and in a chained declustering RAID

Table 2 Expected Performance of Four RAID Architectures

Performance Indicators		RAID-1	RAID-5	Chained Declustering	RAID-x
Max. I/O Bandwidth	Read	nB	$(n\text{-}1)B$	nB	nB
	Large Write	nB	$(n\text{-}1)B$	nB	nB
	Small Write	nB	$nB/4$	nB	nB
Parallel Read or Write Time	Large Read	$2mR/n$	$mR/(n\text{-}1)$	mR/n	mR/n
	Small Read	R	R	R	R
	Large Write	$2mW/n$	$mW/(n\text{-}1)$	$2mW/n$	$mW/n + mW/n(n\text{-}1)$
	Small Write	W	$R+W$	$2W$	$\approx W$
Max. Fault Coverage		$n/2$ disk failures	Single disk failure	$n/2$ disk failures	Single failure

3. Trojans Cluster and Experiments

AT USC Internet and Cluster Computing Laboratory, a prototype Linux PC cluster was built with 16 Pentium II/400MHz processors, running the Redhat Linux 6.0 with kernel 2.2.5. These PC engines are connected by a 100 Mbps Fast Ethernet switch. This Linux cluster became operational since Fall 1999. At present, each node is attached with a 10-GB disk. With 16 nodes, the total capacity of the disk array is 160 GB. All 16 disks form a single I/O space. Figure 2 shows the front view of the prototype Trojans cluster. This cluster is connected to the Internet over fiber links. More information about the Trojans project is available at the Web site: http://andy.usc.edu/trojan/.

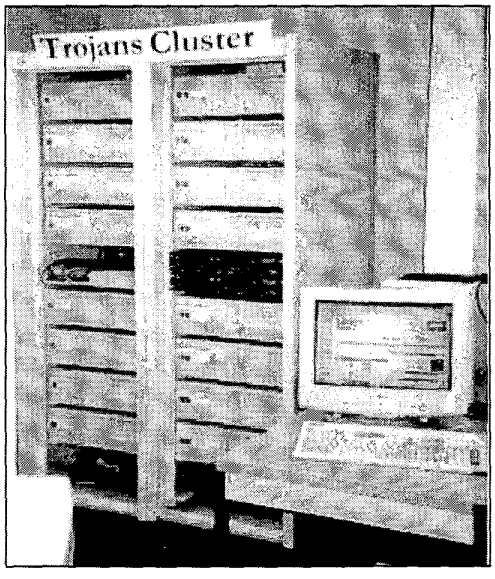


Figure 2 The Trojans cluster built in 1999 at USC Internet and Cluster Computing Lab.

Figure 3 shows a two-dimensional RAID-x architecture with 3 disks attached to each node. We call this a 4 x 3 disk array configuration. All disks within the same stripe group, such as (B_0, B_1, B_2, B_3), are accessed in parallel. Consecutive stripe groups, such as (B_0, B_1, B_2, B_3), (B_4, B_5, B_6, B_7), and (B_8, B_9, B_{10}, B_{11}) can be accessed in a pipelined fashion, because they are retrieved from disk groups attached to the same SCSI buses.

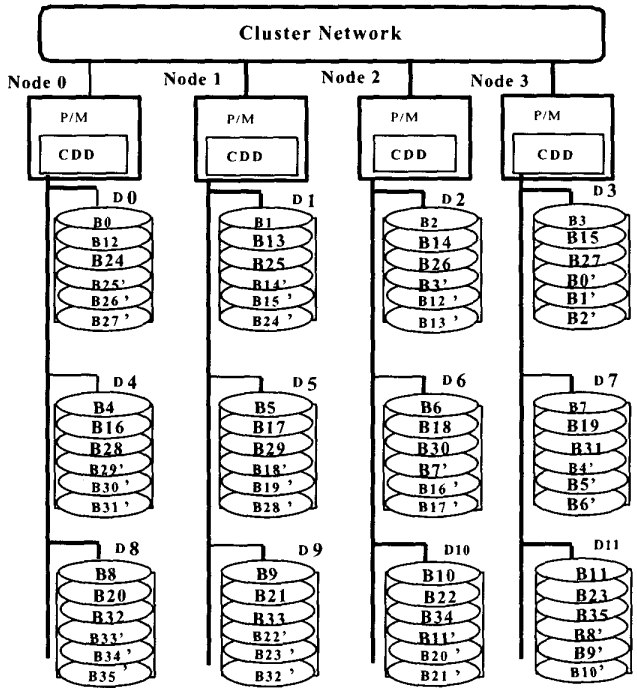


Figure 3 A 4 x 3 RAID-x architecture with orthogonal striping and mirroring
(P: processor, M: memory, CCD: cooperative disk driver, Dj: the j-th disk, B_i: the i-th data block, B_i' the i-th mirrored image in a shaded block)

In general, an n-by-k RAID-x configuration has a *stripe group* involving n disk blocks residing on n disks. Each *mirroring group* has n-1 blocks residing on one disk. The images of all data blocks in the same stripe group are saved on exactly two disks. The block addressing scheme stripes across all nk disks sequentially and repeatedly. The parameter n represents the degree of parallelism in accessing the disks. The parameter k implies the depth of pipelining. Tradeoffs do exist between these two concepts. Both concepts will be applied to implement a new checkpointing scheme for the RAID-x in Section 6.

4. Cooperative Disk Drivers

The *Single I/O space* (SIOS) is crucial to building scalable cluster of computers. A loosely coupled cluster use distributed disks driven by different hosts independently. The *independent* disk drivers handle distinct I/O address spaces. Without the SIOS, remote disk I/O must be done by a sequence of time-consuming system calls through a centralized file server (such as the use of NFS) across the cluster network.

On the other hand, the CDDs work together to establish the SIOS across all physically distributed disks. Once the SIOS is established, all disks are used collectively as a single *global virtual disk* shown in Fig.4a. Each node perceives the illusion that it has several physical disks attached locally.

Figure 4b shows the internal design of a CDD. Each CDD is essentially made from three working modules. The *storage manager* receives and processes the I/O requests from remote *client modules*. The client module redirects local I/O requests to remote disk managers.

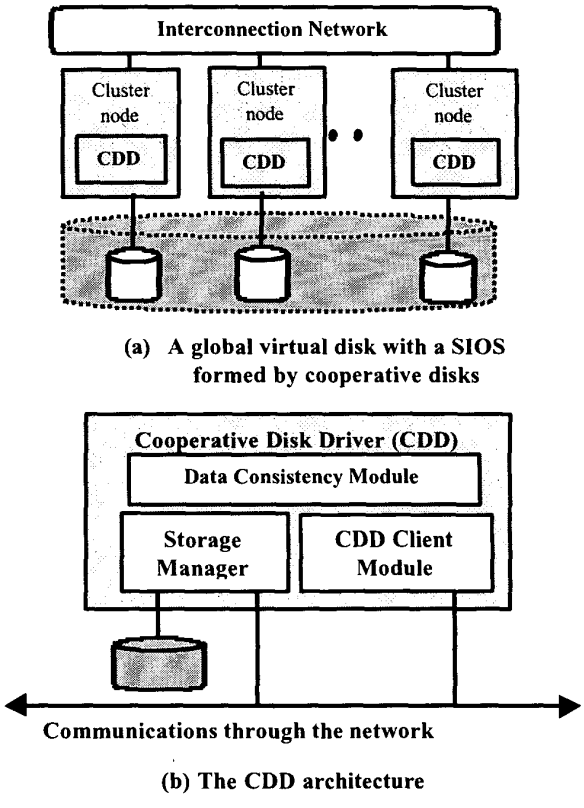
Data consistency problems arise when multiple cluster nodes have cached copies of the same set of data blocks. The xFS approach and the Frangipani approach maintain the data consistency at the file system level. In our design, data consistency checking is maintained at driver level.

The *consistency module* is responsible for maintaining data consistency among distributed disks. A CDD can be configured to run as a storage manager or as a client, or both at the same time. Three possible states of each disk: are (1) a manager to coordinate use of local disk storage by remote nodes, (2) a client accessing remote disks through remote disk managers, and (3) both of the above functions.

The OSM scheme outperforms the chained declustering scheme mainly in parallel write operations. The RAID-x scheme demonstrates scalable I/O bandwidth with much reduced latency in a cluster environment. Both Petal and Tertiary Disk achieve the SIOS at the user level. We achieved the SIOS at the Linux kernel level. Using the CDDs, the cluster can be built serverless and offers remote disk access directly at the kernel level.

Parallel I/O is made possible on any subset of local disks, because all distributed disks form SIOS. No heavy cross-space system calls are needed to perform remote file access. A device masquerading technique is adopted here. Multiple CDDs run cooperatively to redirect I/O requests to remote disks. Our approach simplifies the design and implementation of distributed file management services. Data consistency is maintained by all CDDs with higher speed and efficiency at the data block level.

We introduced a special lock-group table for developing distributed file management services. Each record in this table corresponds to a group of data blocks that have been granted to a specific CDD client with write permissions. The write locks in each record are granted and released atomically. This lock-group table is replicated among the data consistency modules in the CDDs. Which guarantee that file management operations are performed atomically.



(a) A global virtual disk with a SIOS formed by cooperative disks

(b) The CDD architecture

Figure 4 Single I/O space in RAID-x built at Linux kernel level.

5. Benchmark Performance Results

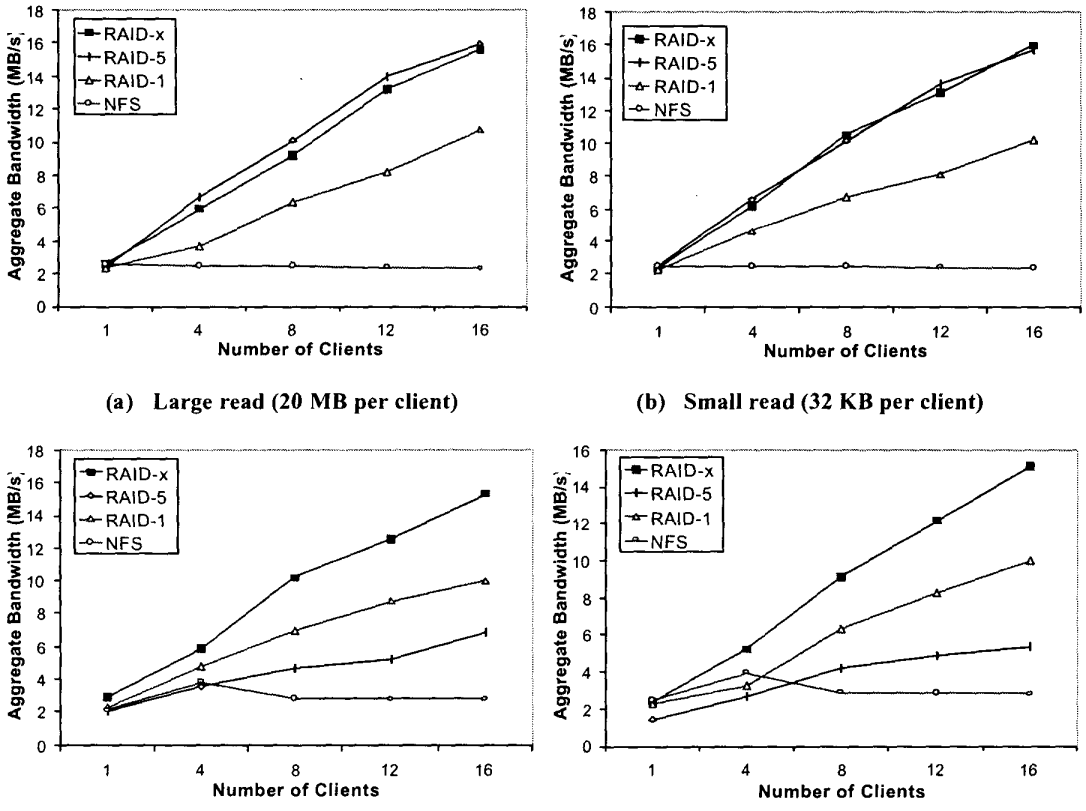
In this section, we report the experimental results obtained in parallel disk I/O and Andrew benchmark experiments. The experiments were designed to test the scalability of different RAID architectures, with respect to increasing number of client requests or increasing number of disks in each of the disk array configurations, implemented on the Trojans cluster. We show the aggregate I/O bandwidth and Andrew benchmark results, followed with a scalability analysis.

5.1. I/O bandwidth vs. request number

Figure 5 shows the performance of four I/O subsystem architectures. For large read and large write, each client accesses a large file of 20MB long, striping across all disks in the array. Therefore, the test is truly focused on parallel I/O capability of the disk array. All files are uncached and each client only reads its own private file. All reads are performed simultaneously using the MPI_Barrier() command.

In case of small read or small write, 32KB data is accessed in one block of the stripe group. The results for small read are very close to that for large read. For parallel writes, RAID-x achieves the best scalability among the four with a 15.3MB/s for 16 clients. In contrast, RAID-5 scales slowly due to the heavy overhead involved in parity calculations. RAID-1 scales even better than RAID-5.

Table 3 shows the improvement factor of 16 clients over 1 client on the USC Trojans cluster. The RAID-x demonstrates the highest improvement factor among the three RAID architectures. Almost three times increase in I/O bandwidth was observed on RAID-x, as the I/O request increases from 1 to 16.



(a) Large read (20 MB per client)

(b) Small read (32 KB per client)

(c) Large write (20 MB per client)

(d) Small write (32 KB per client)

Figure 5 Aggregate I/O bandwidth of three RAID architectures compared with the NFS on the Trojans cluster

Table 3 Achievable I/O Bandwidth and Improvement Factor on the Trojans Cluster

I/O Operations	NFS			RAID-x		
	1 Client	16 Clients	Improve	1 Client	16 Clients	Improve
Large Read	2.58 MB/s	2.3 MB/s	0.89	2.59 MB/s	15.63 MB/s	6.03
Large Write	2.11 MB/s	2.77 MB/s	1.31	2.92 MB/s	15.29 MB/s	5.24
Small Write	2.47 MB/s	2.81 MB/s	1.34	2.35 MB/s	15.1 MB/s	6.43
I/O Operations	RAID-5			RAID-1		
	1 Client	16 Clients	Improve	1 Client	16 Clients	Improve
Large Read	2.32 MB/s	15.97 MB/s	6.88	2.37 MB/s	10.76 MB/s	4.54
Large Write	2.06 MB/s	6.83 MB/s	3.32	2.31 MB/s	9.96 MB/s	4.31
Small Write	1.45 MB/s	5.36 MB/s	3.69	2.27 MB/s	9.98 MB/s	4.39

5.2. Andrew benchmark results

Andrew benchmark is a popular benchmark to test the performance of a file system. In this experiment, the Andrew benchmark was executed on four I/O subsystems with respect to increasing number of client requests up to 32. The performance is indicated by the elapsed time in executing the Andrew benchmark on the target I/O subsystem. These tests demonstrate how the underlying storage structures can affect the performance of the file system being supported. Figure 6 shows the benchmark results for four I/O subsystems.

The NFS shows a poor performance especially in reading files, scanning directories, and copying files. The other three RAID architectures do not share this weakness. The elapsed time to copy files in RAID-5 increases with the number of clients. This is mainly due to the small write problem associated with the RAID-5, because most files copied in Andrew benchmark are small in size.

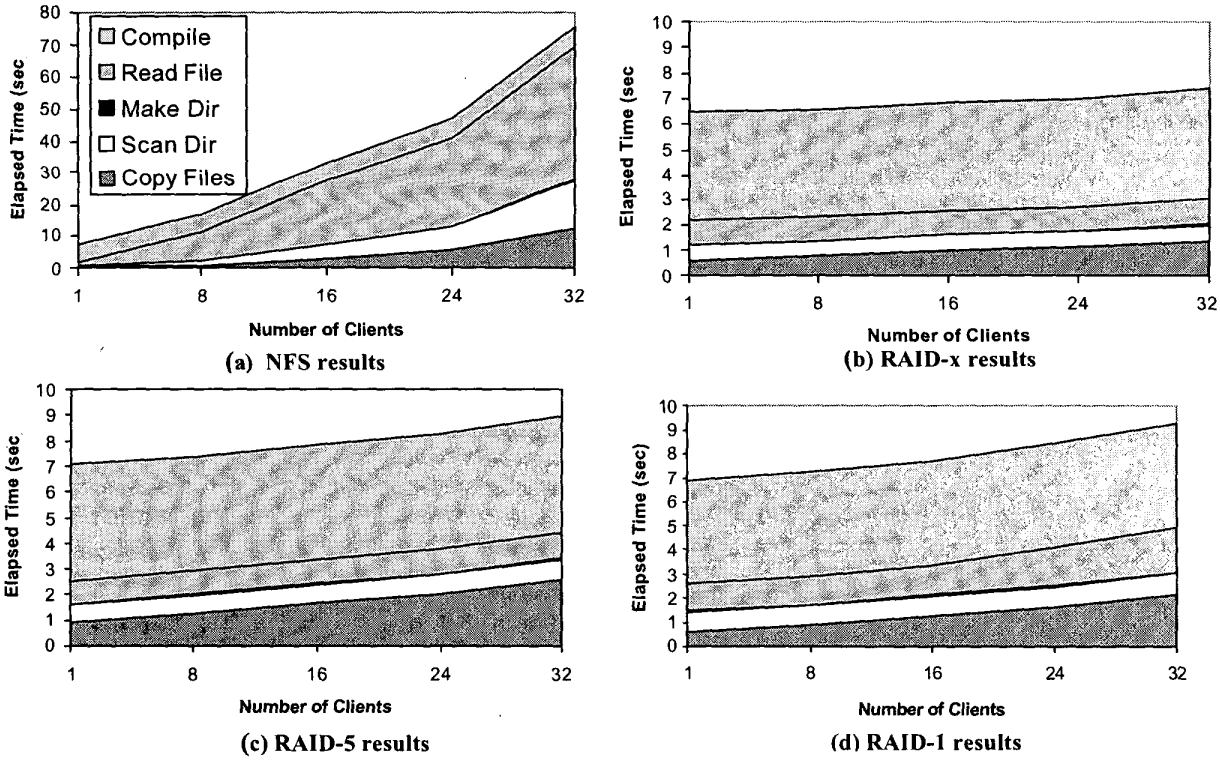


Figure 6 Elapsed times of the Andrew benchmark on the Trojans cluster with the NFS, RAID-x, RAID-5 and RAID-1 configurations

RAID-x shows a slow increase in the elapsed time in all 5 phases of the Andrew benchmark. The overall execution time of the RAID-x outperforms both RAID-1 and RAID-5 by a factor of 7% to 22% for up to 32 I/O clients. Comparing with NFS, RAID-x shows a 700% improvement in Andrew benchmark performance.

6. Striped Checkpointing on RAID-x

The parallel I/O capability of the distributed RAID-x can be applied to achieve fast checkpointing in the cluster system. A striped checkpointing scheme was introduced [23]. This scheme takes advantage of the striping feature of a RAID system. Simultaneous writing of multiple processes may cause a network contention and I/O bottleneck problem to a central stable storage.

To alleviate the network contention, Vaidya [32] has proposed a staggered writing scheme for centralized storage. However, the Vaidya scheme cannot solve the I/O bottleneck problem to access a central stable storage.

We solve both problems by distributing data blocks and their mirrored images orthogonally. Figure 7 shows the concept of striped staggering in coordinated checkpointing on the RAID-x disk array. Successive stripes are accessed in a staggered manner from different stripes on successive 4-disk groups, as demonstrated in Fig.3.

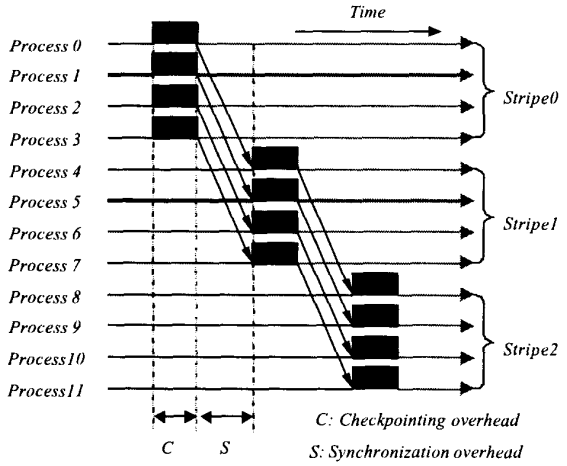


Figure 7 Striped checkpointing with staggering on a distributed RAID-x

Staggering implies pipelined accesses of the disk array. As proved in [23], there exists trade-off between striped parallelism and staggering depth. The layout in Fig.7 can be reconfigured from a 4 x 3 array to a 6 x 2 array, if pipelined access shows less advantage.

Using the OSM, each striped checkpointing file has its mirrored image in its local disk. For each node, transient failures can be recovered from its mirrored image in local disk. Permanent failures can be recovered from striped checkpointing.

The proposed RAID-x architecture can recover from any single disk failure in each stripe group. The total number of disk failure depends on the number of stripe groups to be accessed. For the 4 x 3 array in Fig.3, up-to-3 disk failures in 3 stripe groups can be tolerated.

7. Conclusions

The new RAID-x architecture shows its strength in building distributed, high-bandwidth, I/O storage for serverless PC or workstation clusters. The RAID-x is unique with the OSM architecture, which exploits full-stripe bandwidth, similar to what a RAID-0 can provide. The reliability comes from clustered mirroring on local disks, while orthogonal striping across distributed disks.

We have developed new disk drivers at Linux kernel level to support efficient implementation of the OSM. These device drivers enable remote disk accesses and parallel disk I/O without using a central file server. RAID-x matches RAID-5 in reliability, both are capable of recovering from single disk failures.

The I/O performance of the RAID-x is experimentally proven superior to RAID-1 or RAID-5, evidenced by the measured I/O bandwidth on the Linux cluster at USC. For parallel reads with 16 active clients, the RAID-x achieved 9.7 MB/s throughput, 1.5 and 3.7 times higher than using RAID-1 and NFS, respectively.

For small writes, RAID-x achieved 9 MB/s, 3 times higher than using RAID-5. Running the Andrew benchmark, RAID-x results in a 17% cut in elapsed time, compared with those experienced on a RAID-1 or on a RAID-5 configuration. The achieved I/O bandwidth corresponds to 78% of the limit of the Fast Ethernet.

Linux extensions and reliable middleware streamline the single I/O space, shared virtual memory, global file management, and distributed checkpointing in cluster operations. The new RAID-x design is shown highly scalable with distributed control.

In next phase of the Trojans project, we will develop a distributed file system with I/O load balancing capabilities along with an enlarged RAID-x prototype of several hundreds of disks on a much larger Trojans cluster. In addition to implement RAID-1, RAID-5, and RAID-x configurations, we will also consider other configurations, such as RAID-10 and chained declustering.

Scalable I/O bandwidth makes the RAID-x especially appealing to I/O-centric cluster applications, such as biological sequence analysis, collaborative engineering design, secure E-commerce and data mining, specialized digital libraries, and distributed multimedia processing.

References

[1] T. Anderson, M. Dahlin, J. Neefe, D. Patterson, D. Roselli, and R. Wang. "Serverless Network File Systems". *ACM Trans. on Computer Systems*, Jan. 1996, pp.41-79.

[2] Rajkumar Buyya (ed.). *High Perf. Cluster Computing*, Prentice Hall PTR, New Jersey, 1999.

[3] L. F. Cabrera, and D. E. Long, "Swift: Using Distributed Disk Striping to Provide High I/O Data Rates", *Proceedings of USENIX Computing Systems*, Fall 1991, pp.405-433.

[4] P. Cao, S. B. Lim, S. Venkataraman, and J. Wilkes, "The TickerTAIP Parallel RAID Architecture", *ACM Trans. on Computer System*, Vol.12, No.3, August 1994, pp.236-269.

[5] P. M. Chen, E. K. Lee, G. A. Gibson, R. H. Katz and D. A. Patterson; "RAID: High-Performance, Reliable Secondary Storage", *ACM Computing Surveys*, Vol.26, No.2, June 1994, pp.145-185.

[6] M. Dahlin, R. Wang, T. Anderson, D. Patterson. "Cooperative Caching: Using Remote Client Memory to Improve File System Performance". *Proc. of Operating System Design and Implementation*, 1994.

[7] I. Foster and C. Kesselman, (Eds.). *The Grid: Buleprint for a New Computing Infrastructure*. Morgan-Kaufmann, 1998.

[8] I. Foster, D. Kohr, Jr., R. Krishnaiyer, and J. Mogill. "Remote I/O: Fast Access to Distant Storage". *Proc. of the Fifth Annual Workshop on I/O in Parallel and Distri. Systems*, Nov. 1997, pp.14-25.

[9] G. Gibson, D. Nagle, K. Amiri, F. Chang, H. Gobioff, E. Riedel, D. Rochberg and J. Zelenka, "A Cost-effective, High-bandwidth Storage Architecture", *Proc. of the 8th Conf. on Architectural Support for Programming Langagues and Operating Systems*, 1998.

[10] J. H. Hartman, I. Murdock, and T. Spalink. "The Swarm Scalable Storage System." *Proceedings of the 19th IEEE International Conf. on Distributed Computing Systems (ICDCS '99)*, June 1999.

[11] J. H. Hartman, and J. K. Ousterhout, "The Zebra Striped Network File System", *ACM Transactions on Computer System*, Vol.13, No3, Aug. 1995, pp.274-310.

[12] J. H. Howard, et al., "Scale and Performance in a Distributed File System". *ACM Trans. on Computer System*, Vol.6, No.1, pp.51-81, February, 1988.

[13] H. I. Hsiao and D. DeWitt, "Chained Declustering: A New Availability Strategy for Multiprocessor Database Machines", *Proc. of 6th International Data Engineering Conf.*, 1990, pp.456-465.

[14] Y. Hu and Q. Yang, "DCD - Disk Caching Disk: A New Approach for Boosting I/O Performance", *Proc. of the 23rd International Symp. on Computer Architecture*, 1996, pp.169-177.

[15] K. Hwang, H. Jin, et al, "Reliable Cluster Computing with a New Checkpointing RAID-x Architecture", *Proc. of 9-th Heter. Comp. Workshop* (HCW-2000), Cancun, Mexico, May 1, 2000, pp.171-184.

[16] K. Hwang, H. Jin, E. Chow, C.L. Wang, and Z. Xu. "Designing SSI Clusters with Hierarchical Checkpointing and Single I/O Space". *IEEE Concurrency*, March 1999, pp.60-69.

[17] K. Hwang and Z. Xu. *Scalable Parallel Computing*, McGraw-Hill, New York, 1998.

[18] H. Jin and K. Hwang, "Striped Mirroring Disk Array", *Journal of Systems Architecture*, Elsevier Science, Vol.46, No.6, April, 2000, pp.543-550.

[19] E. K. Lee and C. A. Thekkath. "Petal: Distributed Virtual Disks". *Proc. of the Seventh Int'l Conf. on Arch. Support for Prog. Languages and Operating Systems*, October 1996, pp.84-92.

[20] W. B. Ligon and R. B. Ross. "An Overview of the Parallel Virtual File System". *Proceedings of the 1999 Extreme Linux Workshop*, June 1999.

[21] J. Nieplocha, I. Foster, H. Dachsel, "Distant I/O: One-Sided Access to Secondary Storage on Remote Processors", *Proc. of Symp. High Perf. Distri. Computing* (HPDC-7), 1998, pp.148-154.

[22] N. Nieuwejaar and D. Kotz. "Performance of the Galley Parallel File System". *Proceedings of the Fourth Workshop on I/O in Parallel and Distributed Systems*, pp.83-94, Philadelphia, May 1996.

[23] W. Ro and K. Hwang, "Striped and Staggered Checkpointing on Distributed RAID", Technical Report, University of Southern California.

[24] RAID Advisory Board, *The RAIDbook*, Seventh Edition, December, 1998.

[25] R. Sandberg, D. Goldberg, S. Kleiman, D. Walsh, and B. Lyon, "Design and Implementation of the Sun Network Filesystem", *Proc. of the USENIX Conference*, June 1985, pp.119-130.

[26] S. Savage and J. Wilkes, "AFRAID -- A Frequently Redundant Array of Independent Disks", *Proc. of 1996 USENIX Technical Conference*, January 1996, pp. 27-39.

[27] M. Stonebraker and G. A. Schloss, "Distributed RAID – a New Multiple Copy Algorithm", *Proc. of the Sixth International Conf. on Data Engineering*, Feb. 1990, pp.430-437.

[28] N. Talagala, S. Asami, D. Patterson, and K. Lutz, "Tertiary Disk: Large Scale Distributed Storage", *UCB Technical Report No. UCB//CSD-98-989*.

[29] C. A. Thekkath, T. Mann, and E. K. Lee. "Frangipani: A Scalable Distributed File System". *Proceedings of ACM Symp. of Operating Systems Principles*, Oct. 1997, pp.224-237.

[30] N. H. Vaidya, "A Case for Two-Level Distributed Recovery Schemes", *Proc. ACM International Conf. on Measurement and Modeling of Computer Systems* (Sigmetrics '95), pp.64-73.

[31] J. Wilkes, R. Golding, C. Staelin, and T. Sullivan, "The HP AutoRAID Hierarchical Storage System", *ACM Transactions on Computer Systems*, Vol.14, No.1, February 1996, pp.108-136.

[32] N. H. Vaidya, "Staggered Consistent Checkpointing", *IEEE Transactions on Parallel and Distributed Systems*, 1999, Vol. 10, No. 7, pp. 694- 702.