PERFORMANCE OF BALANCED DISK ARRAY SCHEMES

Walter A. Burkhard, Kimberly C. Claffy, Thomas J.E. Schwarz

University of California at San Diego
La Jolla, California

ABSTRACT

The Balanced Information Dispersal Algorithm (BIDA), a generalization of Rabin's Information Dispersal Algorithm (IDA) [1,2], provides attractive reliability enhancements as well as modest performance gains for the storage of data at a number of storage sites. A Balanced Disk Array (BDA) is an application of BIDA. Although a BDA appears as a single, very reliable, fast direct-access device, it actually consists of a number of independent disks that store data according to a selected dispersal scheme. BDA devices offer exceptionally high fault tolerance combined with better performance than their constituent disks. As a consequence, a BDA achieves high quality with relatively low-quality components. BDA schemes also offer the flexibility to arrange for a desired ratio between read and write times. This paper discusses the construction, failure tolerance, and performance of BDAs.

BALANCED INFORMATION DISPERSAL AND BALANCED DISK ARRAYS

Information dispersal refers to the encoding of a file's contents into N fragments such that any D of these fragments are sufficient to reconstruct the file. Although there are many possible methods of dispersal, the most attractive one is based on Reed Solomon codes. There are fast VLSI chips available that implement the dispersal and reconfiguration of information as a step in error-correcting decoding [3]. Preparata [4] proposes an alternative implementation.

We describe this IDA scheme in more detail. Encoding a file of a sequence of bytes entails multipliying D bytes of the file at a time by an $N \times D$ matrix A. We interpret the entries in A as numbers in the finite field of all possible bytes. Matrix A is chosen such that any D rows are linearly independent. The m^{th} coordinate of the resulting vector represent a single byte in the m^{th} fragment.

Retrieving the file requires the availability of D of these fragments. The first bytes of these fragments form a D-dimensional vector. We invert the submatrix of A consisting of the rows corresponding to the available fragments and multiply the vector with this inverse. This yields the first D bytes of the file. The reconfiguration continues with the remaining bytes in the fragments, one byte at a time.

The Balanced Information Dispersal Algorithm is a generalization and adaptation of IDA to distributed storage. Before storage, BIDA transforms a record into fragments with IDA and then stores the fragments at different sites. To decrease write operation times, BIDA actually stores only a *write quorum* W of fragments. BIDA uses *version numbers* to identify valid fragments, since some sites contain invalid data. For record retrival, BIDA accesses only a *read quorum* R of sites. It uses the first D valid fragments to obtain the original record. Parameters W and R are chosen to insure the existence of D valid fragments in a read quorum. BIDA allows a wide range of values for R and W, yielding a wide spectrum of read and write service times and degrees of fault tolerance.

Balanced disk arrays use a BIDA scheme to store data, storing file fragments on N separate disks. We store version numbers within each fragment and within the file access data. For a BDA write, only the first W disks reached are written; the other disk writes are aborted if another request is pending. A BDA read accesses all disks. The BDA uses the first D valid fragments retrieved from disk to restore the record; it aborts the other reads.

The storage capacity of a BDA is D/N of the combined capacity of the component disks. A smaller choice of W yields better write service times at the expense of increased read times and decreased data safety. It is possible to achieve both lower write and read service times than for a single disk. BDAs are meant to be built with inexpensive disks; the scheme more than compensates for the deficiency of its disk components. In addition to the disks, a BDA requires a buffer to store fragments before encoding as well as an encoding/decoding chip.

The BDA scheme is very versatile. We discuss two enhancements. Usually, read service times are more important for direct access devices. Thus, one would tend to choose a value of W relatively close to N. (For a reconfiguration quorum D=7, we obtain balanced read and write service times at W=12 with N=20.) With higher values of W, we can process up to two retrieval requests in parallel. This leads to improvements at higher loads. Another enhancement benefits read operations of frequently read files, such as the operating system kernel. After a BDA retrieves data, if it has no further pending requests; it does not simply abort after gathering enough fragments. Instead, the BDA disperses the record, after reconfiguration, and stores it at disks that previously only

CH3039-5/91/0000/0045/$01.00 © 1991 IEEE

contained an invalid fragment. This performance enhancement, we call it *adaptive write quorum*, incurs no cost.

FAULT TOLERANCE

A principal drawback to building storage systems of an ensemble of components is the significant degradation of the reliability of such systems compared with the reliability of a single component. The generous use of disk space in a BDA presents a cushion against the failure of single component disks. We use mean time to data loss (MTTDL) as the criterium of BDA reliability. A BDA with 20 disks, a write quorum W of 15, and a reconstruction quorum D of 7 can still fully operate with 8 of its disks failed. Without replacement of individual disks, this BDA has MTTDL of about half the mean time to failure (MTTF) of the component disks.

With replacement of components MTTDL increases considerably. This gain can be so extreme that one can essentially rule out individual disk failure as a cause for data loss; secondary causes, catastrophic events like fire and earthquake, or failure or destruction of the power supply and central components emerge as the most significant factor.

We have analyzed two different repair strategies. In the first, one inspects the BDA at regular intervals. If the number of failed disks surpasses a threshold during an inspection, then one replaces failed disks. Hence, in this scheme, many maintenance visits will be skipped. In the second repair strategy, the failure of more than a threshold number of disks triggers a repair request. In response to a repair request, the repair operation replaces all failed disks by new ones and rewrites the contents of the BDA.

MTTDL with Scheduled Inspections				
N	RT	Min	MTTDL (in years) quarterly inspection	MTTDL (in years) monthly inspection
20	19	18	0.7×10^{0}	2.5×10^{0}
20	19	17	1.6×10^{0}	1.6×10^{1}
20	19	16	4.4×10^{0}	1.4×10^{2}
20	19	15	1.9×10^{1}	1.6×10^{3}
20	18	15	1.7×10^{1}	4.4×10^{2}
20	19	13	3.6×10^{2}	3.5×10^{5}
20	17	13	1.4×10^{2}	1.1×10^{4}
20	19	11	1.8×10^{4}	1.8×10^{8}
20	17	11	5.0×10^{3}	3.4×10^{6}
20	19	9	1.9×10^{6}	1.9×10^{11}
20	17	9	3.8×10^{5}	2.6×10^{9}

MTTDL with Triggered Repair			
N	RT	Min	MTTDL (in years) One Week Time Lag
20	19	18	5.2×10^{1}
20	19	17	4.2×10^{2}
20	19	16	3.6×10^{3}
20	19	13	3.3×10^{6}
20	17	13	1.2×10^{5}
20	19	11	4.3×10^{8}
20	17	11	1.7×10^{7}
20	19	9	8.2×10^{10}
20	17	9	3.1×10^{9}

Table 1. Mean Time to Data Loss under the regular inspection (above) and the triggered repair regime for a BDA with "N" disks, which is repair when only "RT" disks are operational. The BDA needs Min disks to insure no loss of data. The component disks have MTTF of only 20000 hours (2.3245 years).

In our analysis, we concentrated on data loss due to individual disk failures. We used Markov chain models to obtain closed-form formulae for the MTTDL and the probability of a repair at a scheduled maintenance time. The (hypothetical) component disks have a very low MTTDL. We present some of our numerical results in Table 1. The derivations appear in the companion technical report [5].

Individual disks suffer from inperfections in the magnetic medium, preventing the use of some blocks. BDA blocks are usable if all but one or two of the disk blocks constituting it are free from bad spots. A write to a BDA block containing a bad disk block takes longer, because bad disk blocks can never partake in a write quorum; however, a read to such a BDA block is unaffected.

Soft and hard read errors occur at individual disks. A read that has failed due to a soft error is simply repeated after a full rotation of a disk (10-16.7 ms). A soft error occuring during an access in a BDA read incurs a much smaller penalty since the next available fragment replaces the unreadable one. Hard failures occur more frequently in a BDA since there are more disks. The inherent redundancy of the BDA architecture, however, makes their effects practically invisible.

PERFORMANCE

Response times of a direct access device (DAD) depend heavily on the character of the load; often temporal and spatial locality of access requests leads to better service and response times. However, modelling these various loads is difficult; we thus restricted ourselves instead to random locations and Poisson (i.e., unstructured random) arrivals. This yields conservative estimates of actual performance.

We developed formulae describing the service time for a disk modelled after the IBM 3380 J. We then developed service time formulae for the different BDA schemes, with and without external version number information. We also developed worst-case scenario service times for BDA devices with faulty disks. If access to less than half of the disks is needed, the scheme shows better average service times and a significantly lower variance in all cases. (Typical variances are 3 to 4 ms compared to 30 ms for a single disk.) This more deterministic behaviour of the BDA results in lower response times. This is valuable in certain applications such as on-line storage of massive visual information that demand predictable service times.

Figure 1 presents some of our results. We provide access times for a single disk on all graphs for comparison purposes. We give the response time for writes and reads for varying write quora W. The gradual change in performance with varying write quorum shows that a small number of faulty disks leads to minor performance degradation. If a BDA with 20 disks suffers loss of two disks, the worst-case read response degrades to the one for a BDA with write quorum W-2. The write response for this BDA equals that for a BDA with two fewer disks. Our graphs also contain read response times for parallel reads, in which case half of the disks in a BDA serve one read, while the other half serves a second read. These load numbers reflect the load of one half of the BDA.

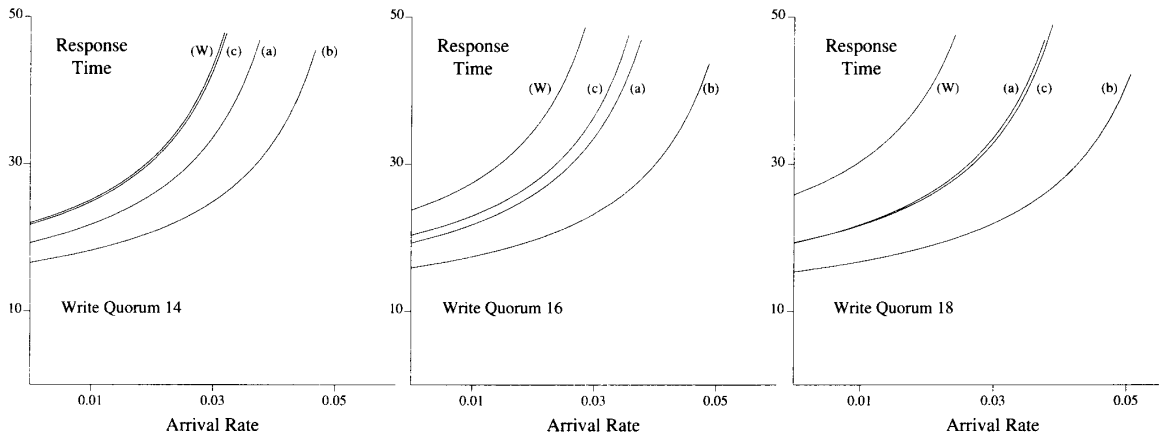

Figure 1: Performance of a BDA with 20 disks and reconfiguration quorum 5. We give the response time of a single disk access (a), a BDA read (b), one parallel BDA read (c), and of a BDA write (W).

Table 2 presents the results of simulations, during which a hypothetical disk was modelled. The results are insensitive to the exact behaviour of the disk. One can see that lower write quora result in lower write but higher read service times. Again, the results for a single disk provide a base for comparison.

Response Time Simulation for BDA (Parallel Reads)						
Number of Disks	Write Quorum	Recon. Quorum		Count	Expec-tation	Stand. Deviation
1			Write	1688	22.662914	9.806096
1			Read	3357	22.164433	9.459054
1			Write	5018	23.786966	11.030768
1			Read	10068	23.247120	10.664573
1			Write	100081	24.996933	12.842823
1			Read	199655	24.951401	12.774114
20	12	7	Write	1693	23.548140	5.478806
20	12	7	Read	3363	22.661612	5.489290
20	12	7	Write	5044	24.575336	7.211687
20	12	7	Read	10066	23.940790	7.517098
20	12	7	Write	10094	26.143948	9.610193
20	12	7	Read	19883	25.414124	9.528119
20	14	7	Write	1699	25.834021	5.774207
20	14	7	Read	3390	21.304720	5.521911
20	14	7	Write	5026	26.815161	7.123455
20	14	7	Read	10128	22.422985	7.310118
20	14	7	Write	10035	28.177578	9.240656
20	14	7	Read	19972	23.887743	9.481475
20	16	7	Write	1708	28.415106	5.902241
20	16	7	Read	3394	20.023277	5.494655
20	16	7	Write	5044	29.317209	7.413608
20	16	7	Read	10094	21.217060	7.412131
20	16	7	Write	10047	30.875088	10.140890
20	16	7	Read	19938	22.731216	9.643435
20	18	7	Write	1705	31.638123	6.501423
20	18	7	Read	3407	19.280891	6.045768

Table 2. Simulation Results of BDAs with parallel reads on demand. The simulation cold-starts and continues for 1000 seconds. We ordered the table from the conservative to the progressive layout.

Response Time Simulation for BDA (No Parallel Execution)						
20	10	5	Write	1635	21.159634	4.877164
20	10	5	Read	3277	20.653952	5.108156
20	10	5	Write	4954	22.035728	6.209320
20	10	5	Read	10009	21.695374	6.610047
20	10	5	Write	9948	23.314436	8.217979
20	10	5	Read	20284	23.013409	8.559411
20	9	5	Write	1627	20.142593	4.791084
20	9	5	Read	3281	21.843645	5.476469
20	9	5	Write	4931	21.097952	6.292918
20	9	5	Read	9989	22.802784	6.924872
20	9	5	Write	9959	22.536701	8.434684
20	9	5	Read	20236	24.120380	8.906157
20	8	5	Write	1635	19.257492	5.308617
20	8	5	Read	3281	22.937519	5.555192
20	8	5	Write	4958	20.166397	6.543950
20	8	5	Read	9947	23.981602	7.206654
20	8	5	Write	9965	21.753538	9.116640
20	8	5	Read	20193	25.524538	9.518342
20	6	3	Write	1632	16.680147	4.342917
20	6	3	Read	3260	19.951841	5.434194
20	6	3	Write	4996	17.541433	5.809189
20	6	3	Read	9997	20.705112	6.747925
20	6	3	Write	10006	18.580551	7.410272
20	6	3	Read	20211	21.663252	8.031772
20	6	2	Write	1636	15.187653	3.873252
20	6	2	Read	3266	17.395592	5.495386
20	6	2	Write	5039	15.930542	4.981722
20	6	2	Read	9986	17.934507	6.198398
20	6	2	Write	9957	16.648890	6.094142
20	6	2	Read	20163	18.650053	7.025160

Table 2. Simulation Results of BDAs with parallel reads on demand. The simulation cold-starts and continues for 1000 seconds. We ordered the table from the conservative to the progressive layout (cont.).

The performance of BDAs depends crucially on the randomization effects of accessing several disks. As the rotation of different disks is not synchronized, this randomization is given for the rotation phase. To insure that the seek times of different disks are independent, we use a different track numbering for each disk. Hashing of track numbers achieves this numbering. Several other heuristics achieve a similar randomization effect. One could return the heads to different (it home tracks) after every request as long as no other request is already waiting for service. The results of these schemes are more difficult to predict analytically.

RELATED WORK

BDA are similar to RAIDs [6]. In contrast to RAIDs, they offer write times similar to those of the individual disk components and suffer far less performance degradation due to failed disks BDA control units are less complex. BDAs use disk capacity much more extravagantly and offer less parallelization. As essentially serial devices, they allow easy commits of data base transactions. Their high fault tolerance allows one to build arrays of BDA modules.

BDAs are not the only application of the BIDA algorithm. At UCSD, Petar Stojadinovic is constructing a UNIX-based dispersal system that stores fragments of files on four different systems. He is currently evaluating the system run-time performance. The software implementation strives for fault tolerance.

Another application of BIDA is a variant of the BDA proposed here. Small write quora in combination with small reconfiguration quora lead to devices that process

very fast writes with reasonably large read times. In these DADs, parallel writes and adaptive write quora make the operation even faster. To achieve high levels of fault tolerance, we adopt a more elaborate maintenance strategy. When a disk fails, we read and rewrite the contents of the BDA, thus maintaining the write quorum as the minimum number of valid fragments.

REFERENCES

1. Rabin, M.O., "Efficient Dispersal of Information for Security, Load Balancing and Fault Tolerance," *Journal of the Association for Computing Machinery*, Volume 36, No. 2, April 1989, pp. 335-348.

2. _____, "The Information Dispersal Algorithm and its Applications," *Combinatorics, Compression, Security and Transmission*, Springer Verlag, 1990, pp. 406-419.

3. *Advanced Hardware Architectures*, Product Specification, Moscow, ID, 1988.

4. Preparata, F.P., "Holographic Dispersal and Recovery of Information," *IEEE Transactions on Information Theory*, Vol. 35, No. 5, 1989, pp. 1123-1124.

5. Burkhard, W.A., K. Claffy and T. Schwarz, "Performance of Balanced Disk Arrays," Technical Report, UCSD, 1991.

6. Patterson, D., G. Gibson and R. Katz, "A Case for Redundant Arrays of Inexpensive Disks (RAID)," Proceedings of the SIGMOD International Conference on Data Management, Chicago 1988, pp. 109-116.

7. Patterson, D., and J. Hennessy, *Computer Architecture, A Quantitative Approach*, Morgan Kaufmann Publ., San Mateo, CA 1990.

8. MacWilliams, F.J., and N.J.A. Sloane, *The Theory of Error-Correcting Codes*, North Holland Mathematical Library, North Holland, 1978.

9. von Neuman, J., "Probabilistic Logics and the Synthesis of Reliable Organizations for Unreliable Components," Automata Studies, Princeton University Press, 1958, pp. 43-98.

10. Ng, S., "Some Design Issues of Disk Arrays," *Proceedings of CompCon 1989*, pp. 118-123.

11. Schultz, M., G. Gibson, R. Katz and D. Patterson "How Reliable is a RAID?," *Proceedings of CompCon 1989*, pp. 118-123.