Providing Fault Tolerance In Parallel Secondary Storage Systems

Arvin Park

K. Balasubramanian

Department Computer Science
Princeton University
Princeton, New Jersey 08544

CS-TR-057-86

November 7, 1986

Abstract

Reliability is a critical concern for designers of parallel data storage systems. These systems consist of large numbers of storage devices which can provide high rates of data transfer. However, the consequential dependence on large numbers of devices can make such systems more prone to failure then non-parallel systems which depend on only a single storage device. This problem can be remedied by employing a modest amount of redundant storage to provide fault tolerance. In fact, providing fault tolerance in parallel data storage systems requires less redundancy (and is therefore more cost effective) than providing fault tolerance for non-parallel systems. This paper describes and analyses an effective method of providing fault tolerance in parallel data storage systems.

Keywords and Phrases - *Fault Tolerance, Mean Time Between Failures, Secondary Storage, Striping.*

1. Introduction

Parallelism has been the focus of much recent computer architecture research. Multiple CPU's have been used to increase computational speed and parallel memory banks have been employed to increase memory bandwidth. Recent research has extended this paradigm further down the memory hierarchy into parallel secondary storage systems [Kim85] [Sale86] [Arul82] [Poly86].

Reliability becomes a critical concern in such systems because of their dependence on large numbers of mechanical and electronic components. If naively designed, such systems can be substantially more prone to failure than conventional non-parallel secondary storage systems.

This paper examines reliability issues in parallel secondary storage systems. We use error correcting codes, and a clever regeneration procedure to provide fault tolerance for such systems. This method of providing fault tolerance is shown to be more efficient than conventional methods of providing fault tolerance in non-parallel systems.

2. Parallel Secondary Storage Architectures

Disk Striping has been proposed as an efficient method of exploiting parallelism in secondary storage systems [Kim85] [Sale86]. Such systems are typically composed of a set of N disk drives which operate in parallel to provide high rates of I/O bandwidth. Each logical disk block is partitioned into N equal sized sub-blocks, and each sub-block is stored on a different disk drive. This allows a single block access to utilize the full bandwidth of the N disk drives.

Two major types of errors can occur in such a system. Many failures are caused by spurious electronic, transmission, or mechanical errors. These errors cause no permanent loss of data and a subsequent request to the same memory block will, with high probability, succeed. We call these *spurious* errors. Other errors are caused by permanent hardware failures. In these situations, a functional subunit or the entire system is permanently disabled. Some data may consequently be lost. We call this more serious type of failure a *hard* error. We examine both types of errors in this paper.

3. Error Correcting Codes

Most disk systems presently utilize checksums to detect errors on disk accesses. If a block's checksum does not match the checksum calculated during a block read, then the data was not transmitted properly, and the read operation is performed again. This error detection method corrects errors that occur during a block access, but it can do nothing to mitigate the effects of a disk failure, when the whole block along with the checksum may be lost. To detect and correct for this type of error, something more is required.

We will employ error correcting codes to provide fault tolerance across a number of memory units. This is presently done in many semiconductor memory systems, but until recently hasn't been applied towards designs of secondary storage systems. Error correcting strategies change when applied to secondary storage. This is because each functional unit (disk drive) can provide operational status to the memory system. The system can then act upon the information provided. In other words, if a number of drives have failed in the system, we can query the disk drives find out which ones have

failed. Correction then becomes easier, since expensive error correcting codes are not required to detect error position.

If error position information is available, N bit error detection capability will suffice to correct N bit errors [Arul82]. (Merely test all 2^N possible combinations for the failed bits using the error detection capability, only the correct combination will produce no detectable error. This is an exhaustive method, but serves to prove our point. Faster techniques, which can be implemented in hardware, exist [Berl84].) As an example consider a single parity bit which suffices to detect a single bit error. This parity information along with error position information allows one to correct single bit errors.

In this paper we will use two types of error detection codes. Single error detection can be provided for by adding a single parity bit. This uses only one extra bit of information to detect a single bit error across N bits. Double bit error detection can be provided by a modified Hamming code. This method uses M redundant bits to detect double bit errors in $2^{M-1} - (M-1) - 1$ bits of data [Berl84]. This means O(log$N$) redundant bits of data are required to provide double error detection capability for N bits of data. Other methods of double error detection use fewer bits on particular word lengths, but no general method performs significantly better than the modified Hamming code.

4. Spurious Errors

This type of error will not cause catastrophic system failure or loss of data, but it can degrade system performance and complicate system software. To model such a system we assume it is composed of data striped bit-wise across N disk drives. (Note that the data may be striped by word or sub-block, but this doesn't affect our results. It is only essential that data from a single logical block be equally distributed across N disk drives. When Data is striped by sub-block we can still use standard error correcting codes by maintaining that the kth bit in the error-correcting drive sub-block corresponds to the error correcting bit for the word consisting of the kth bits of all the data sub-blocks.) We assume probability of failure access on a single request to a single disk drive is ρ, and that M failures can be tolerated on a given access. We can then calculate $\rho(N,M)$ = the probability of more than M failures on a single access.

$$\rho(N,M) = 1 - \left(\sum_{i=0}^{M} \rho^i (1-\rho)^{N-i} \binom{N}{i} \right)$$

To determine what this means for a a practical system we assume $N = 16$, and let ρ and M take on several values in Figure 1. As we can see from the graph, providing the ability to tolerate single bit errors will compensate for the increased failure rate of a parallel disk system. (This is true for practical values of ρ, $.0001 \leq \rho \leq .01$)

5. Hard Errors

To model hard errors we assume our system is composed of N disk drives each with an exponential failure rate with constant λ. the probability of failure of a given disk drive in a period Δt then becomes $f(t) = \lambda e^{-\lambda t} \Delta t$. By convolving failure rates of multiple disk drives we find the probability distribution for the first disk failure out of an N disk system is $f_N(t) = N\lambda e^{-N\lambda t} \Delta t$.

The metric that we are interested in is the MTBF (mean time between failures). Note that the MTBF of a disk drive with an exponential failure rate with constant λ is $1/\lambda$. From this we can derive an expression for the MTBF of an N disk system in which M failures can be tolerated.

$$MTBF(N,M) = \frac{1}{N\lambda} + \frac{1}{(N-1)\lambda} + \ldots + \frac{1}{(N-M)\lambda} \approx \frac{M+1}{N\lambda}, \quad M << N$$

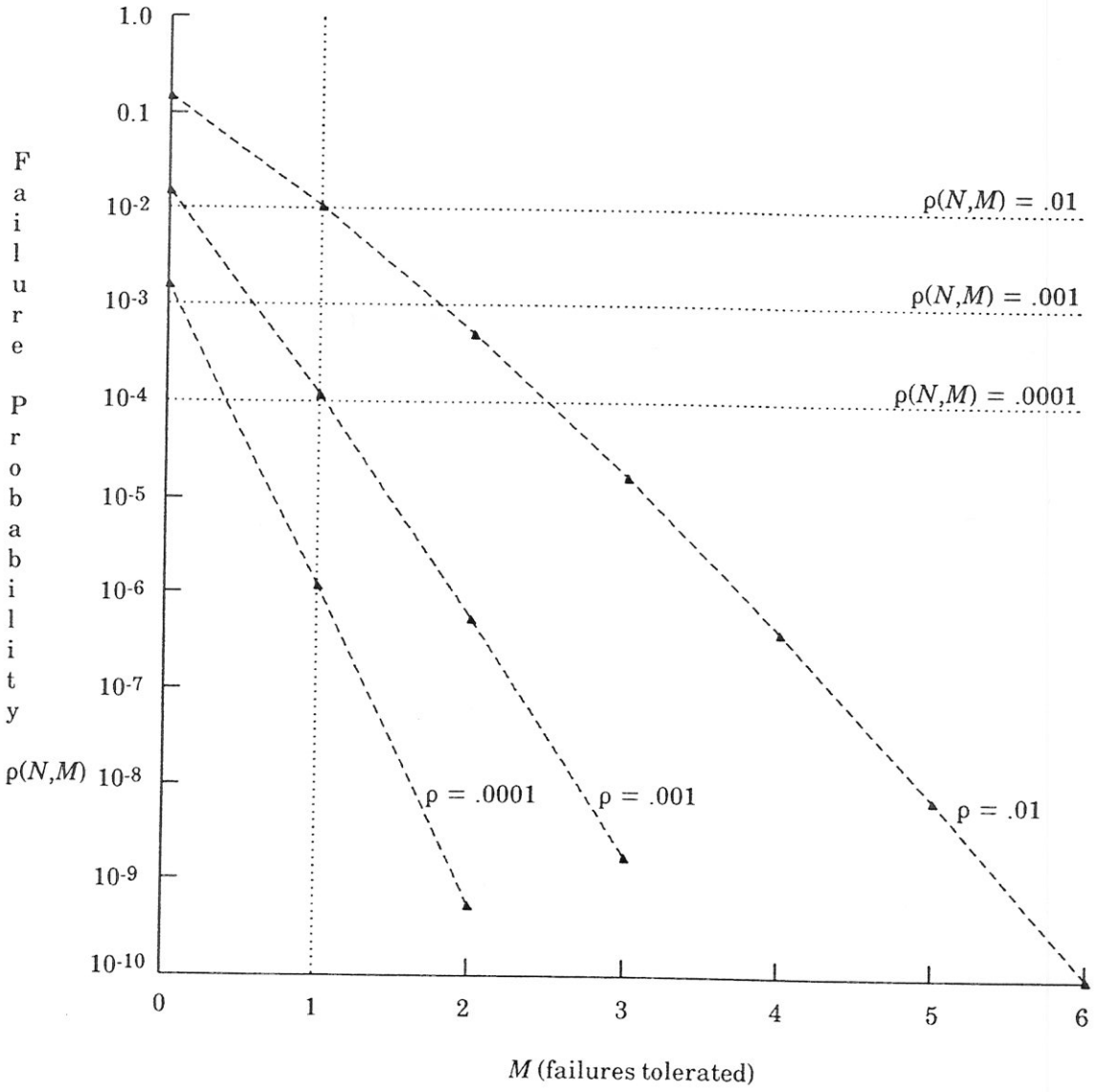

Figure 1: Failure Rate as a Function of Fault Tolerance Level $N=16$

Note that the MTBF increases almost linearly with the number of failures that can be tolerated. This is offset by an inverse linear decrease in MTBF due to multiple disk drives. Thus, to produce a system with an MTBF $\geq 1/\lambda$, we would have to be able to tolerate quite a few failures. This is clearly not practical as N increases. This problem can be solved by adding regeneration capabilities to the fault tolerant secondary storage system.

6. Regeneration

When a disk drive fails, a spare drive can be brought on line, and the original contents of the failed drive can be reconstructed and copied onto the spare drive. To do this it is necessary to scan the entire contents of the parallel disk bank to reconstruct the contents of the spare drive. This doesn't necessitate bringing down the entire system. A background process can scan through the disk drives performing the regeneration while the system continues to operate with a slightly reduced performance.

If a subsequent disks should fail during the regeneration process, and this failure can be tolerated, the regeneration process does not have to start over for the first failed drive. The scan can proceed through the disk bank, regenerating both failed drives. When the end of the disk bank is reached the first failed drive will be completely regenerated. The scan process can then proceed cyclically through the disk bank to complete regeneration of the second failed drive. The scan process can be thought of as continually scanning cyclically through the disk bank. To completely regenerate a disk, it is only necessary to wait for one complete cycle of the scan process.

The regeneration process adds significantly to the MTBF of the system. For the system to completely fail, more than M failures must occur within the time required to regenerate one drive (the first one that failed). This means more than M failures must occur in a time window of size R (where R is the regeneration time). To estimate the MTBF of a parallel secondary storage system with regeneration, we assume the system is composed of N disk drives, and can tolerate M failures. Failures for each disk drive are exponentially distributed with parameter λ and the regeneration time is R.

From this we calculate the probability that a single (given) disk drive fails in R time window.

$$P(\textit{single disk drive fails in } R \textit{ time window}) = \int_{t=0}^{t=R} \lambda e^{-\lambda t} \Delta t \; = 1 - e^{-\lambda R}$$

Call this quantity α. We can now approximate the probability that more than M disks fail in a time period R. (Note that R is in general $<< 1/\lambda$.)

$$P(\textit{more than } M \textit{ disks fail in a time period } R)$$

$$= 1 - P(M \textit{ or fewer disks fail in a time period } R)$$

$$= 1 - \Big(P(0 \textit{ disks fails}) + P(1 \textit{ disks fail}) + \ldots + P(M \textit{ disks fail}) \Big)]$$

$$= 1 - \left(\sum_{i=0}^{i=M} \binom{N}{i} (1-\alpha)^{N-i} \alpha^{i} \right)$$

To get an MTBF for the resulting system, we simply multiply the repair time by the expected number of repair times until first failure. The expected number of repair times is just the inverse of the probability of failure in a single repair period.

$$MTBF(N,M) = \frac{R}{1 - \left(\sum_{i=0}^{i=M} \binom{N}{i} (1-a)^{N-i} a^{i} \right)}$$

To produce numerical estimates for MTBF, assume the regeneration time R is two hours. Assume that $N = 16$ and that $\lambda = .0001$ (If failure rates are assumed exponential, this translates into a 10,000 hour MTBF for a single disk drive which is typical of present day industry standards). From these figures, we can compute the MTBF's for different values of fault-tolerance M. Different values of M require different amounts of numbers of error correction drives. A single redundant drive suffices to correct single bit errors so the total N for single bit fault tolerance becomes 17. Six redundant drives can be used to correct double bit errors on 16 data drives by using a modified Hamming Code. This means 22 drives are required to provide double bit fault-tolerance. Using these values of N we have calculated MTBF figures for systems which can tolerate zero, one, and two bit failures. These are listed in Table 1.

$M(N)$	0(16)	1(17)	2(22)
MTBF	626 hours	42.0 years	18,600 years

Table 1: MTBF As a Function of Fault Tolerance M

As can be seen, providing the ability to tolerate a single disk failure extends the system MTBF to 42 years. At this point, clearly other system components (such as control processors, and data paths, error detection and correction hardware, power supplies, software, human operators) will constrain fault tolerance more than individual disk failures [Gray85]. These observations lead to a viable architecture for a high bandwidth fault-tolerant secondary storage system.

7. Architecture

We propose an architecture for fault tolerance which makes use of error correcting codes and redundant disk drives to provide a high degree of fault tolerance and a large MTBF. Our system is composed of a set of disk which act as a single logical disk. The data is distributed across drives so that every logical block is evenly distributed across each of the drives. Each disk access can fully utilize the aggregate bandwidth of all of the drives.

There are three types of drives in our systems. "Data" drives contain data in bit-striped form . "ECC" drives contain error detection information which can be used too regenerate the contents of a failed drives. A small pool of "spare" drives is used to replace failed drives. The number of spare drives required is dependent on the availability of a service technician to replace failed drives. Enough spare drives should be provided so that the pool of spare drives cannot be depleted by the regeneration process before failed drives can be replaced. (Obviously we require the number of spares to be greater than or equal to the fault-tolerance M.)

Several architectural features are required to make this system feasible. "Live insertion" of disk drives must be possible so that the task of replacing failed drives can be completed without a system shutdown. This technology is presently available for printed circuit cards, and it doesn't appear complicated to construct such a system for small disk drives.

Synchronization of parallel drives has been examined in [Kim85]. It appears to be an easy task to synchronize the rotational motion of a set of disk drives by using a phase-locked loop [Kim85]. Synchronization improves average response time. A system of parallel disks would ordinarily have to wait for the slowest drive to perform its disk access before transmitting its composite logical block to the memory system. This average response time would almost always be the worst case response time of a single drive. If the disks are synchronized, the average response time for the entire system should be similar to the average response time of a single drive. Synchronizing disks also reduces buffering requirements to form a composite block from N individual disk blocks.

Our proposed architecture is illustrated in Figure 2. This system includes consists of 16 data drives, two spare drives and a redundant parity drive. This parity drive enables the system to tolerate a single disk failure. To increase reliability, we have provided redundant data paths to our control processor. A highly fault tolerant design is illustrated in Figure 3. This system cosists of 16 data drives, 4 spare drives, and 6 error correcting drives which enables the system to detect and correct two simultaneous single disk failures. Three control processors are provided, each with separate data paths to each of the disk drives. The three processors act as a single TMR (triple modular redundant) processor which is immune to single processor failure.

8. Comparison with Mirror Disks

We now compare our fault tolerant architecture with *mirror disk* systems which are presently used to provide fault tolerant secondary storage [Bart81] [Gray85]. A mirror disk system maintains two copies of the same data on separate but identical disks. If one disk should happen to fail, another copy remains intact on the second disk. The damaged copy may be restored by scanning the valid copy in much the same way as a bad disk is restored in the parallel architecture. The only way data can be lost is if both copies of the same data item are destroyed.

We contrast this with our fault tolerant parallel disk system which consists of a set of N striped disk drives, and a single parity drive that can be used to correct single disk errors. Note that both systems can tolerate single disk failures, but neither system can tolerate all possible double disk failures. The mirror disk system is also much less space efficient than the parallel disk systems. The mirror disk system doubles the amount of disk space required to store data, while the parallel system uses only one extra disk drive to provide fault tolerance for N disk drives.

To compare these systems it is not sufficient to merely mention the number of failures each system can tolerate. One must examine the relative probabilities of these failures. We model a mirror disk system as a collection of N pairs of disk drives ($2N$ total drives). Where each pair of drives contains two copies of the same data. We contrast this with a parallel disk system of $N+1$ disk drives. Where N drives contain data and 1 extra drive contains parity information for the entire set of N disk drives. Both systems have the same storage capacity, and both can tolerate single disk failures.

We assume the mirror disk, like the parallel system, can regenerate failed disk drives. And we assume this regeneration time R is identical for both mirror disk and parallel disk systems.

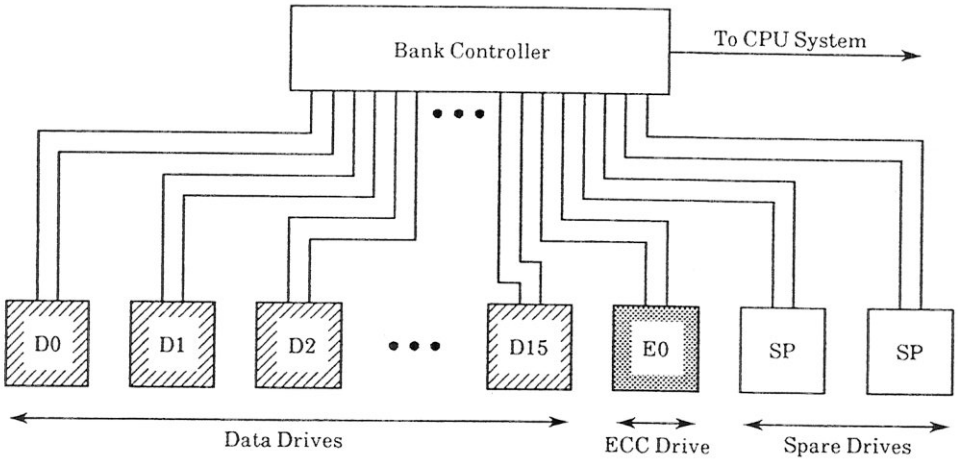

Figure 2: Fault Tolerant Secondary Storage Architecture

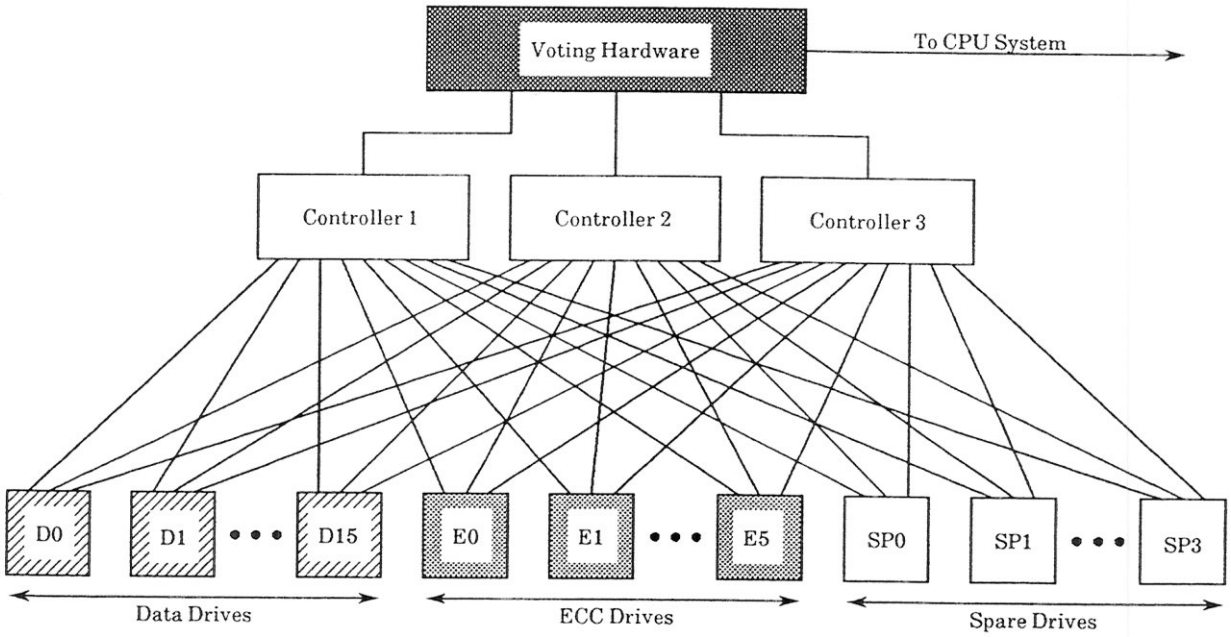

Figure 3: Highly Fault-Tolerant Secondary Storage Architecture

If both members of a mirror disk pair fail in the same repair period, data will be lost, and the mirror disk system will fail. This is differs from the parallel system where any two disk failures (out of the $N+1$ disks) will cause a system failure. We assume that all disk drives are identical, and that each disk fails with probability α in a repair time period R.

In a single repair time, the probability that at least one disk out of a mirror disk pair will not fail is:

$$P(\textit{mirror disk pair doesn't fail in time period } R) = P(\textit{no failures}) + P(1\ \textit{failure})$$

$$= (1-\alpha)^2 + 2(1-\alpha)\alpha$$

The probability that none of the N pairs fails in a single repair period is:

$$P(\textit{no pair fails}) = P(\textit{single pair doesn't fail})^N = \left((1-\alpha)^2 + 2(1-\alpha)\alpha\right)^N$$

$$= \left(1 - 2\alpha + \alpha^2 + 2\alpha - 2\alpha^2\right)^N = \left(1 - \alpha^2\right)^N$$

We next calculate the MTBF:

$$MTBF(\textit{for system of } N \textit{ mirror disk pairs}) = \frac{\textit{Repair Time}}{\left(P(\textit{failure in a single repair period})\right)}$$

$$= \frac{R}{\left(1 - P(\textit{no failures in a single repair period}\right)} = \frac{R}{\left(1 - \left(1 - \alpha^2\right)^N\right)}$$

To compare this with our previous analysis of parallel disks let R, N, and α assume their previous values. $R = 2$ hours, $N = 16$, $\alpha = .0002$. Using these values we find:

$$MTBF = 3.125 \; X \; 10^9 \; hours = 356.5 \; years$$

Note that this MTBF is higher than the single fault tolerant system, but lower than double fault-tolerant system. Since the double fault-tolerant system employs fewer redundant drives (O(log$N$) vs. O(N)), and produces a larger MTBF it appears to be a more cost effective method for providing fault tolerance in secondary storage systems. As we have noticed previously, the MTBF figures that we have arrived at were produced under the assumption that only disk drives can fail. This is clearly not the case in actual systems. As the probability of system failure due to disk failures diminishes, other factors will begin to constrain the MTBF. Power supplies, data paths, and processors will fail first [Gray85]. Even providing a single redundant parity drive makes disk failures an insignificant factor in system MTBF. This is clearly a more efficient, and cheaper, method of providing fault tolerance than blindly doubling the requisite storage.

Another issue to consider when comparing these two types of systems is system throughput. During read operations, a mirror disk system maintains two copies of the same data that can be read simultaneously. This doubles the system throughput for read operations only. The parallel disk system can perform only one block read or write operation at a time. However, these operations can proceed at the combined bandwidth of N parallel disk drives.

Conclusion

We have developed a efficient method of providing fault-tolerance in parallel secondary storage systems. This fault tolerance is provided without the great amount of redundant storage overhead presently utilized in non-parallel fault-tolerant secondary storage systems.

Our method of providing fault tolerance makes highly parallel secondary storage systems very attractive. Besides offering orders of magnitude more secondary storage bandwidth, the parallel

nature of these systems allows error correcting codes to be employed to efficiently provide a high degree of fault tolerance.

Acknowledgments

We would like to thank Ken Salem for his valuable comments and suggestions. This research was supported by the Defense Advanced Research Projects Agency of the Department of Defense and by the Office of Naval Research under Contracts Nos. N00014-85-C-0446 and N00014-85-K-0465, and by the National Science Foundation under Cooperative Agreement No. DCR-8420948. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

References

[Arul82] J. A. Arulpragasam, R. S. Swarz, "A Design For Process State Preservation on Storage Unit Failure", *Proceedings of the Tenth International Symposium on Fault Tolerant Computing*, October 1980, Kyoto, Japan, pp. 47-52.

[Bart81] J. F. Bartlett, "A NonStop Kernel", *Proceedings of the Eighth Symposium on Operating System Principles*, Vol. 15, No. 5, December 1981. pp. 22-29.

[Berl84] E. R. Berlekamp, "Algebraic Coding Theory", Aegean Park Press, 1984, pp. 1-20.

[Gray85] J. Gray, "Why Do Computers Stop and What Can Be Done About It?", Tandem Technical Report 85.7, June 1985.

[Kim85] Kim, M. Y., "Parallel Operation of Magnetic Disk Storage Devices: Synchronized Disk Interleaving", *Proceedings of the Fourth International Workshop on Database Machines*, March, 1985, pp. 299-329.

[Poly86] Polymorhphic Systems Corporation, Marketing Literature.

[Sale86] K. Salem, H. Garcia-Molina, "Disk Striping", Unpublished Report, Department of Computer Science, Princeton University, 1986