

\hat{B} : Disk Array Data Layout Tolerating Multiple Failures

Barbara T. Theodorides Walter A. Burkhard

Gemini Storage Systems Laboratory

Department of Computer Science and Engineering

University of California, San Diego

La Jolla, CA 92093-0114, USA

btheodor@cs.ucsd.edu burkhard@cs.ucsd.edu

Abstract

We present \hat{B} a novel data layout method for tolerating multiple disk failures within disk arrays. In a disk array with $2n$ disks, \hat{B} tolerates at most $2(n-1)$ simultaneous failures; reconstruction work is spread over the surviving disks using only exclusive-or operations. The data layout is based upon \hat{B} array-codes; our approach provides an efficient software implementation. \hat{B} utilizes the minimal amount of redundant storage space. Our detailed performance comparison with RAID-5 and EVENODD shows \hat{B} read operations to be very competitive especially in the presence of failures; \hat{B} write operations are more expensive than RAID-5 and EVENODD write operations. In the presence of failures, the performance gradually degrades as the number of failures increases.

keywords: archival storage systems, data layouts, performance, algorithms.

text word count: 3500

contact author: Walter Burkhard
 Gemini Storage Systems Laboratory
 Department of Computer Science and Engineering 0114
 University of California, San Diego
 La Jolla, California 92093-0114

This material has been cleared at UCSD.

category: Regular Paper.

1 Introduction

Disk arrays provide significant advantages over single disks [11]. Utilizing an array of disks to provide storage space achieves improved runtime performance via the aggregate bandwidth as well as shorter seek latencies associated with smaller disk drives. The mean time to data loss (MTTDL) of a disk array can be improved via inclusion of redundant information with client data.

Many existing data layout schemes incorporate RAID-5 [8] redundancy which utilize parity values capable of providing tolerance against a single disk failure. Initial attempts to provide additional fault tolerance utilized the information dispersal algorithm [12]; this scheme is also well known within the algebraic coding community. In practice, the redundancy calculations for such schemes often involve use of finite field multiplication operations. More recently, the EVENODD data layout [3] provides multiple fault tolerance in which all redundancy calculations utilize only exclusive-or (XOR) operations.

There are many reasons to consider data layouts capable of tolerating multiple simultaneous disk failures. Even though modern disk drives individually provide 1000000 hour MTTF, the use of hundreds or even thousands of drives together for a data set will give rise to very frequent disk failure [17]. Video and audio applications will be placing strong demands on storage systems which must deliver data at regular intervals; existing analysis shows increasing the number of redundant stripe units per group provides a more cost-effective approach to improving the MTTDL compared to increasing the number of groups or relying on manufacturer's likely improvements to individual disk MTTF [4]. Storage systems are known to be more likely to fail as the workload increases. Disks can exhibit latent stripe unit failures which are detected only when trying to access the stripe unit(s); data will be irretrievably lost in RAID level 5 redundancy schemes if the failed stripe unit is detected when reconstructing data for a crashed disk. Certain safety-critical applications require very large MTTDL which is not obtained by tolerating only a single failure.

Section 2 presents our architectural configuration as well as our failure assumptions. In section 3, we describe the basics of the \hat{B} data layout via graph theoretic factorization notions. Section 4 presents our evaluation of \hat{B} for diverse workloads and failure assumptions; the evaluation is derived from simulation results. Related work is described within Section 5 and we end with a conclusion.

2 System Model

A disk array contains $2n$ identical disk drives. Each drive has an independent controller allowing disks to service independent requests in parallel. Each controller provides the abstraction of a linear address space of sector addresses with the disk it controls, thereby hiding bad media portions as well as the position of each sector in terms of surface, cylinder and track position. For simplicity, we assume that each disk drive has an identical number of usable sectors.

There is also an *array controller* that translates user requests into individual operations on the disks of the array, supervises their completion, and dynamically reconfigures the array in the presence of failures. The array controller provides the client linear address space to the disk clients. We assume the array controller can identify failed disk(s). The array controller maintains redundant data values during write operations.

Disk space is logically structured as *stripe units* where each stripe unit consists of a fixed number of sectors. Each stripe unit will contain either client data or redundant data. The client requests that a number of stripe units be accessed to be read or written. All the redundant data is invisible to the client; only the client linear address space is visible.

3 \hat{B} Data Layout

The \hat{B} data layout is based upon the \hat{B} array codes of Xu et al. [18]. The \hat{B} array code is a two dimensional structure containing $2n$ columns and n rows; each code symbol designates a stripe unit. The array code is presented via the graph theoretic notion of perfect one-factorizations.

Suppose $G = (V, E)$ is a graph. A factor of G is a spanning subgraph of G and a one-factor of G is a one-regular spanning subgraph of G . A factorization of G is a set of factors of G which are pairwise edge disjoint – no two have a common edge – whose union is G .

A factorization of G consisting of only one-factors denoted $\{F_0, F_1, \dots, F_{k-1}\}$ is a *perfect one-factorization* (P1F) if for any distinct pair F_i, F_j of factors, $F_i \cup F_j$ induces a Hamiltonian cycle within G for $0 \leq i, j < k$. Figure 1 presents a K_6 perfect one-factorization.

The \hat{B} data layout is based upon a perfect one-factorization of a suitably sized complete graph. Perfect one-factorizations are of interest to graph theorists [15, 16] as well. Since P1Fs exist only for graphs with an even number of vertices and vertices correspond to disks, the data layout utilizes an even number of disks. Graph theorists have conjectured over over the past 40 years that every complete graph with an even number of vertices has a P1F.

When p is an odd prime, the complete graphs K_{2p} and K_{p+1} have perfect one-factorizations [10].

Here is a perfect one-factorization of K_{2p} given in [16]. Let $V = \{0, 1, 2, \dots, 2p - 1\}$ denote the vertex set and define the one-regular spanning subgraphs G_s for $0 \leq s < 2p$ with $s \neq p$

$$G_s = \begin{cases} \{(i, j) \mid i \not\equiv j, i + j \equiv s \pmod{2p}\} \cup \{(s/2, s/2 + p)\} & \text{if } s \text{ is even} \\ \{(i, j) \mid i \text{ odd}, i - j \equiv s \pmod{2p}\} & \text{otherwise} \end{cases}$$

As an example, figure 1 contains the G_s factors for $p = 3$.

Xu et al. modify the P1F factors to obtain the \hat{B} data layout. Let $2n = 2p - 2$ and then define the subsets F_i via two steps. First rename the factors so for $1 \leq i \leq 2n$ F_i is the factor containing the edge $(0, i)$; these renamed factors are referred to as *complete*. Then from each F_i factor delete two edges – one incident on vertex 0 and the other incident on $2p - 1$. Figure 1 also contains the final F_i modified factors for $p = 3$.

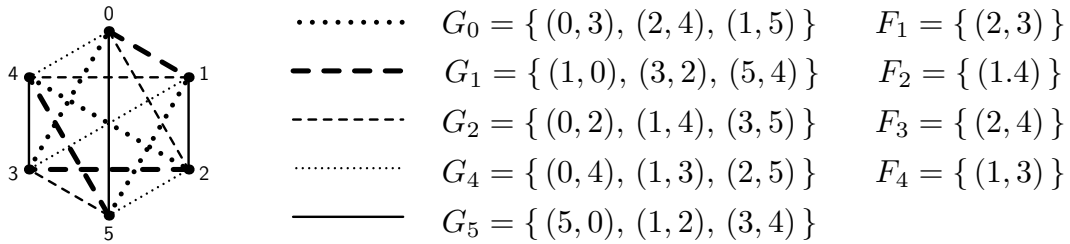


Figure 1: K_6 with G_s factors and associated F_i modified factors.

A perfect one-factorization of K_{p+1} is also given in [16]. Let $V = \{0, 1, 2, \dots, p - 1, \infty\}$ denote the vertex set. For $p > 3$ the one-regular spanning subgraphs P_s for $0 \leq s \leq p - 1$ with $n = (p + 1)/2$ are defined as

$$P_s = \{(\infty, s), (s - 1, s + 1), (s - 2, s + 2), \dots, (s - n + 1, s + n - 1)\}$$

with mod p arithmetic. Figure 2 contains the P_s factors for $p = 7$.

Modification of the P1F subsets proceeds as above. First the factors are renamed F_i for $1 \leq i \leq p - 1$ so F_i is the factor containing the edge $(0, i)$. Each F_i factor loses two edges – one incident on vertex 0 and one on vertex ∞ . Figure 2 contains the F_i factors for $p = 7$.

Perfect one-factorizations for other complete graphs would be modified in a similar fashion. Wallis [16] presents perfect one-factorizations for all even sized complete graphs from size four through 50 and many more sizes through 98; these sizes are very appropriate for \hat{B} layouts. However the existence of very large complete graph perfect one-factorizations is mentioned as well.

$$\begin{aligned}
P_0 &= \{(\infty, 0), (1, 6), (2, 5), (3, 4)\} & F_1 &= \{(2, 6), (3, 5)\} \\
P_1 &= \{(\infty, 1), (0, 2), (3, 6), (4, 5)\} & F_2 &= \{(3, 6), (4, 5)\} \\
P_2 &= \{(\infty, 2), (1, 3), (0, 4), (5, 6)\} & F_3 &= \{(1, 2), (4, 6)\} \\
P_3 &= \{(\infty, 3), (2, 4), (1, 5), (0, 6)\} & F_4 &= \{(1, 3), (5, 6)\} \\
P_4 &= \{(\infty, 4), (3, 5), (2, 6), (1, 0)\} & F_5 &= \{(1, 4), (2, 3)\} \\
P_5 &= \{(\infty, 5), (4, 6), (3, 0), (2, 1)\} & F_6 &= \{(1, 5), (2, 4)\} \\
P_6 &= \{(\infty, 6), (5, 0), (4, 1), (3, 2)\} & &
\end{aligned}$$

Figure 2: P_s factors and F_i modified factors: $p = 7$.

disk 1	disk 2	disk 3	disk 4	disk 1	disk 2	disk 3	disk 4	disk 5	disk 6
d_1	d_2	d_3	d_4	d_1	d_2	d_3	d_4	d_5	d_6
$d_2 \oplus d_3$	$d_1 \oplus d_4$	$d_2 \oplus d_4$	$d_1 \oplus d_3$	$d_2 \oplus d_6$	$d_3 \oplus d_6$	$d_1 \oplus d_2$	$d_1 \oplus d_3$	$d_1 \oplus d_4$	$d_1 \oplus d_5$
				$d_3 \oplus d_5$	$d_4 \oplus d_5$	$d_4 \oplus d_6$	$d_5 \oplus d_6$	$d_2 \oplus d_3$	$d_2 \oplus d_4$

Figure 3: \hat{B}_4 and \hat{B}_6 Data Layouts

The $2n$ vertices designate disk drives within a single reliability group; each edge within modified factor F_i specifies a parity value to be stored on disk drive i . The \hat{B}_{2n} data layout contains $2n$ disks; each disk i contains one user stripe unit together with $n - 1$ parity stripe unit values specified by F_i for $1 \leq i \leq 2n$. The \hat{B}_4 and \hat{B}_6 data layouts are presented within Figure 3.

Within this paper, our performance experiments will utilize single groups of eight disks using \hat{B}_8 derived from K_{10} . The \hat{B}_8 data layout is presented within Figure 4.

disk 1	disk 2	disk 3	disk 4	disk 5	disk 6	disk 7	disk 8
d_1	d_2	d_3	d_4	d_5	d_6	d_7	d_8
$d_2 \oplus d_3$	$d_1 \oplus d_6$	$d_1 \oplus d_8$	$d_1 \oplus d_3$	$d_2 \oplus d_8$	$d_1 \oplus d_5$	$d_1 \oplus d_4$	$d_1 \oplus d_7$
$d_4 \oplus d_5$	$d_4 \oplus d_8$	$d_2 \oplus d_5$	$d_2 \oplus d_7$	$d_3 \oplus d_7$	$d_2 \oplus d_4$	$d_3 \oplus d_6$	$d_2 \oplus d_6$
$d_6 \oplus d_7$	$d_5 \oplus d_7$	$d_4 \oplus d_7$	$d_6 \oplus d_8$	$d_4 \oplus d_6$	$d_3 \oplus d_8$	$d_5 \oplus d_8$	$d_3 \oplus d_5$

Figure 4: \hat{B}_8 Data Layout

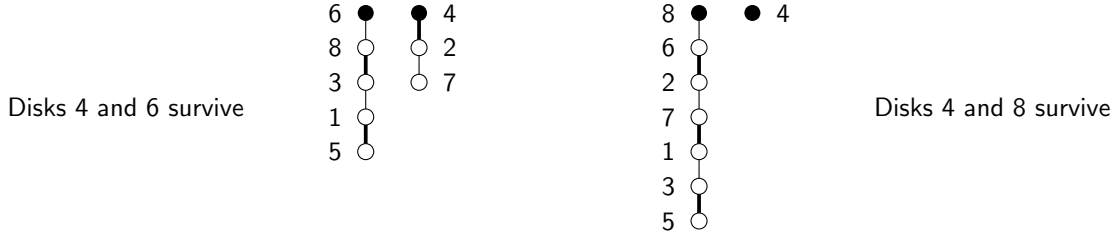


Figure 5: \hat{B}_8 reconstruction fragments for disks 4 and 6 as well as disks 4 and 8.

3.1 \hat{B} Structural Properties

A perfect one-factorization covers K_{2m} ; each factor contains m edges and a P1F must contain $2m - 1$ factors. Xu et al. modify the P1F one-factors as well as discarding a one-factor. Each of the remaining one-factors loses two edges as mentioned previously; for $1 \leq i \leq 2m - 2$, the F_i subsets contain $m - 2$ edges and F_i does not contain an edge incident on vertex i . Accordingly F_i describes the contents of disk i within \hat{B}_{2m-2} . Each F_i edge designates a parity value to be stored on disk i ; there are $m - 2$ parity values together with one client data value stored on disk i . Each disk will contain parity values involving client data on $2m - 4$ other disks within the group. Let $2n = 2m - 2$; the \hat{B}_{2n} data layout is defined in terms of the $2n$ F subsets.

The \hat{B} data layout allows reconstruction of missing client data when up to $2n - 2$ disks have failed. A *reconstruction sequence* is an ordering of surviving disk reads sufficient to reconstruct the missing client data. Suppose disks i and j survive; the two *complete* factors F_i and F_j form a Hamiltonian cycle through $2n + 2$ vertices. With four edges incident on vertices 0 and $2n + 1$ (or ∞) deleted, the Hamiltonian cycle is broken into two fragments that reach all failed disks as well as the surviving disks. Since the edge (i, j) is not present on either disk i or j , the fragments present a reconstruction sequence; moreover, a surviving disk will always be at the beginning of a fragment. Figure 5 presents two varieties of reconstruction fragments, one for only disks 4 and 6 surviving and the other for disks 4 and 8 surviving. The light edges designate disk 4 as the source of parity value while bold designates either disk 6 or 8.

When more than two disks survive, there are often several reconstruction sequences possible; these reconstruction sequences can have differing lengths. For example with disks 2 and 6 failing, here are three reconstruction sequences: 8, 4, 5, 3; 7, 4, 1; and 7, 4, 8. Our reconstruction algorithm creates several reconstruction sequence candidates; one of the shortest among the candidates is utilized to accomplish the reconstruction.

The \hat{B}_{2n} data layout requires the least amount of additional storage space to ensure

reconstruction for up to $2n - 2$ disk failures because the associated error correcting codes are maximal distance separable (MDS). The number of redundant bits that are modified by the change of a single client data bit is minimal [18].

4 Performance

We provide performance comparison results created by running RAID-5, EVENODD and \hat{B} under a highly-concurrent workload of read or write operations. We consider small operations (single stripe unit) as well as larger operations. We are able to compare performance for a wide variety of failure scenarios from fault-free up to $n - 2$ failures within a group of n disks for \hat{B} as well as fewer failures for RAID-5 and EVENODD. The experiments have been run on Raidframe [5] a tool allowing various RAID architectures to be evaluated. The Raidframe synthetic workload generator creates concurrent sequences of requests for disk accesses via multiple client threads. Within Raidframe, any RAID architecture converts a client request into a set of read and write accesses on specific disks of the array. Individual disk requests are serviced by an event driven simulator which is calibrated for several real disk models.

The Raidframe simulator has been expanded to include the \hat{B} RAID architecture; the data layouts presented in section 3 are implemented. The Raidframe system has been extended to accommodate more than two failed disks. The reconstruction scenario mentioned previously has been implemented. The modifications consist of approximately 2000 additional lines of C code. In addition, operational characteristics of the IBM18LZX disk drive have been included in the disk drive parameters file.

For our experiments, the array parameters are configured as follows. The individual disks are the IBM18LZX each with approximately 18 GB storage space, an average seek time of 4.9 milliseconds, and average rotational latency of 3 milliseconds. These drives are partitioned into 15 zones with between 195 to 382 sectors per track. Individual disks utilize the shortest seek time first queuing policy. The synthetic workload consists of uniformly distributed random independent stripe unit accesses from the client's address space; all accesses are stripe unit aligned. These small accesses, especially writes, are typical of demanding workloads. The Raidframe simulator and similar workloads have been utilized in several studies of various data layouts [1, 2, 5, 7, 14].

Our Raidframe experiments center on \hat{B} , EVENODD and RAID-5 configurations in which both \hat{B} and RAID level 5 are configured with eight disks per reliability group. EVENODD has $p + 2$ disks for prime p ; we present comparative results for seven and nine disk reliability groups. The reliability groups are composed of sixteen sector (8192 byte) stripe units. All

experiments are run to obtain a 95% confidence interval having width 1% of the to-be-determined average response time. Since both the sample average and sample standard deviation are unknown at the beginning of an experiment, the number of samples must be determined dynamically. The sample standard deviation varies widely depending upon the RAID configuration; we utilize the two-sample approach of Stein [13] to determine the number of samples sufficing to obtain confidence intervals of the desired width. All the graphical presentations show the confidence intervals as well as the response times.

4.1 All disks operational: reads

For single stripe unit reads, the small read operation, the \hat{B} and RAID-5 configurations obtain very similar response times in the fault free mode as shown in Figure 6a. As would be expected, the seven disk and nine disk EVENODD configurations have slightly larger and smaller response times than the eight disk \hat{B} and RAID-5 configurations. For 32KB and 96KB read operations, the results are much the same as seen in Figures 6e and 7c. \hat{B} and RAID-5 show slightly differing response times for 96KB reads since these configurations have different data layouts.

4.2 Single disk failure: reads

For single stripe unit reads, all surviving disks within the RAID-5 and all but one within the EVENODD configurations must provide reconstruction data as well as maintaining their original read activities; this doubles the workload for RAID-5 and almost doubles it for EVENODD. However, only two of the surviving disks within the \hat{B} scheme must read data to reconstruct the desired data. Within the \hat{B}_{2n} configuration, there are $2n - 2$ pairs of disks sufficient to reconstruct the desired stripe unit. The average workload increases by a factor of $(n + 1)/(n - 1)$. Figure 6b shows the superior response time of the \hat{B} configuration for one stripe unit reads. EVENODD-7 designates the EVENODD configuration containing seven disks; EVENODD-9 denotes the nine disk configuration. EVENODD-9 has the second best response time, then RAID-5 and EVENODD-7 the worst response time. Reading more than a single stripe unit will generally reduce the workload increase as the additional stripe units would have to be read in many cases. For four stripe unit read operations in Figure 6f, the \hat{B} configuration maintains its excellent behavior with the best response time, EVENODD-9 is again second best, then the EVENODD-7 response time and finally RAID-5 is worst. For twelve stripe unit reads, EVENODD-9 has slightly better response time than \hat{B} , then the RAID-5 response time and finally EVENODD-7 is worst as shown in Figure 7d.

4.3 Two disk failures: reads

For two disk failures, of course, RAID-5 cannot reconstruct the missing data. For both EVENODD and \hat{B} , reconstruction of a single stripe follows a process similar to the for a single failure. For EVENODD, the algorithmic reconstruction scheme suffices. For two failures, from three to four stripe units must be read for each reconstructed stripe. For our single stripe unit, four stripe unit and twelve stripe unit read operations, the probability a failed disk is encountered is double that for a single disk failure. The \hat{B} response times are smaller than those for EVENODD-7 or EVENODD-9; at most half the disks will be involved in the reconstruction process for \hat{B}_8 while all disks could be involved within either EVENODD. For one stripe unit, four stripe unit and twelve stripe unit reads with two failures, \hat{B} has the best response time as shown in Figures 6c, 7a and 7e.

4.4 Three or more disk failures: reads

For three or more disk failures, only the \hat{B} data layout suffices for reconstruction. Our discussion generally proceeds as for two disk failures. The response times increase rapidly with workload since the incremental reconstruction workload is larger and it is distributed over fewer surviving disks. For three failures, from four up-to six stripe units must be read for each reconstructed stripe. In \hat{B}_8 , with only five surviving disks, the average workload increases by a factor ranging from $9/5$ to $11/5$. The experimental response times are shown in Figures 6d, 7b, and 7f; the response times gracefully degrade with increasing numbers of simultaneous disk failures. Reconstruction sequences can involve single disks providing more than a single stripe unit especially for greater numbers of disk failures; this greatly increases the service and response times.

4.5 All disks operational: writes

Completion of any write operation requires that all instances of a value be updated for the client data as well as its associated parity values. The *update cost* for a write operation counts the number of sites the operation will modify when a stripe unit is modified. The RAID, EVENODD and \hat{B} data layout schemes have differing update costs. The number of disks modified per stripe may be less than the update cost in the presence of failed disks.

For single stripe unit writes, the so-called small write operation, the \hat{B} configuration update cost is $2n - 1$, for RAID-5 the update cost is two, and for EVENODD the update cost is at least three with the average greater than three. Figure 8a shows the response times in this situation. For four stripe unit and twelve stripe unit writes the operations

have similarly relative performance with RAID-5 the best as shown in figures 8e and 9d. The RAID-5 response time is approximately one-half that of \hat{B} for a workload greater than double that of \hat{B} .

4.6 Single disk failure: writes

For single stripe unit, four stripe unit and twelve stripe unit writes, the results are presented in figures 8b, 8f and 9d. Generally RAID-5 provides the best runtime performance with EVENODD-9, EVENODD-7 and then \hat{B}_8 providing the second, third, and fourth best performances. EVENODD-9 and RAID-5 response times become closer for the larger writes; the RAID-5 implementation can entail reading the unmodified stripe units of a stripe if the write operation requires updating at least one-half the stripe. Within EVENODD-9, the larger writes are more likely to incur a large uptime cost. likely to

4.7 Two disk failures: writes

Only the EVENODD and \hat{B} data layouts accommodate two disk failures. For both a single stripe unit and four stripe unit reads, EVENODD-7 and EVENODD-9 provide better response times than does \hat{B}_8 as shown in figures 8c and 9a. However, for twelve stripe unit writes, the \hat{B}_8 response time is considerably better than that of either EVENODD configuration as shown in figure 9e. The twelve stripe unit write involves at least one complete data row within either EVENODD configuration and the cost of maintaining the common diagonal S in which $n - 3$ stripe units, within an n disk EVENODD configuration, are updated on a single disk, have a prominent effect.

4.8 Three or more disk failures: writes

The single stripe unit write response times for to six failures is presented in figure 8d. For one or two failures, the reponse time is better than for the no failure case. The response time curves for three and four failures are almost the same but not as good as the no failure case. For six failures, with only two surviving disks, often several stripe units must be read to reconstruct the previous client data; in this situation, the last disk to read data must then write a new value.

For four or twelve stripe unit write operations, the results are similar but there will be more disks which must read and then write. The pair of operations will occur more than once on a disk as the number of failures increases. For six failures, many of the stripe units on one of the two surviving disks will be read followed by writing the modified value. The

response times increase accordingly over that for a single stripe unit write. Figures 9b and 9f present these response times. The four and twelve stripe unit write operation for either no failure or a single failure are almost the same.

5 Related Work

There are two varieties of related studies: information dispersal and coding methods. Information dispersal [9, 12] utilizes complex finite field operations to create independent redundant values. The scheme, which has been utilized within DATUM [1], provides much choice of design parameters and provides for minimal update cost. However finite field multiplication operations must be utilized in all instances beyond RAID-5 and these operations are not to be found within commodity processor instruction sets.

Coding methods for tolerating multiple failures have been proposed by Gibson et al. [6] as well as Blaum et al. [3]; both of these schemes use only exclusive-or operations to create redundant values. Gibson et al. present double-erasure and triple-erasure reconstruction encodings with minimal update cost; however their schemes do not readily generalize to greater numbers of erasures. Blaum et al. present the EVENODD encoding capable of double-erasure reconstruction; the scheme can be extended to accommodate additional erasures. In spite of possessing the MDS property, the average EVENODD update cost is not minimal.

6 Conclusions

We have presented \hat{B} an architecture for redundant disk arrays that can be configured to tolerate an arbitrary number of simultaneous disk failures. With the trend dictating large numbers of small disks, there is a pervasive need for more reliability than schemes tolerating a single disk failure or even double disk failures. The \hat{B} data layout depends upon graph theoretic notions of perfect one-factorizations; \hat{B} is the only existing method requiring an optimal additional amount of space while tolerating an fixed but arbitrary number of simultaneous disk failures.

We have compared the performance of \hat{B} data layouts with both RAID-5 and EVENODD and found it to perform as well as either for read operations in the fault free mode and much better for read operations in the common failure mode scenarios either other data layout can tolerate. The \hat{B} performance for write operations in the fault free mode as well as common failure mode scenarios is not as good as either RAID-5 or EVENODD; however it degrades

gracefully as the number of failures increases beyond what either RAID-5 or EVENODD can tolerate;

References

- [1] G.A. Alvarez, W.A. Burkhard, and F. Cristian. Tolerating multiple failures in RAID architectures with optimal storage and uniform declustering. *Proc. of the 24th International Symposium on Computer Architecture*, pages 62–72, 1997.
- [2] G.A. Alvarez, W.A. Burkhard, L.J. Stockmeyer, and F. Cristian. Declustered disk array architectures with optimal and near-optimal parallelism. *Proc. of the 25th International Symposium on Computer Architecture*, pages 109–120, 1998.
- [3] M. Blaum, J. Brady, J. Bruck, and J. Menon. EVENODD: An efficient scheme for tolerating double disk failures in RAID architectures. *Proc. of the 21st International Symposium on Computer Architecture*, pages 245–254, 1994.
- [4] W.A. Burkhard and J. Menon. Disk array storage system reliability. *Proc. of the International Symposium on Fault-tolerant Computing*, pages 432–441, 1993.
- [5] W.V. Courtright II, G.A. Gibson, M. Holland, L.N. Reilly, and J. Zelenka. *RAIDframe: A Rapid Prototyping Tool for RAID Systems*. Computer Science: Carnegie Mellon University, 1996.
- [6] G. Gibson, L. Hellerstein, R. Karp, R. Katz, and D. Patterson. Coding techniques for handling failures in large disk arrays. In *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 123–132, 1989.
- [7] M. Holland and G.A. Gibson. Parity declustering for continuous operation on redundant disk arrays. *Proc. of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 23–35, 1992.
- [8] E.K. Lee and R.H. Katz. The performance of parity placement in disk arrays. *IEEE Transactions on Computers*, 42:651–664, June 1993.
- [9] Y.-D. Lyuu. *Information Dispersal and Parallel Computation*. Cambridge Univ. Press, Cambridge, England, 1992.

- [10] E. Mendelsohn and A. Rosa. One-factorizations of the complete graph – a survey. *Journal of Graph Theory*, 9:43–66, 1985.
- [11] D.A. Patterson, G.A. Gibson, and R.H. Katz. A case for redundant arrays of inexpensive disks (RAID). *Proc. of the ACM SIGMOD International Conference on Management of Data*, pages 109–116, 1988.
- [12] M.O. Rabin. Efficient dispersal of information for security. *Journal of the ACM*, 36(6):335–348, April 1989.
- [13] C. Stein. A two sample test for a linear hypothesis whose power is independent of the variance. *Annals of Mathematical Statistics*, 16(3):243–258, 1945.
- [14] D. Stodolsky, G.A. Gibson, W.V. Courtright II, and M. Holland. *A redundant disk array architecture for efficient small writes*. CMU CS94-170 Carnegie Mellon University.
- [15] D.G. Wagner. On the perfect one-factorization conjecture. *Discrete Mathematics*, 104:211–215, 1992.
- [16] W.D. Wallis. *One-Factorizations*. Kluwer, 1997.
- [17] Q. Xin, E.L. Miller, D.D.E. Long, S.A. Brandt, T. Schwarz, and W. Litwin. Reliability mechanisms for very large storage systems. *Proceedings of the 30th IEEE/11th NASA Boddard Conference on Mass Storage Systems and Technologies*, pages 146–156, April 2003.
- [18] L. Xu, V. Bohossian, J. Bruck, and D.G. Wagner. Low-density MDS codes and factors of complete graphs. *IEEE Transactions on Information Theory*, 45(6):1817–1826, 1999.

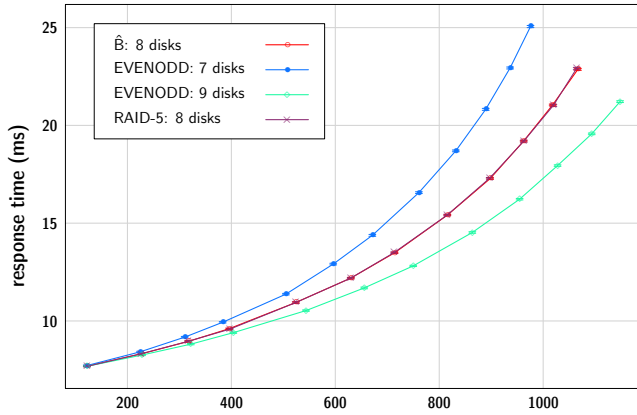


Figure 6a: 8KB read operations per second: all disks operational

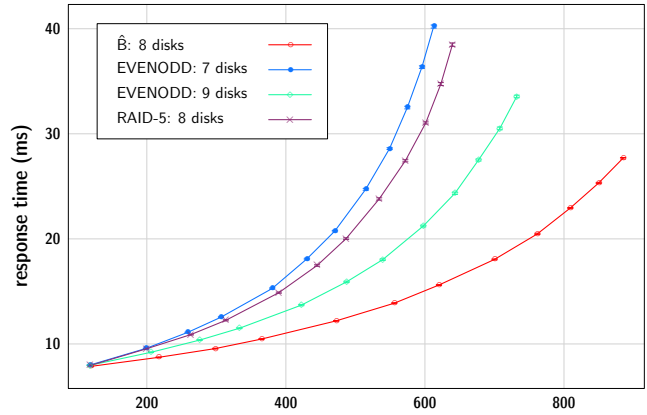


Figure 6b: 8KB read operations per second: one failed disk

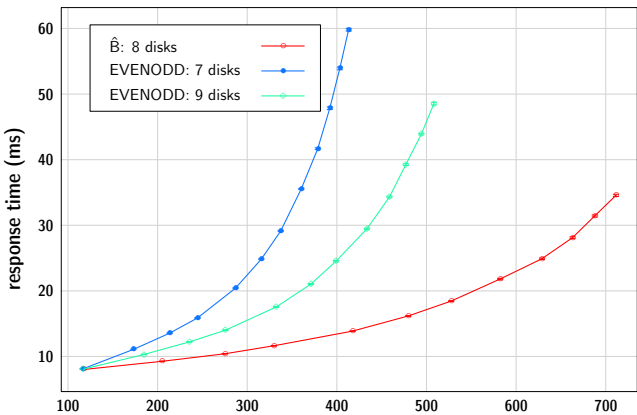


Figure 6c: 8KB read operations per second: two failed disks

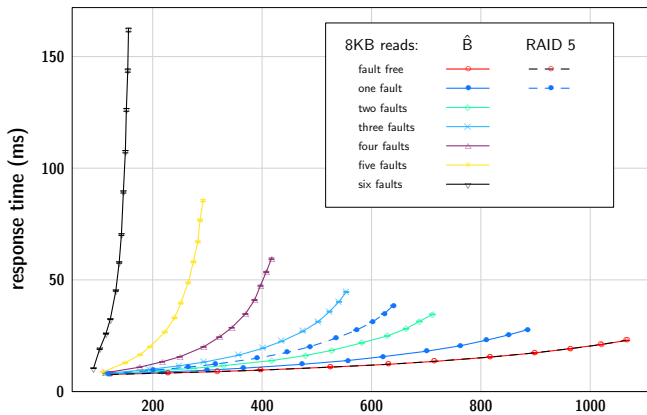


Figure 6d: 8KB read operations per second

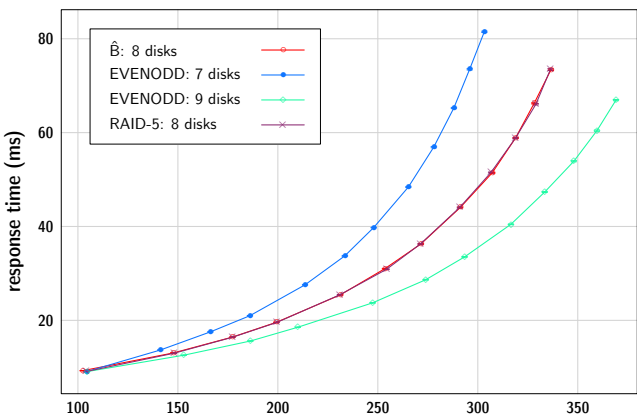


Figure 6e: 32KB read operations per second: all disks operational

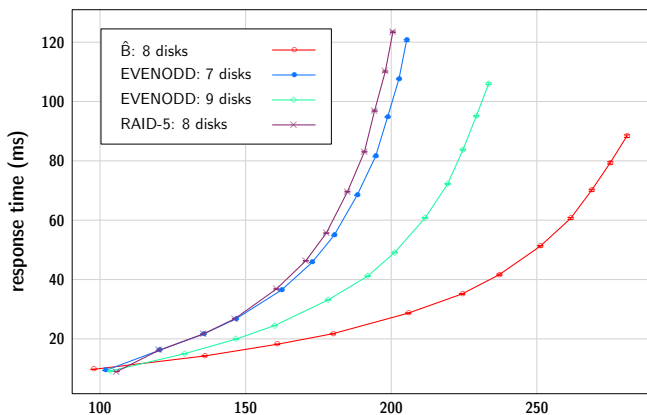


Figure 6f: 32KB read operations per second: one failed disk

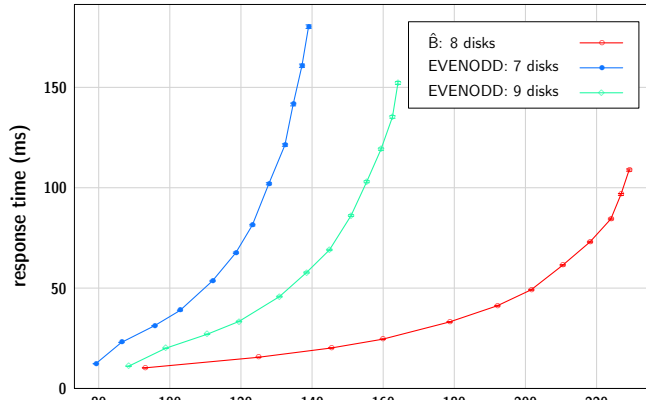


Figure 7a: 32KB read operations per second: two failed disks

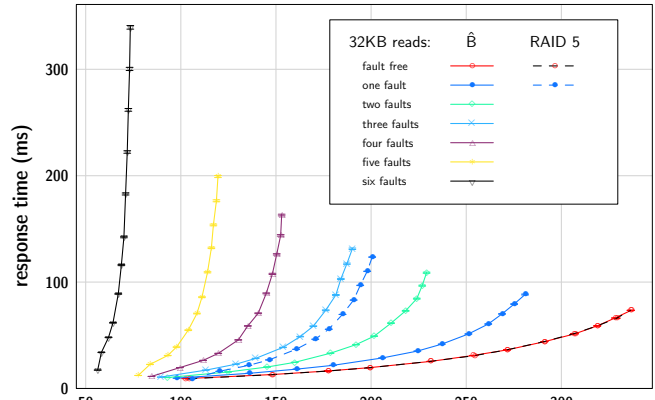


Figure 7b: 32KB read operations per second

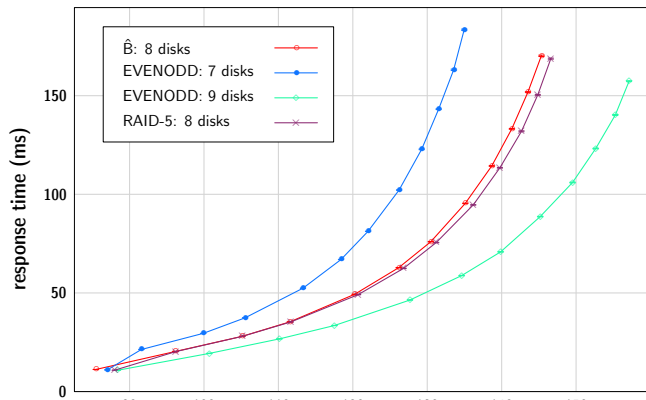


Figure 7c: 96KB read operations per second: all disks operational

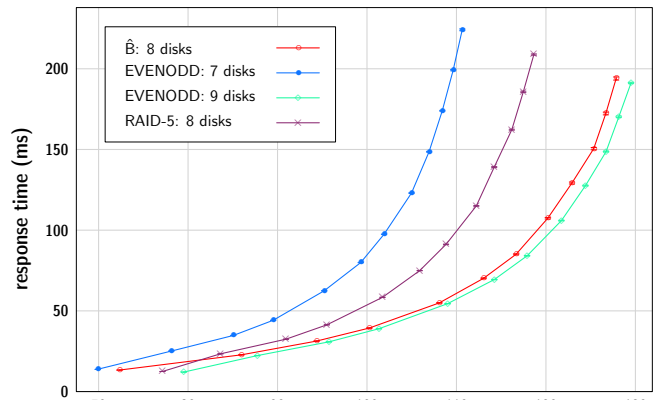


Figure 7d: 96KB read operations per second: one failed disk

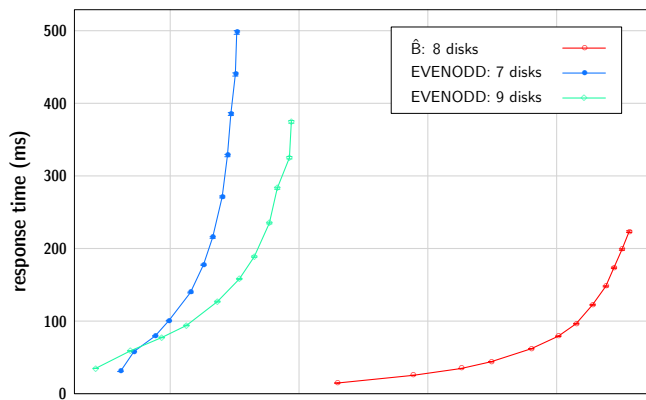


Figure 7e: 96KB read operations per second: two failed disks

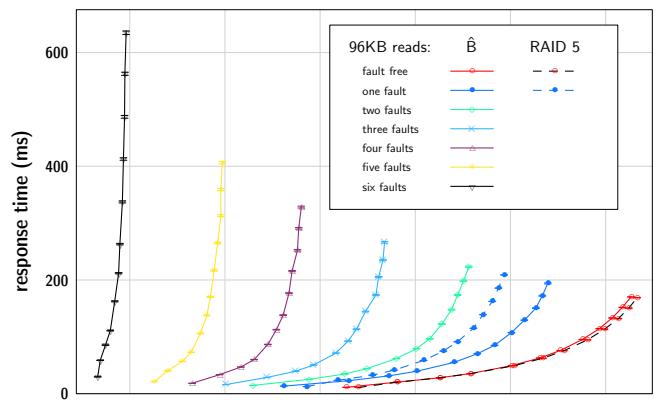


Figure 7f: 96KB read operations per second

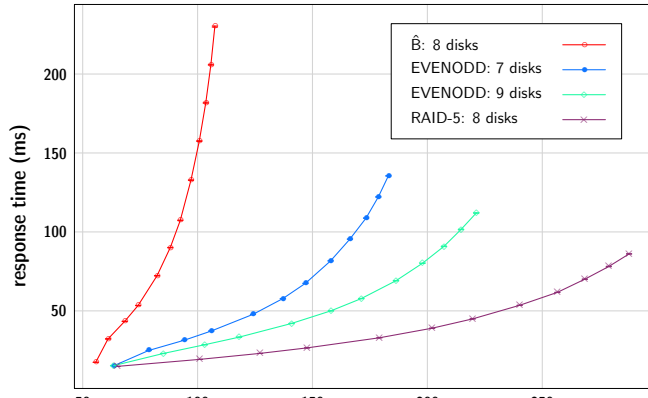


Figure 8a: 8KB write operations per second: all disks operational

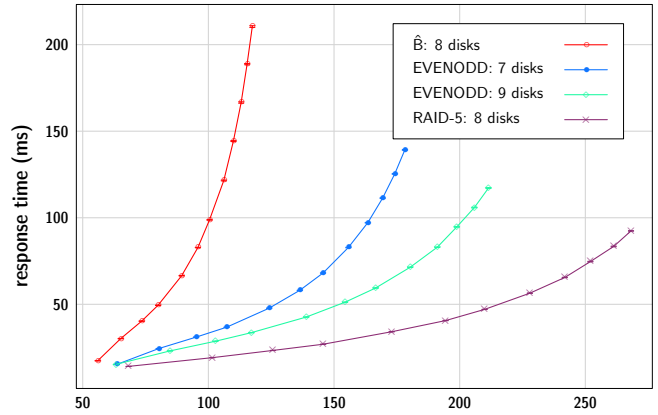


Figure 8b: 8KB write operations per second: one failed disk

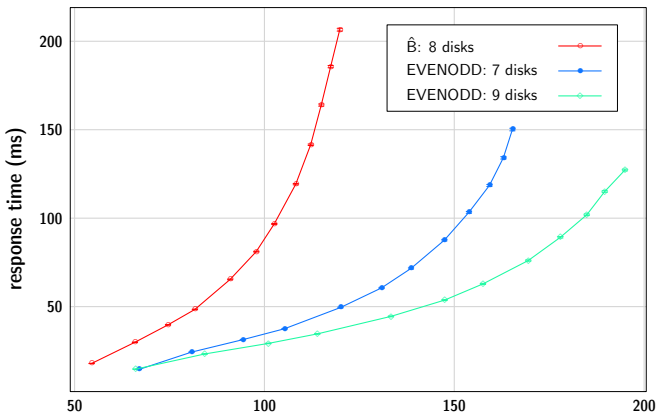


Figure 8c: 8KB write operations per second: two failed disks

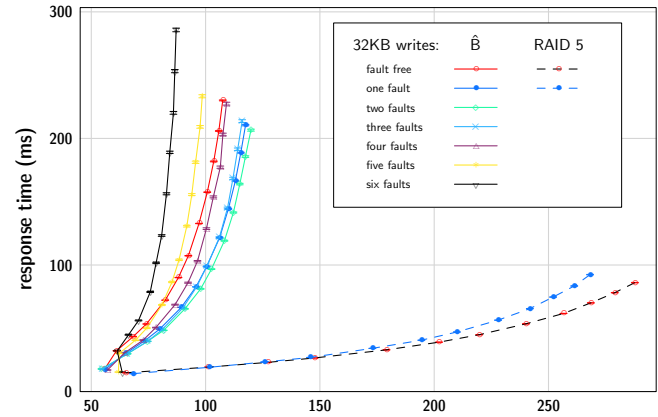


Figure 8d: 8KB write operations per second

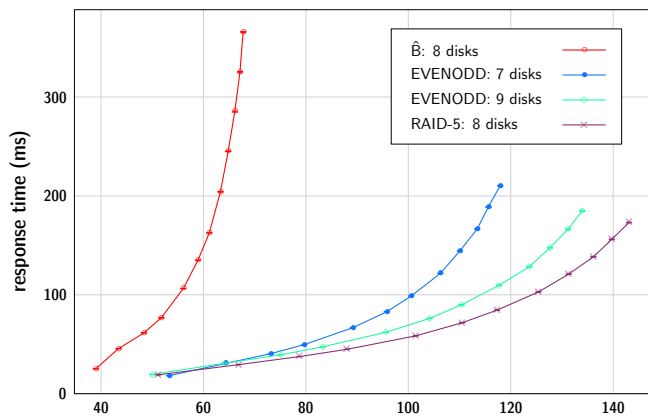


Figure 8e: 32KB write operations per second: all disks operational

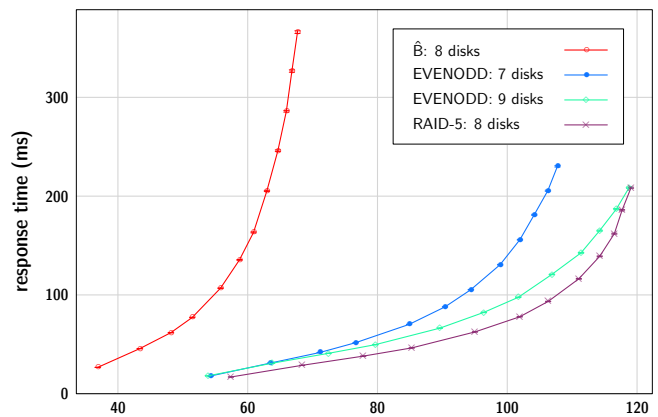


Figure 8f: 32KB write operations per second: one failed disk

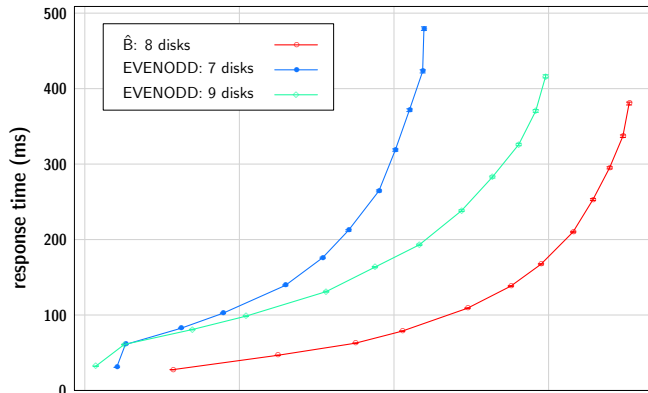


Figure 9a: 32KB write operations per second: two failed disks

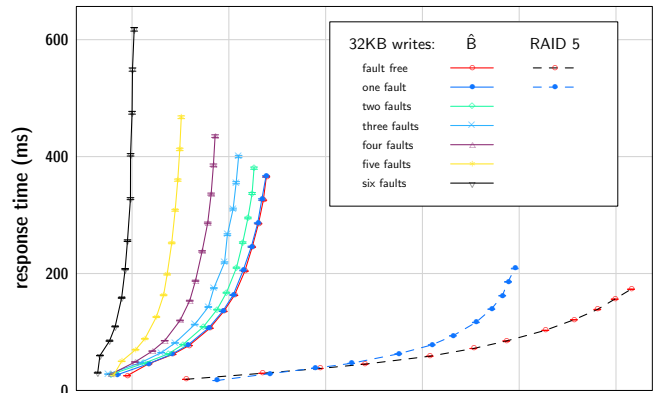


Figure 9b: 32KB write operations per second

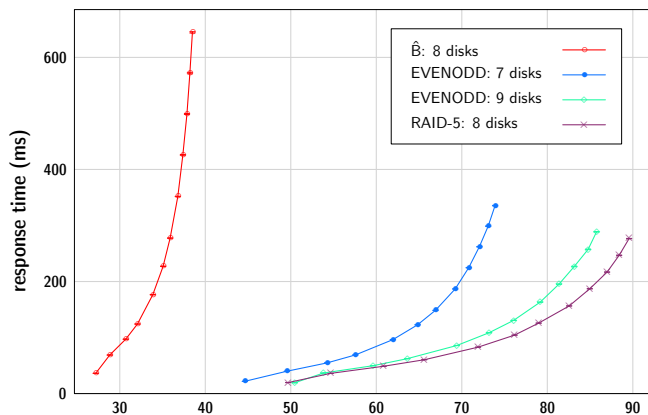


Figure 9c: 96KB write operations per second: all disks operational

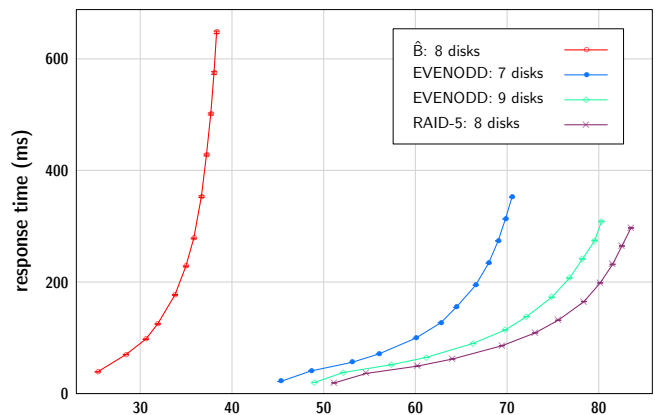


Figure 9d: 96KB write operations per second: one failed disk

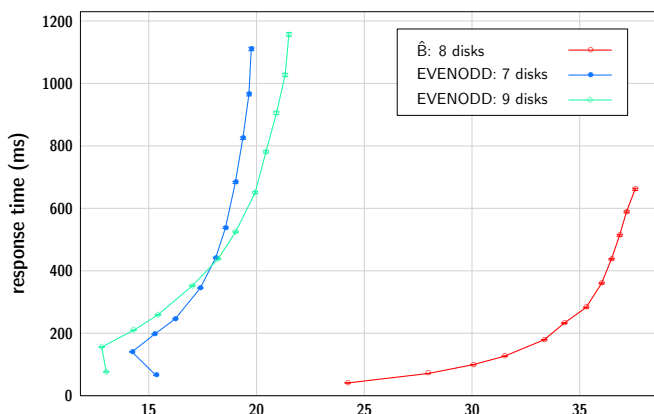


Figure 9e: 96KB write operations per second: two failed disks

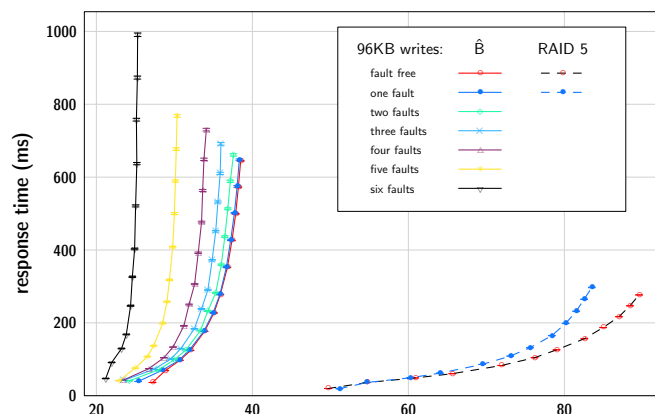


Figure 9f: 96KB write operations per second