Parity Declustering Data Layout for Tolerating Dependent Disk Failures in Network RAID Systems

Wang Gang, Liu Xiao-Guang and Liu Jing
Department of CS, NanKai University, Tianjin, 300071
E-mail: {wang_gang, lxgmail}@eyou.com

Abstract

In this paper, we present a new data layout – "string parity declustering data layout". This data layout incorporates advantages of both orthogonal data layout [2] and weighted parity declustering data layout. The simulation results show that, it improves the reliability of RAID systems (especially network RAID systems) greatly.

1. Introduction

In recent years, the performance gap between CPU, memory system and I/O subsystem has been widening. If the trend continues, future improvements in CPU and memory system performance will be wasted as computer become increasingly I/O bound. To overcome the impending I/O crisis, Patterson et al. have proposed Redundant Arrays of Inexpensive Disks (RAID) [1]. Along with the rapid pace of network technology, network storage has become more and more common in past years. We applied parity declustering technology (originally suggested by Muntz and Lui [3], evaluated by Holland and Gibson [4], and more recently improved upon by many people) to network RAID systems, and proposed a new data layout method – "weight parity declustering data layout" [8]. This data layout improves the failure-recovery performance and the reliability of network RAID systems. However, it doesn't consider dependent disk failures that can severely limit the reliability of network disk arrays.

Most I/O subsystems require support hardware that is shared by multiple disks. For example, power supplies, cabling, cooling, controllers, and computers, network equipments (in network storage systems) are often shared across multiple disks. A collection of multiple disks and shared devices is called a "string" [2]. String may fail if any of the support hardware fails, and string failures can render many disks unavailable. Obviously, if data layout is not designed carefully, one string failure may cause the RAID system to fail. MTTDL of this kind of layouts is:

$$MTTDL_{RAID} = \frac{1}{\frac{GN(N+1)MTTR_{disk}}{MTTF_{disk}^2} + \frac{M}{MTTF_{string}}} \quad (1)$$

Namely, a M string disk array only has 1/Mth reliability of a single string. Generally, the reliability of a string is far less than that of a single disk (especially in network RAID systems that use workstations or PCs as I/O nodes, a string is a computer), thereby the reliability of disk arrays is very poor.

To solve this problem, Gibson et. al. proposed orthogonal organization in [2]. As shown in Fig. 1, orthogonal disk arrays organize parity groups with no more than one disk from each group on any one string, so it guarantees that a single string failure can be endured as long as no other disk or string failure occurs before the string is repaired. Orthogonal organization sloves the string-failure problem successfully, but it is substantially a multiple-group RAID level 5. It has bad degraded- and reconstruction-mode performance [4], and then impacts the reliability. MTTDL of orthogonal organizations is:

$$\frac{\frac{MTTF_{disk}^2}{GN(N+1)MTTR_{disk}}}{\frac{1+\alpha_F}{1+(2N+1)\varepsilon_{dd}+N\varepsilon_{ds}}+\alpha_F\frac{1+\alpha_F\phi/G+(1+\alpha_F/G)(\alpha_R/\alpha_{Rd}+GN\varepsilon_{sd}+(N+1)\phi\varepsilon_{ss}}{((N+1)/MTTF_{string}+(2N+1)\varepsilon_{ss}+GN\varepsilon_{sd})\Psi(N+1)+\Psi(N)}} \quad (2)$$

where λ_d=1/MTTF$_{disk}$, λ_s=1/ MTTF$_{string}$, μ_d=1/ MTTR$_{disk}$, μ_s=1/ MTTR$_{string}$, μ_{dr}=1/ MTTR$_{disk\text{-}recovery}$, $\Psi(g)=\alpha_{Rd}+GN\varepsilon_{dd}+g\Phi\varepsilon_{ds}$, α_F=MTTF$_{disk}$/MTTF$_{string}$, α_R=MTTR$_{disk}$/MTTR$_{string}$, α_{Rd}=MTTR$_{disk}$/MTTR$_{disk\text{-}recovery}$, ε_{dd}=MTTR$_{disk}$/MTTF$_{disk}$, ε_{ss}=MTTR$_{string}$/MTTF$_{string}$, ε_{sd}=MTTR$_{string}$/MTTF$_{disk}$, ε_{ds}=MTTR$_{disk}$/MTTF$_{string}$ and $\phi=\frac{1}{1}+\frac{1}{2}+\frac{1}{3}+\cdots+\frac{1}{G}$.

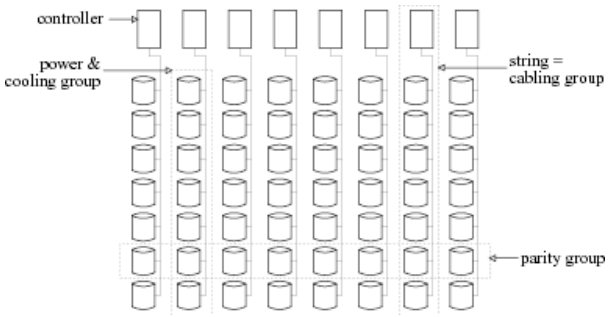


Fig. 1 Orthogonal organization

2. String Parity Declustering Data Layout

To solve imperfections of orthogonal organization and weighted parity declustering data layout, we proposed a new data layout method – "string parity declustering data layout". By using both parity declustering and orthogonal organization technologies in data layout optimization, string parity declustering data layout can tolerate single string failure, and also has good degraded- and reconstruction-mode performance.

The new data layout optimization algorithm comes from simulated annealing algorithm [5]. Performing simulated annealing algorithm on simple randomized layouts can significantly reduce the imbalance in their reconstruction workload., then small randomized layouts can have good reconstruction workload distribution as well. The algorithm is simple: a randomized layout is used as the initial layout, the basic simulated annealing move is the swap of the positions of two units picked from the array at random. For a data layout L, the objective function is:

$$H_{SA}(L) = \sum_{disks\ i,j,i\neq j}(X_{ij})^2 \quad (3)$$

Where X_{ij} is the number of units on disk j that must be read to reconstruct the contents of disk i. Since the mean is fixed by the choice of the number of disks and the length of parity stripe, this is equivalent to minimizing the variance. Minimizing the objective function will make the reconstruction workload distributed more evenly.

Obviously, the data layout generated by simulated annealing algorithm doesn't guarantee to tolerant a single string failure, because of randomness in layout initialization and transformation. We improved simulated annealing algorithm by placing some constraints on layout initialization and transformation to guarantee that no string contains more than one unit from one parity stripe.

Firstly, the layout initialization method was ameliorated. Let N+1 denote the number of strings, and assume parity stripe length is equal to the number of strings. Let G denote the number of disks per string, and number disks from 0 to G • (N+1)-1 in row-column sequence. Let r denote the size (number of rows) of the layout. The new algorithm builds the initial layout as such: distribute unit j (0~N) of stripe i (0~G-1) in each row to disk i• (N+1)+j (disk i on string j). Obviously, this layout is an orthogonal organization. To accelerate optimization, the initial layout can be made more stochastic by using permutation algorithm described in [6]. But only units from one string (unit j of one stripe and unit j of another stripe) can be exchanged in order to guarantee the units from the same stripe are still on different strings.

Secondly, the layout transformation method was improved as well. Like the initial layout randomization described above, only units from one string can be swapped.

According to the environment that the layout will be used in, objective function in (3) (for local RAID) or objective function proposed in [8] (for network RAID) can be used in the new algorithm. We assume that the environment is network RAID system, so the latter is used:

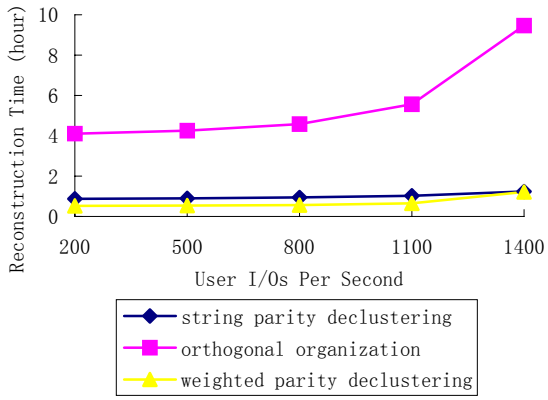
$$H_{WEIGHTED}(L) = \sum_{disks\ i,j,i\neq j} (X_{ij} \bullet e_{ij})^2 \qquad (4)$$

Where e_{ij} denotes the cost that read a unit from disk j when reconstruct disk i. The new algorithm is as follows:

1. initialize parameters for simulated annealing algorithm: t, α, m, n
2. build initial layout – an orthogonal layout L, as current layout
3. do step 4-5 m times
4. generate four random numbers: u (0~r-1), d1, d2 (0~G-1) and s (0~N), swap unit u on disk d1•(N+1)+s and unit u on disk d2•(N+1)+s, build new layout L'
5. if $H_{WEIGHTED}(L') <= H_{WEIGHTED}(L)$ or $random(0,1) \le e^{\frac{H_{WEIGHTED}(L) - H_{WEIGHTED}(L')}{t}}$, accept L' as current layout (L=L'), otherwise still use L as current layout (do nothing)
6. if $H_{WEIGHTED}(L)$ changed in m iterations, t = t•α, goto step 3. otherwise, if $H_{WEIGHTED}(L)$ hasn't changed in last n generations, the algorithm ends.

We have done the simulation to compare the reconstruction-mode performance and the reliability of weighted parity declustering data layout, orthogonal organization and string parity declustering data layout. We used RAIDframe [7] as the simulation platform. Some parameters are described above, other parameters are: the stripe unit size is 8KB. local/remote disk speed ratio is 3 , and Seagate ST32171W was select as disk model. The workload we used comes from [4], it is based on access statistics measured on an airline-reservation system.

Fig.2 illustrates the results. The left figure shows that, orthogonal organization has the worst reconstruction performance in these three data layouts because it doesn't benefit from parity declustering technology. And as the workload climbs, the performance gap becomes wider. As a result of distinguishing between local disk access time and remote disk access time, reconstruction performance of weighted parity declustering disk arrays is better than that of string parity declustering disk arrays. But they are very close. As the workload climbs, the two curves approach each other gradually, because weighted parity



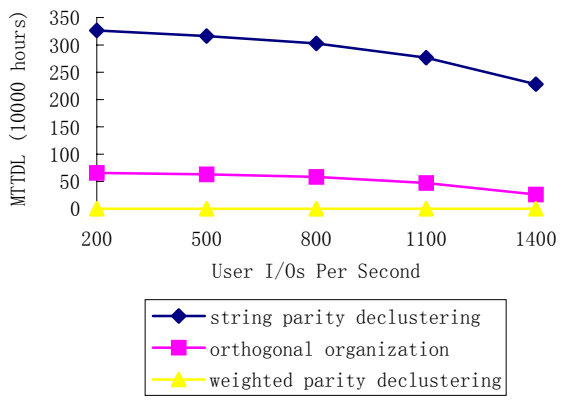


Fig. 2 Comparing three data layouts: reconstruct time and mean time to data loss

The convergence speed of the algorithm is fast. We have done the experiments on the off the shelf PC, the parameters are set as such: N+1=8, G=5, r=117, t=0.5, α=0.9, l=100 and n=10, generating a layout only need several seconds.

3. Simulation Results

declutering data layout is comparatively closer to RAID level 5 [8].

Using the approach proposed in [2], we got MTTDL of string parity declustering data layout:

$$\frac{\frac{MTTF_{disk}^2}{GN(N+1)MTTR_{disk}}}{\frac{G+\alpha_F}{1+(N+1+NG)\varepsilon_{dd}+N\varepsilon_{ds}} + \alpha_F \frac{1+\alpha_F\phi/G+(1+\alpha_F/G)(\alpha_R/\alpha_{Rd}+GN\varepsilon_{sd}+(N+1)\phi\varepsilon_{ss}}{((N+1)/MTTF_{string}+(2N+1)\varepsilon_{ss}+GN\varepsilon_{sd})\psi(N+1)+\psi(N)}} \qquad (5)$$

Using formula (1), (2) and (5), we calculated MTTDL

of three data layouts. The right chart in Fig.2 shows the results, we assume that $MTTF_{disk}$=1,000,000 hours, $MTTF_{string}$=20,000 hours, and $MTTR_{disk}$=$MTTR_{disk\text{-}recovery}$. Although MTTF of string is very low, the reliability of string parity declustering disk arrays is much better than the reliability of a single disk, and is several order of magnitudes better than that of weighted parity declustering disk arrays, and 5~8 times better than that of orthogonal disk arrays. The new data layout improves data reliability significantly.

4. Conclusion

In large disk arrays, reliability is a serious problem. Orthogonal organization can tolerate a single string failure, and improve data reliability greatly. Parity declustering technology improves degraded- and reconstruction-mode performance. We proposed a new data layout method – string parity declustering data layout. This method incorporates the advantages of both technologies. The simulation results show that string parity declustering data layout is much more reliable than other two data layouts, it also has good degraded- and reconstruction-mode performance.

References

1. David A. Patterson, Garth A. Gibson, Randy H. Katz. “A Case for Redundant Arrays of Inexpensive Disks (RAID)”. Proceedings of the 1988 ACM SIGMOD International Conference on Management of Data. 109-116.
2. Garth A. Gibson, David A. Patterson. “Designing Disk Arrays for High Data Reliability”. Journal of Parallel and Distributed Computing 17(1-2), 4-27. 1993.
3. R. Muntz, J. Lui. “Performance Analysis of Disk Arrays under Failure"'. Proceedings of the Conference on Very Large Data Bases, pp.162-173. 1990.
4. M. Holland, G. A. Gibson, D. P. Siewiorek. “Architectures and Algorithms for On-Line Failure Recovery In Redundant Disk Arrays.” Journal of Distributed and Parallel Databases. Vol. 2, No. 3. 1994。
5. Eric J. Schwabe, Ian M. Sutherland, Bruce K. Holmer. “Evaluating Approximately Balanced Parity-Declustered Data Layouts for Disk Arrays”. Parallel Computing, vol 23, No.4-5, pp. 501-523. 1997.
6. Donald. E. Knuth. “The art of computer programming”. Addison-Wesley Pub. Co.. 1973.
7. W. V. Courtright II, G. A. Gibson, M. Holland J. Zelenka. “RAIDframe: Rapid Prototyping for Disk Arrays.” Proceedings of the 1996 ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, p. 268--269. 1996.
8. Wang Gang, Liu Xiaoguang, Liu Jing. “Data Layout of Network Based RAID”. accepted by Journal of Computer Science.