A RAID Reconfiguration Scheme for Gracefully Degraded Operations

Hai Jin[1] Kai Hwang[2] Jiangling Zhang[1]
Huazhong University of Science and Technology, Wuhan, 430074, China[1]
University of Southern California, Los Angeles, California, 90089[2]

Abstract

One distinct advantage of Redundant Array of Independent Disks (RAID) is fault tolerance. But the performance of a disk array in degraded mode is so poor that no one uses the RAID after failure. Continuous operation of RAID in degraded mode is very important in many real time applications, which can not be interrupted in providing continuous services.

In this paper, we propose an efficient architectural reconfiguration scheme to enhance the performance of RAID-5 in degraded mode, called reconfigurable RAID-5. It reconfigures RAID-5 to RAID-0 in degraded mode. Using this scheme, the calculation of the failure data and the generation of parity in writing the new data to the failed disk can be reduced. It also alleviates the small write problem for RAID-5 in degraded mode.

We use the phase parallel model to analyze the total execution time of the RAID-5 and of the reconfigurable RAID-5. Through theoretical analysis and benchmark test, we find the performance of the reconfigurable RAID-5 can be 200 times better than conventional RAID-5.

1 Introduction

Redundant Arrays of Independent Disks (RAID)[3][6] are capable of providing improved levels of reliability, availability, and performance over single disk. A disk array usually provides protection against loss of data from one or more disk failures by maintaining redundant information within the array. Moreover, data availability can be maintained on one or more disk failures by using the redundant information to reconstruct data stored on the failed disk in real time. The design challenge of disk array is that the disk failure should be isolated at the level of redundant disk array controller[4]. Thereby can transparently handle a large class of errors.

Figure 1 shows two typical redundancy scheme of RAID. *Data units* represent the unit of data access supported by the array. *Parity units* represent redundancy information generated from the bitwise exclusive-or (parity) of the collection of data units. Using the redundant scheme of parity can only tolerant single disk failure. Other redundant schemes include using Reed-Solomon code or EVENODD code[2][5] for tolerating up to two disk failures, information dispersal based method DATUM for tolerating multiple disk failures in the disk arrays[1]. The redundancy group formed by a parity unit and the data units it protects is called *parity group*.

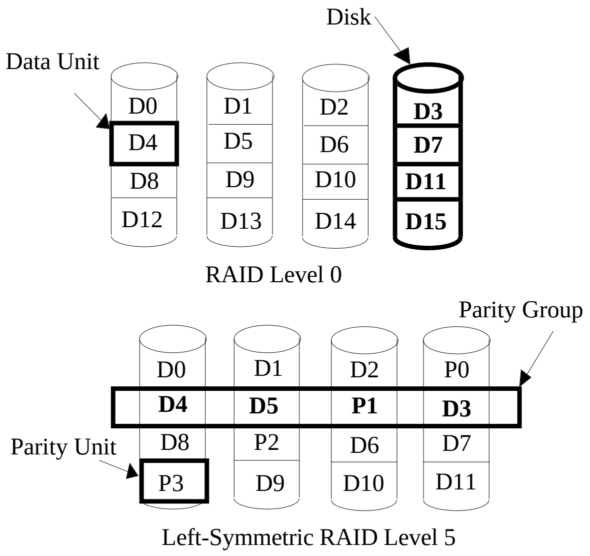

Figure 1. Data Layout for RAID-0 and RAID-5

RAID-0 offers no redundancy so it is not fault tolerant. Data is block-interleaved for optimizing small transfer size with each request being serviced by a single drive. Data may also be bit interleaved, optimizing performance for large transfer sizes, such as VOD application, by using every drive to transfer data in parallel. The main advantages of RAID-0 are highest disk I/O performance, easy to manage and without extra write overhead.

RAID-5 is block interleaved to optimize bandwidth and relies on parity-based redundancy. Both data and parity are evenly distributed throughout the array. There exist a variety of strategies to evenly distribute the data units and parity units[11]. This illustration uses the left-symmetric layout. Parity is the bitwise exclusive-or of all data units in the parity group. A disk array of RAID-5 can continue its operation in degraded mode with one failed disk, but its performance is poor. But in some application, such as multimedia system, the performance of continuous operation of disk array in degraded mode is very critical.

In this paper, we first study some related research background of the architecture of RAID-5 in degraded mode and then propose a new architecture reconfiguration scheme to improve the performance of RAID-5 in degraded mode. The main idea is based on the fact that in degraded mode the fault tolerant ability of RAID-5 is the same as that of RAID-0. There is no need to keep parity information in degraded mode of RAID-5. The best disk array organization in this case is RAID-0.

2 Background and Motivation

Typically, a disk array operates in the following three modes: normal mode, degraded mode and rebuild mode.

Degraded mode refers to the situation that in the case of one disk failure and no attempt has been made to rebuild the data to place it on another drive, disk array system can still work without interrupted[13]. There is a significant increase in the system load. Since reads targeted to the failed disk cause N-1 read accesses to the corresponding blocks on surviving disks to reconstruct lost data block. Writes access targeted to the failure disk cause N-1 read accesses to the corresponding blocks on surviving disks to compute the parity block and one write access to write the newly generate parity information to the block of the corresponding disk drive. The array is said to be in rebuild mode during the time that data on the failed drive is being rebuilt and places on another drive.

The main research on improving the performance of RAID-5 in degraded mode is sparing technique. It is used to reduce the time of disk array in degraded mode and change to the rebuilt mode as quick as possible[3][15]. It can minimize the probability of losing data and the duration of degraded mode operation.

The main problem existing in the sparing technique is the extra space overhead. For a disk array with sparing technique, there are N+2 disks with a fraction (N+1)/(N+2) of the space dedicated to data and parity blocks and 1/(N+2) to space blocks.

For a disk array of RAID-5, it can only tolerate single disk failure. Any disk failure when disk array in degraded mode will cause the loss of data. This is the same situation with the disk array of RAID-0 in normal mode, where any single error will cause the loss of data. That implicates that even we provide the parity information in the disk array of RAID-5 in degraded mode, there is actually no any benefit for the future fault tolerant of disk failure. The only usage of such parity information is used to reconstruct the failure data. For the sparing technique, it uses spare disk space to store this failure data. As there is no use of the parity information for the fault tolerant for the disk array of RAID-5 in degraded mode, it is possible to store the reconstructed failure data in the place where the parity information should store in stead of in the spare disk space. This is the basic point of our new reconfiguration architecture scheme. We call it *reconfigurable RAID-5*.

3 The Basic Reconfiguration Scheme of Reconfigurable RAID-5 Architecture

3.1 Definition

We first give some definitions used in our scheme.

Reconfigurable RAID-5 We call architecture of RAID-5 in degraded mode using our proposed new reconfiguration scheme *reconfigurable RAID-5*. This is because in this new architecture, disk array in degraded mode is not fully RAID-5, it is a combination of RAID-5 and RAID-0.

Real parity group vs. **pseudo parity group** The redundancy scheme in reconfigurable RAID-5 architecture is parity group related. Some parity groups of disk array are RAID-5 structure while the other parity groups are RAID-0 structure. The parity group with RAID-5 structure is called *real parity group***,** while the parity group with RAID-0 structure is called *pseudo parity group*. Whether the parity group is real parity group or pseudo parity group depends on whether this failure data block in this parity group has been accessed in degraded mode. If the failure data block in one parity group is accessed in degraded mode, this parity group is reconfigured to RAID-0 structure and becomes pseudo parity group, otherwise the parity group remains RAID-5 structure, be it real parity group. From the viewpoint of redundancy scheme, real parity group maintains the parity information, while pseudo parity group keeps no redundant information.

Pseudo normal mode In reconfigurable RAID-5 architecture, except the three working modes, there exists another working mode, called *pseudo normal mode*. If the access is referring to the failure data block in pseudo parity group, we call the operation is in pseudo normal mode. This is quite different from normal mode or degraded mode. In normal mode, data access is refer to the normal data block without failure data reference, while in degraded mode, data access is refer to the failure data block in real parity group. Both data accesses in normal mode and degraded mode are with respect to the disk array structure of RAID-5. In pseudo normal mode, data access is to the disk array structure of RAID-0.

ParityLocationMap There is a bit map stored on the non-volatile storage in reconfigurable RAID-5 architecture, called *ParityLocationMap*. In this map, each bit corresponds to one location of parity information in RAID-5. The content of each bit indicates whether the location is stored as parity information or data. We use logical “1” to indicate the location of storing parity information, while logical “0” indicate the data. ParityLocationMap is a dedicate data structure in reconfigurable RAID-5.

3.2 Basic Reconfiguration Scheme of Reconfigurable RAID-5 Architecture

The reconfiguration scheme of reconfigurable RAID-5 consists of two parts, one part for the data access in degraded mode (including the data access in pseudo normal mode), the other for the data access in rebuild mode.

1. Data Access in Degraded Mode

We first discuss the data access of reconfigurable RAID-5 architecture in degraded mode. Suppose the total disks in the array is *N*, the failure disk is $disk_i$. We only concern about the data accessing of failure data block D_i on $disk_i$.

For the read operation from failure data block D_i, the ParityLocationMap is first searched to find out the data type B_i of parity location P_i with the same parity group.

If the data type B_i of P_i is parity, then this parity group is real parity group and is in degraded mode. In this case, *N-1* accesses to the corresponding blocks on surviving disks to reconstruct D_i. At the same time D_i is copied to the location of P_i, and the corresponding bit in the ParityLocationMap should be reset to indicate the data type as data. The data layout in this parity group is RAID-0. This procedure can be illustrated in figure 2.

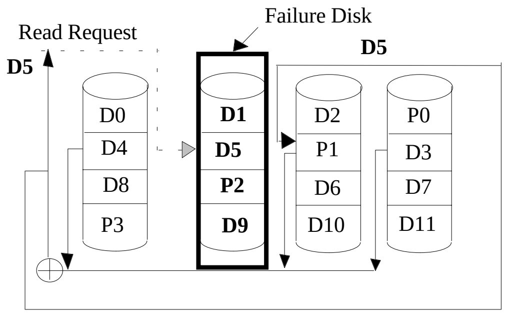

Figure 2. Read failure data from the real parity group in degraded mode

If the data type B_i of P_i is 0, then this parity group is pseudo parity group and is in pseudo normal mode. In this case, failure data block D_i can be read out directly from the place where parity information P_i should reside. As there is no procedure of reconstruct failure data, the read operation is just the same as the one in RAID-0 just one extra search of ParityLocationMap. This procedure is shown in figure 3.

For the write operation to failure data block D_i, whether parity group is real parity group or pseudo parity group, the new data block D'_i should write to the location of P_i. The corresponding bit in the ParityLocationMap should be reset to indicate the data type as reconstructed failure data. The procedure of generate parity is avoided. This procedure is illustrated in figure 4.

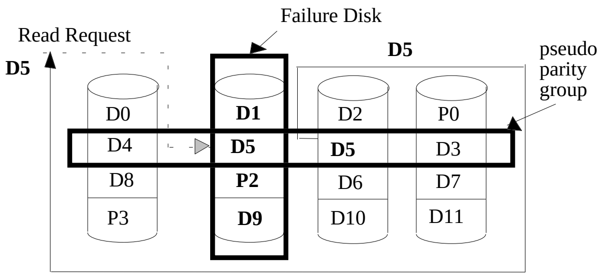

Figure 3. Read failure data from the pseudo parity group in pseudo normal mode

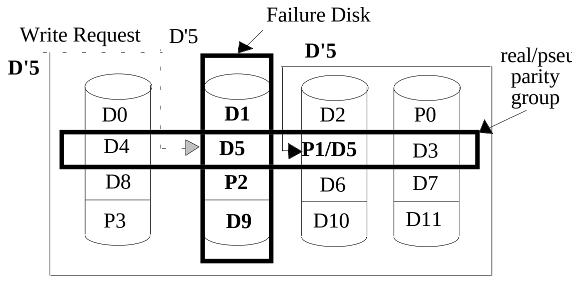

Figure 4. Write data block to the failure disk

2. Data Access in Rebuild Mode

Before discussing the data access operation in rebuild mode, we will first introduce an important concept which used in the process of rebuild operation.

Rebuild line Rebuild line refers to the parity group location where the current rebuild operation is performed. All the data accesses below this rebuild line should perform normal data access operation just as in normal mode, while the data accesses above this rebuild line should perform degraded mode operation.

Let us consider the rebuild operation in reconfigurable RAID-5 system. Suppose the failure disk $disk_i$ is replaced by a new disk drive $disk'_i$, the current rebuild line is L_i. The parity information of this parity group should be P_i if it still preserves. Before the rebuild process begins, the PaityLocationMap should be checked to find out the data type B_i of current parity group.

If the data type B_i indicates the content of current parity information location is parity, that means the rebuild line is of real parity group, the rebuild operation should be performed as the ordinary rebuild operation in traditional RAID-5 system. That is, all the corresponding blocks on surviving disks should be read out to reconstruct failure data D_i and then copy D_i to the corresponding location of newly replaced disk drive. Figure 5 shows the procedure of rebuild in this case.

If the data type B_i indicates the content of current parity information location is data, that means the rebuild line is of pseudo parity group. In this case, rebuild operation should be performed in two parallel steps. On one hand, the data D_i in the position of P_i should be copied to the corresponding position of newly replaced disk. On the other hand, all the corresponding blocks on surviving disks should be read out to generate parity information P_i and copy P_i to the location where the former D_i reside. At the same time, the data type of this parity location in ParityLocationMap should be set. Figure 6 shows the procedure of rebuild in this case.

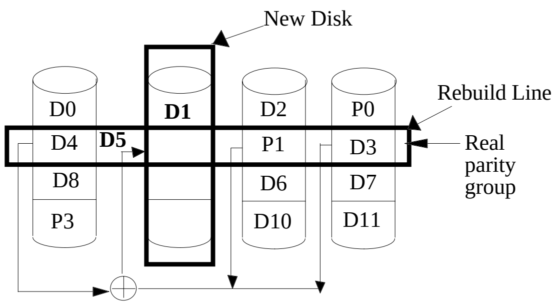

Figure 5. Rebuild failure data when the rebuild line is of real parity group

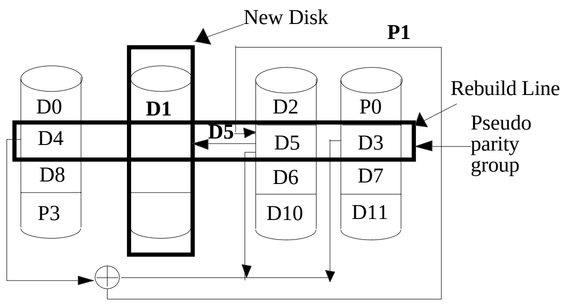

Figure 6. Rebuild failure data when the rebuild line is of pseudo parity group

4 Space Overhead Consideration and Performance Evaluation

In this section, we will discuss two performance topics related to the reconfigurable RAID-5 architecture.

4.1 Space Overhead Consideration

In reconfigurable RAID-5 disk array system, an important data structure is ParityLocationMap, which used to store the status of each parity location to indicate whether it is parity information or data. According to the status of parity location, we can judge whether the parity group is real parity group or pseudo parity group, therefore we can determine the data access in this parity group is in degraded mode or pseudo normal mode.

Because ParityLocationMap should be kept in a non-volatile permanent storage, this causes extra space overhead compared with traditional RAID-5 system. Let us estimate how much the extra space a reconfigurable RAID-5 system needs so that we can determine whether this extra space overhead is a heavy burden to the users.

Suppose the actual capacity of disk array of reconfigurable RAID-5 system is M, that is the capacity user can actually use without including the capacity parity information occupies. In the other word, if there are N disks in the RAID-5 system, the capacity of each disk is M_1, the actual capacity of RAID-5 disk array system is $M=(N-1)\times M_1$. Suppose the stripe block size is B. Then the extra space overhead to maintain the ParityLocationMap is:

$$\frac{M}{8\times B}=\frac{(N-1)M_1}{8\times B} \quad bytes \tag{1}$$

Let us give a concrete example to show the extra space overhead of a reconfigurable RAID-5 system. Suppose the scale of a reconfigurable RAID-5 system is 11 disk drives. 10 disk spaces for data and 1 disk spaces for parity information. The capacity of each disk is 10GB. The stripe block size is 32KB, which is often used in the disk array in the environment of UNIX operating system. Then according to the above equation, we can get the extra space overhead to maintain the ParityLocationMap in such reconfigurable RAID-5 system is 391KB. For the disk array with 100GB capacity to have extra space overhead of 391KB to maintain ParityLocationMap is acceptable. Compared with sparing technique, the space overhead in reconfigurable RAID-5 is nominal.

4.2 Basic Phase Parallel Model of Parallel Storage System

In this section, we will apply the phase parallel model[8] to the performance evaluation of storage system.

Phase parallel model is used for the modeling of parallel computation. It divides the execution of a parallel program into a sequence of phases. The next phase begins only after all operations in the current phase have finished. There are three different phase types, parallelism phase, computation phase, and interaction phase. The execution time of the superstep on n processors is[8]:

$$T_n = T_{comp} + T_{interact} + T_{par} \tag{2}$$

According to the principle of phase parallel model, we propose a scheme to evaluate the storage system using the phase parallel model. The phases in parallel storage system are classified as following different types:

(1) *CPU working phase*, which performs the process management in parallel storage controller, such as command split, optimize and combine; redundancy information generate; failure data reconstruct and rebuild; data decomposit and combine.

(2) *Disk service phase*, which one or more disks in the parallel storage system each execute a number of local I/O service operations. "Local" here means that all data needed by a disk drive are available in its local memory. Disk service phase of each disk drive composes of two sequential operations: seek operation and rotation operation. Therefore, the time for each disk service phase composes two parts, seek time and rotation delay.

(3) *Data channel interaction phase*, which executes interaction operations between each disk drive and CPU. It includes data transfer phase and channel delay phase. Therefore the time for data channel interaction phase includes two parts, data transfer time and channel delay time. It is the maximum value of data transfer time and channel delay time of each disk drives within per channel.

Thus, the total execution time for a superstep of I/O on *n* disk drives can be calculated as follow:

$$T_n = T_{CPU\_working} + T_{channel\_interaction} + T_{disk\_service} \quad (3)$$

Suppose the total workload is *w*. The time of CPU working phase can be express as two parts, busy time and control time. Busy time refers to the time of command split, optimization, and combination, while control time is the time for disk array controller to process amount of data, such as data striping and combination, parity calculation and the recovery of lost data. That is:

$$T_{CPU\_working} = T_{busy} + T_{control} \quad (4)$$

The busy time of CPU can be calculated as [10]:

$$T_{busy} = w \cdot (I/B) \cdot 1/IPS + p \cdot w \cdot \tau \quad (5)$$

here, *I/B* is the average number of instructions for CPU to process one byte of data. The value of *I/B* depends upon the different application environments. *1/IPS* is the average time to execute one instructions. It depends on the speed of CPU. *p* is the times to access disk while exchanging amount of data between CPU and disk. The value of *p* varies with the management of device and the organization of file. τ is the access period of memory.

The control time can be calculated as follow[10]:

$$T_{control} = \left(I_{strip} + I_{check}\right) \cdot w \cdot t_{control} \quad (6)$$

here, I_{strip} is the average instruction number for controller to perform data striping while writing, or to combine the data while reading for every byte. I_{check} is the average instruction number for controller to calculate parity information while writing in both normal mode and degraded mode, or to recovery lost data by using parity information and other data on surviving disk drive while reading in degraded mode. $t_{control}$ is the average time for controller to perform each instruction.

Thus, for read operation of RAID-5 in normal mode:

$$T_{control} = I_{strip} \cdot w \cdot t_{control} \quad (7)$$

while for the write operation:

$$T_{control} = \left(I_{strip} + I_{check}\right) \cdot w \cdot t_{control} \quad (8)$$

For read/write operations of RAID-5 in degraded mode:

$$T_{control} = \left(I_{strip} + I_{check}\right) \cdot w \cdot t_{control} \quad (9)$$

The disk service time of each disk drive in the disk array can be calculated as follow [10]:

$$T_{disk\_service}(x, y) = p \cdot k \cdot m \cdot [T_s + T(x) + T(y)] \quad (10)$$

here, *k* is the number of tracks for disk drive to refer during one disk access. It depends upon the organization and the amount of data to access in one time. Suppose the capacity of one track is B_T, then $k > w/B_T$. *m* is the access times of each disk drive when CPU exchanges amount of data with that disk drive. It has the relationship with the amount of data to exchange and the amount of data each disk drive can access once, suppose L_o. T_s is the time needed for the head of each disk drive to stall during one disk access.

T(x) is the time needed for the head of disk drive to seek *x* tracks during one disk access. It is the function of random variable *x*. That is, the time for the head of disk to seek is different during each seek operation. We use the following analytic expression to calculate *T(x)* [10]:

$$T(x) = a\sqrt{x-1} + b(x-1) + c \qquad (M > x \geq 1) \quad (11)$$

here, *x* is the seek distance, *M* is the total track number of the disk. The coefficients *a*, *b*, and *c* can be gotten using minimum seek time, T_{min}, that is the time to seek one track (here *x=1*), maximum seek time, T_{max}, that is the time for head to seek from most inner track to most outer track (here *x=M-1*), and the average seek time, T_{avg}.

T(y) is the time needed for the head of disk drive to rotate *y* sectors during one disk access. It is the function of random variable *y*. It has the relationship with the rotation speed of disk drive and the angles needed to rotate in order to wait for the needed sector after indexed. The way to calculate *T(y)* is [10]:

$$T(y) = (t_r / S)y \quad (12)$$

Here, *S* is the number of sectors per track of disk. t_r is the one rotation time of disk drive.

The data channel interaction time of each disk drive in the storage system can be calculated as follow [10]:

$$T_{channel\_interactiont} = p \cdot m \cdot T_r + d_o \quad (13)$$

d_o is the channel delay time. It depends on how many disk drives will be accessed each time. The more disk drives involved, the longer the channel delay time. T_r is the time needed for amount of data to exchange between

CPU and each disk drive at one time. It is the time when L_O so much data to exchange between CPU and disk drive. Suppose the data transfer rate for each disk drive in the disk array is D_r, then T_r can be calculated as follow [10]:

$$T_r = 8L_o / D_r \tag{14}$$

Therefore, we can use the above equations to calculate the total execution time of storage system.

4.3 Performance Evaluation of Reconfigurable RAID-5

Using the theoretical method discussed above, we can discuss the total execution time of traditional RAID-5 and the reconfigurable RAID-5 in degraded mode respectively.

Among the three phases in above model, disk service phase and data channel interaction time are much more time-consuming. They make great contribution to the total execution time. This is because of the mechanical delay in the components of disk drives. The magnitude of disk service time is usually in milliseconds, while the memory access time is in microseconds and the CPU working time is in nanoseconds. Therefore, we only discuss the disk service time and channel interaction time in the total execution time.

Let us first calculate the disk service time and channel interaction time of traditional RAID-5 in degraded mode. For the read access of failure data block, *N-1* accesses to the corresponding blocks on the surviving disks have to be performed to reconstruct lost data block. The disk service time of the storage system should be the maximum value of the disk service time of each disk drives in the array. Suppose there are *c* disk drives in each channel. The longest channel delay time should be *c* times of channel delay of single disk drive. Then the total execution time not including CPU working time in this case is:

$$T_{RAID\_5\_read} = \underset{i=1\,(i \neq j)}{\overset{N}{MAX}} (T_{disk\_service}(i, x, y)) + c \cdot d_o \tag{15}$$

j is the disk number of failure disk.

For the write access of failure data block, *N-1* accesses to the corresponding blocks on the surviving disks have to be performed to reconstruct lost data block first. Then the new parity information is generated by performing exclusive OR operation on the newly reconstructed lost data block, the new data block should be written to the failure disk, and the old parity. The newly calculated parity information should be written to the corresponding disk. The longest channel delay consists two parts, one is the longest channel delay while *N-1* data blocks are read out from the disk array, the other is the channel delay while the new parity information is writing to the parity disk block. The total execution time in this case is:

$$\begin{aligned} T_{RAID\_5\_write} &= \underset{i=1}{\overset{N}{MAX}} (T_{disk\_service}(i, x, y)) \\ &+ T_{disk\_service}(disk_{parity}, x, y) + (c+1) \cdot d_o \end{aligned} \tag{16}$$

Turn back to the data access of reconfigurable RAID-5 architecture. For simplicity, we only discuss the situation of the data access in pseudo normal mode. Because only in pseudo normal mode, the advantages of reconfigurable RAID-5 architecture can be illustrated completely, while in degraded mode reconfigurable RAID-5 has the same characteristics as traditional RAID-5 system. For the read access of failure data in pseudo normal mode, failure data block can be read out directly from the place which parity information should reside. The only extra operation is performed by CPU in the controller to search the ParityLocationMap to find out the physical address where parity information should reside. Compared with the exclusive OR operation of reconstructing the failure data in RAID-5, the time to search ParityLocationMap is much less. As there is only one disk drive involved in the read access, the channel delay is only one disk drive channel delay. The total execution time of read operation for reconfigurable RAID-5 architecture is the same as single disk drive execution time. That is:

$$T_{Variant\_RAID\_5\_read} = T_{disk\_service}(disk_{parity}, x, y) + d_o \tag{17}$$

For the write access of failure data in the pseudo normal mode, the new data block is just written to the location where parity information should reside. It avoids the generation of new parity information. Thus, the total execution time is the same as the read access of reconfigurable RAID-5 architecture in pseudo normal mode. That is:

$$T_{Variant\_RAID\_5\_write} = T_{disk\_service}(disk_{parity}, x, y) + d_o \tag{18}$$

In order to indicate the performance gain in the reconfigurable RAID-5 architecture, we define *Performance Improvement Ratio (PIR)* as the ratio of the total execution time of RAID-5 to that of reconfigurable RAID-5 system. That is:

$$PIR_{read} = \frac{T_{RAID\_5\_read}}{T_{Variant\_RAID\_5\_read}} \tag{19}$$

$$PIR_{write} = \frac{T_{RAID\_5\_write}}{T_{Variant\_RAID\_5\_write}} \tag{20}$$

From the above discussion, we find the disk service time and channel interaction time of reconfigurable RAID-5 architecture is greatly reduced compared with that of RAID-5. The other advantage is that for the write access in reconfigurable RAID-5, it also delimitates the small write problem in degraded mode, which is the severe problem in RAID-5.

5 Simulation Results

We have already implemented the reconfigurable RAID-5 architecture on the platform of HUST_RAID. It is a complete experimental platform of disk array subsystem with two string, each string connected three 540MB Quantum hard disk drives. Totally, there are six disk drives in the system. We use Qbench designed by Quantum Co. as the benchmark to test the performance of disk array subsystem under the different schemes.

We first test the sequential access of RAID-5 in degraded mode and reconfigurable RAID-5 in pseudo normal mode, respectively. Using the equations (19) and (20), we can get the performance improvement ratio for sequential read and sequential. The result is shown in figure 7. It is very easy to get the data of RAID-5 in degraded mode using Qbench. In order to test the true performance of reconfigurable RAID-5, that is to guarantee the data access of failure disk is in pseudo normal mode, we first run Qbench in degraded mode of reconfigurable RAID-5 without recording the data several times. This can guarantee the later data access of failure disk is referring to the location of parity information via searching the ParityLocationMap. In this way, we can get the approximate true performance data of reconfigurable RAID-5 in degraded mode.

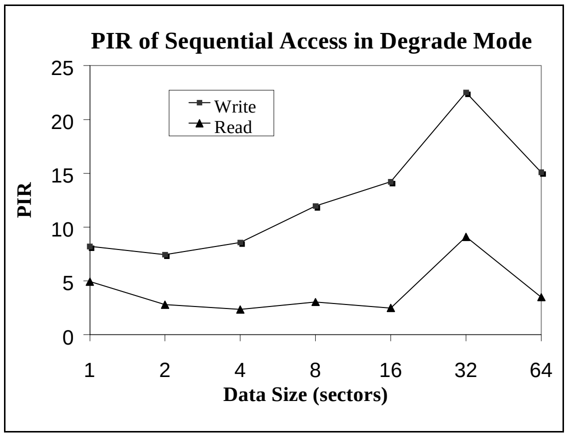

Figure 7. Service time of sequential access in degraded mode

Using the same scheme, we get the data of disk service time and channel interaction time for random access of RAID-5 in degraded mode and reconfigurable RAID-5 in pseudo normal mode, respectively. The result is shown in figure 8.

From the test result, we can see that using reconfigurable RAID-5 scheme for the disk array in degraded mode can greatly enhance the on-line performance. For the data access to the failure disk of RAID-5 in degraded mode, all the surviving disk drives should be accessed to reconstruct the failure data. Thus, greatly reduce the performance of storage system. Because in this case, RAID-5 system has no contribution to the fault tolerant, maintaining the parity information becomes a burden of the storage system. For the reconfigurable RAID-5 architecture, for sequential access the performance improvement ratio can achieve up to more than 20. For random access, the performance improvement ratio is even large, up to more than 200.

The great difference between the performance improvement ratio for the different access pattern relies on the fact that for the sequential access the data locality is much higher than random access. The prefetching of data from disk drives in the array greatly reduced the disk service time and channel delay time.

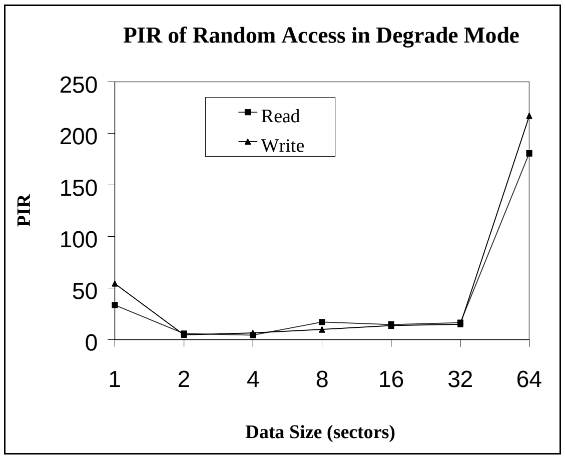

Figure 8. Service time of random access in degraded mode

From the above test result, we also find that channel delay time still occupy the very large portion in the data channel interaction phase. In our platform, there are only two channels and each channel connected three disk drives. Therefore, these three disk drives will have a contention for data channel for the data transferring. In the degraded mode of RAID-5 in our platform, the longest channel delay are three times of single disk drive, because there exist one data channel that the data from the three disk drives should be read out to reconstruct the failure data. In reconfigurable RAID-5 architecture scheme, as there is no need to maintain the parity information, the failure data can be accessed directly from the location of parity information should reside. In most storage system, the number of data channel is limited due to the complexity of design, but the scale of storage system can be very large. The more disk drives in on string, the longer channel delay for the data access in RAID-5 system. From this point of view, reconfigurable RAID-5 architecture scheme is very attractive.

6 Future Work and Conclusions

In this paper, we propose a new and efficient architecture reconfiguration scheme to enhance the performance of RAID-5 in degraded mode. The main idea is based on the fact that in degraded mode of RAID-5 parity information has no contribution to the fault tolerant to the extra disk failure, in this case the fault tolerant ability of RAID-5 is the same as that of RAID-0. There is no need to keep parity information in degraded mode of RAID-5 at all. The best disk array organization in this case is RAID-0. Reconfigurable RAID-5 is the best candidate architecture scheme for RAID-5 to enhance the on-line performance in degraded mode. It reconfigures RAID-5 to somewhat like RAID-0 in degraded mode. Using this scheme, we reduce the calculation of the failure during each read operation and the generation of parity information when operating the data to the failure disk.

The other advantage is that for the write access in reconfigurable RAID-5, it also delimitates the small write problem in degraded mode. We also implemented the reconfigurable RAID-5 architecture scheme on the platform of HUST_RAID. From theoretical analysis and the benchmark test, we find the performance improvement ratio (PIR) of reconfigurable RAID-5 to RAID-5 in degraded mode can be more than 200 fold in the best case. Therefore using reconfigurable RAID-5 architecture scheme can enhance the performance of RAID-5 in degraded mode greatly.

From the study of reconfigurable RAID-5 architecture scheme, we can extend our architecture reconfiguration scheme from tolerating single disk failure to tolerating multiple disk failures. If disk array subsystem have the ability of fault tolerant of two disk failures in the system, we usually use Reed-Solomon code or EVENODD code[2][5]. In the case of one disk failure, the fault tolerant of storage system can only tolerate one extra disk failure. The reliability in this case is the same as that of RAID-5. Thus, we can reconfigure the disk array architecture using complex redundant scheme of Reed-Solomon code or EVENODD code to the simple parity code. Thus greatly reduce the complexity of the procedures of encoding and decoding. The performance in this degraded mode will improve.

Continue with this procedure, if there is another disk failure, we will reconfigure the disk array architecture from RAID-5 to reconfigurable RAID-5.

Same reconfiguration scheme can also applied to the method of DATUM [1] for tolerating multiple disk failures in the disk arrays. The architecture reconfiguration scheme for this multi-level mapping from different disk array architecture to reconfigurable RAID-5 architecture in degraded mode while keeping the on-line performance is still an open problem.

References

[1] G. A. Alvarez, W. A. Burkhard, and F. Cristian, "Tolerating Multiple Failures in RAID Architectures with Optimal Storage and Uniform Declustering", *Proceedings of the 24th Annual ACM/IEEE International Symposium on Computer Architecture*, 1997, pp.62-72

[2] M. Blaum, J. Brady, J. Bruck, and J. Menon; "EVENODD: An Optimal Scheme for Tolerating Double Disk Failures in RAID Architectures," *Proceedings of 21st Annual International Symposium on Computer Architecture*, 1994, pp.245-254

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

[4] W. V. Courtright II, and G. A. Gibson; "Backward Error Recovery in Redundant Disk Arrays", *Proceedings of the 1994 Computer Measurement Group Conference (CMG)*, Vol.1, 1994, pp.63-74

[5] D. Feng, H. Jin, and J. L. Zhang; "Improved EVENODD Code", *Proceedings of 1997 IEEE International Symposium on Information Theory*, 1997, p.261

[6] G. A. Gibson*; "Redundant Disk Arrays: Reliable, Parallel Secondary Storage"*, MIT Press, 1992

[7] J. L. Hennessy, and D. A. Patterson; "*Computer Architecture: A Quantitative Approach*", Second Edition, Morgan Kaufmann, 1996

[8] K. Hwang, Z. Xu*; "Scalable Parallel Computing: Technology, Architecture, programming*", WCB/McGraw-Hill Co., 1998

[9] R. Jain, J. Werth, and J. C. Browne; "*Input/Output in Parallel and Distributed Computer Systems*", Kluwer Academic Publishers, 1996

[10] H. Jin, H. Yang, and J. L. Zhang; "On-line Performance Evaluation of RAID 5 using CPU Utilization", *Proceedings of Signal and Data Processing of Small Targets 1998*, SPIE, Vol. 3373, 1998, pp. 498-509

[11] Hai Jin, and Kai Hwang, "Reconfigurable RAID Architectures for Designing Workstation Clusters with a Single I/O Space", Technical Report, The University of Hong Kong, November 1998

[12] E. Lee, and R. Katz; "The Performance of Parity Placement in Disk Arrays*", IEEE Transactions on Computers*, Vol.C-42, No.6, 1993, pp.651-664

[13] J. Memon, and D. Mattson; "Comparison of Sparing Alternatives for Disk Arrays", *Proceedings of the 19th Annual International Symposium on Computer Architecture*, 1992, pp.318-329

[14] A. L. N. Reddy, J. Chandy, and P. Banerjee; "Design and Evaluation of Gracefully Degradable Disk Arrays", *Journal of Parallel and Distributed Computing*, Vol.17, No.1, 1993, pp.28-40

[15] A. Thomasian, and J. Menon; "RAID5 Performance with Distributed Sparing", *IEEE Transactions on Parallel and Distributed Systems*, Vol.8, No.6, June 1997, pp.640-657