11-19-2007

An Architectural Approach to Improving the Availability of Parity-Based RAID Systems

Lei Tian
University of Nebraska - Lincoln, tian@cse.unl.edu

Hong Jiang
University of Nebraska - Lincoln, jiang@cse.unl.edu

Dan Feng
Huazhong University of Science and Technology, China

Qin Xin
Data Domain Inc.

Sai Huang
Huazhong University of Science and Technology, China

University of Nebraska–Lincoln, Computer Science and Engineering
Technical Report TR-UNL-CSE-2007-0022
Issued Nov. 19, 2007

An Architectural Approach to Improving the Availability of Parity-Based RAID Systems

Lei Tian[1,2], Hong Jiang[1], Dan Feng[2], Qin Xin[3], Sai Huang[2]
[1] Department of Computer Science and Engineering, University of Nebraska-Lincoln
[2] School of Computer Science and Technology, Huazhong University of Science and Technology, China
[3]Data Domain Inc.
E-mail: {tian, jiang}@cse.unl.edu, dfeng@hust.edu.cn, qxin@ieee.org, dshust@163.com

Abstract:

In this paper, we propose an architectural approach, Supplementary Partial Parity (SPP), to addressing the availability issue of parity encoded RAID systems. SPP exploits free storage space and idle time to generate and update a set of partial parity units that cover a subset of disks (or data stripe units) during failure-free and idle/lightly-loaded periods, thus supplementing the existing full parity units for improved availability. By applying the exclusive OR operations appropriately among partial parity, full parity and data units, SPP can reconstruct the data on the failed disks with a fraction of the original overhead that is proportional to the partial parity coverage, thus significantly reducing the overhead of data regeneration, especially under heavy workload. By providing redundant parity coverage, SPP can potentially tolerate more than one disk failure with much better flexibility, thus significantly improving the system's reliability and availability.

Due to its supplementary nature, SPP provides a more efficient and flexible redundancy protection mechanism than the conventional full parity approach. SPP offers multiple optional levels depending on partial parity coverage and performance/cost targets. According to the actual workload and the available resource, the SPP approach can be adaptively and dynamically activated, deactivated and adjusted while the original RAID system continues to serve user requests on-line. We conduct extensive trace-driven experiments to evaluate the performance of the SPP approach. The experiments results demonstrate that SPP significantly improves the reconstruction time and user response time simultaneously.

1. Introduction

Partial or complete disk failures are becoming increasingly common and frequent in modern-day large-scale data centers as the number and capacity of disks increase rapidly while individual disk failure rates remain almost unchanged. It thus becomes imperative for storage systems in general, and those in data centers in particular, to be capable of tolerating disk failures, hopefully more than one simultaneous failure, and regenerating data of the failed disks in as smallest amount of time as possible without noticeably affecting user applications. To address the problem, most current research work focuses not only on the statistical analyses, trend predications, and impact evaluations of the disk failures in the real world [1-3], but also on improving existing failure-tolerant technologies deployed in the file system layer [4, 5] or inside disk-based systems [6-9].

As a fundamental component for availability, redundant disk arrays [10] have been widely deployed in modern storage systems to resist disk failures. By employing redundancy mechanisms such as mirroring, parity-encoding, and hot-sparing, disk arrays can tolerate disk failures and automatically rebuild the lost data on a failed disk onto a replacement disk in the background. Although many online recovery mechanisms have been proposed and proven to be effective to some extent, the problem of availability in

large-scale storage systems remains a daunting challenge in light of the increasing demand for availability and thus clear trend of ultra scales in such systems. In an ultra-scale storage system, such as a Peta-byte-scale data center [11] where hundreds of thousands of hard-disks are deployed, disk failures are becoming more frequent and omnipresent, giving rise to the likelihood that multiple reconstructions in the corresponding RAID sets could be carried out simultaneously due to significantly high disk failure rates and increasingly long recovery time. For example, [12] expected that hundreds of concurrent reconstructions would soon become inevitable in such large data centers and proposed that "systems need to be designed to operate in repair". On the other hand, while hard-disk technology has advanced rapidly in terms of capacity and cost, other performance parameters of hard-drives such as bandwidth, seek latency and failure rate have been improved much more slowly in contrast, effectively lengthening the time to rebuild a failed disk, resulting in a longer "window of vulnerability" in which a second disk failure may occur. This problem will only be exacerbated in the emerging mobile data centers where disk arrays are more prone to failures than their stationary counterparts because of their harsh application and operational environments. In addition, mobile disk arrays tend to deploy small-form-factor disks, which have higher annual failure rates than desktop or enterprise hard disks, for the advantages of power-efficiency and mobility, thus further sacrificing system reliability and availability.

We believe that *availability should be a first-class RAID concern*, similar to the design principle in [4] ----"*reliability should be a first-class file system concern*". Therefore, the design of effective and efficient recovery mechanisms, which reduce both reconstruction time and user response time while being able to tolerate multiple failures, is an important approach to providing both availability and long-term reliability for storage systems.

However, existing the-state-of-art redundancy protection mechanisms and recovery approaches cannot fully cope with the above problems. D-GRAID [6] ensures the availability of most files within the file system even though unexpected multiple disk failures occur, and provides the live-block recovery to restore only live file system data. Its weakness lies in that it needs acquiring the liveness information of upper file systems or databases for recovery, thus incurs more implementation complexity and portability issues. PRO [7] exploits the popularity of workloads and sequentiality of hard disks to rebuild the frequent-accessed data areas prior to other data areas, so the improvement by PRO will be limited if the popularity of workloads is not high or the underlying disks are not hard disks.

In this paper, we propose a Supplementary Partial Parity mechanism (SPP), an architectural approach to improving the availability of parity-encoded RAID systems. The basic idea of the SPP approach is to significantly reduce the recovery overhead of reading data from the survival disks and of the exclusive OR (XOR) calculation for data regeneration, by introducing a supplementary parity mechanism. In addition to the conventional parity units, called *full parity* in this paper, in a RAID4/RAID5/RAID6 disk array, that cover the data units on all of the component data disks for each parity group, our SPP approach stores parity units on dedicated spare disks or potentially free blocks on data disks to cover the data units on a subset of the component data disks for each parity group. As a result, each component disk is protected by a combination of the conventional full parity approach and our SPP approach. By exploiting the unique feature of the XOR calculation, we reduce the overhead of both reading data blocks from the survival disks and XOR calculation for data reconstruction by a factor proportional to the SPP coverage percentage, thus improving both recovery time and user response time during recovery significantly and simultaneously. In other words, in general, with m dedicated spare disks that each provides SPP coverage for one $(m+1)$th of the n component disks, the reconstruction overhead is approximately reduced to $1/(m+1)$ of the original, particularly when n is large. Furthermore, and importantly, an SPP (with the Vertical Orientation, see Section 3 for details) with m spare disks can be shown to tolerate up to $(m+1)$ disk failures.

The proposed SPP approach aims to offer an efficient and flexible mechanism for highly-available RAID systems with minimal complexity and overhead. Similar to the classic RAID architectural level, the SPP architecture has multiple optional levels to meet the various application demands and performance/cost considerations, as detailed in Section 3. The main difference among these levels lies in the coverage and layout of SPP, resulting in different tradeoffs among availability, reliability and cost. More importantly, according to the actual workloads, available storage resource and the availability requirements, the SPP configuration parameters such as the SPP coverage range, placement strategy and update policy etc. can be adaptively and dynamically adjusted on-line due to its highly flexible design.

It must be emphasized that the SPP approach is not a substitution for the full parity mechanism but a powerful supplement to the latter. We argue that SPP is a powerful yet flexible approach in that SPP's deployment does not need any change to the original data or parity layout of the standard parity-encoded disk arrays, and the SPP function of an SPP-enabled disk array can be activated or deactivated adaptively and dynamically while the disk array continues its normal operations. In other words, in the SPP approach free hot-spare disks can be judiciously deployed to generate/update and store the supplementary partial parity units during idle or lightly-loaded periods to significantly improve recovery time and user response time, and to provide enhanced protection against future disk failures.

To prove the concept of SPP and assess its performance, we design and incorporate one common case of the SPP Level-1 in a commonly used software RAID implementation, called Multiple Devices (MD) in Linux, and run extensive trace-driven experiments on this MD implementation. The experimental results demonstrate that SPP improve the reconstruction time by consistently 50% with limited recovery bandwidth, and improve the user response time during recovery by up to 60.42%with limited bandwidth to serving user requests compared with the RAID5 disk array without SPP, and prove that the extra costs for SPP are negligible.

The rest of the paper is organized as follows: background and motivation are presented in Section 2, and Section 3 describes the architecture and design space of the SPP approach. The organization and prototype implementation are presented in Section 4. In Section 5, we conduct result evaluations and detailed analysis. Section 6 presents related work and Section 7 concludes and points out the future work.

2. Background and Motivation

RAID systems have been the essential building blocks of large-scale data centers requiring high performance, high capacity and high availability. Redundant disk arrays with high I/O parallelism, fault-tolerance, and low storage capacity overhead, and parity-encoded RAID levels, such as RAID4, RAID5 and RAID6, are commonly used in data centers.

In general, redundant disk arrays can tolerate one or more disk failures. A redundant disk array operates in one of the following three modes: the *operational mode* when there is no disk failure, the *degraded mode* when one disk drive fails while the disk array still serves the I/O requests with a performance degradation and the risk of data loss, and the *recovery mode* when the disk array is rebuilding data on the failed disk to a replacement disk in the background upon a disk failure. The period when the disk array is in the degraded or recovery mode is called a "window of vulnerability" because another disk failure during this time will cause data loss. After all of the data blocks are rebuilt, the disk array returns to the operational mode.

Therefore, it is very critical to reduce the recovery time to minimize the window of vulnerability and alleviate the performance degradation during recovery. However, most research efforts on RAID systems during the last decade or so have been focused on improving RAID performance, such as solving the small write problem, while relatively less work has been directed towards further improving RAID availability, an increasingly important issue in light of the new reliability challenges brought by large disk capacity and disk array scale.

To seek for a solution to address the availability problem, we make the following important observations about current RAID-structured storage systems based on research investigations and user experiences [13-16]:

1). *Online expansion and reconfiguration lead to reduced availability*. To meet the ever-growing demand for higher capacity and performance, the scale of RAID-structured storage systems has been increasing steadily, where an increasing number of hard disks may be deployed to constitute a single RAID set. Therefore, most existing hardware RAID products, as well as some software RAIDs such as Linux MD, support online addition of new disks to a disk array and the necessary data and parity layout re-organization and re-configuration without interruption of operations. However, the lengthy online reshaping process to expand an existing disk array carries two potential reliability risks. First, during the expansion period there always exists a "window of vulnerability" in which data can be lost given a disk failure or a power down. Second, while the expansion improves the capacity and I/O performance, system reliability and availability decrease because more disks result in a higher disk failure rate. Consequently, the availability of RAID systems does not scale up with the expansion of hard drives, making it very necessary to provide a more efficient and flexible approach to guarantee the availability of large-scale RAID-structured storage systems.

2). *Rapid declines in hard disk costs encourage trading disk capacity and bandwidth for improved system reliability and availability.* With the advancement of the hard disk technology, hard drives are rapidly increasing in their capacity while decreasing in their cost [13]. As a result, the number of spare disks is no longer an issue for a large-scale data center. RAID-structured storage systems usually have multiple available disks as global or local hot spare disks for their multiple RAID sets. It is thus sensible to trade the capacity and bandwidth of these free disks for higher system reliability and availability.

3). *Workload fluctuations provide idle/lightly loaded periods to be leveraged for availability-improving techniques.* Workloads of user applications have been shown to be fluctuating [14, 15]. During the work time, user workloads tend to be heavy while becoming relatively light during the other time. Even during the busy times, there still exist many idle periods between I/O bursts [16]. Leveraging the idle or lightly loaded periods has been a common practice to improve the performance of storage systems.

4). *The salient feature of the XOR operation enables techniques such as SPP to significantly improve availability*. The exclusive OR calculation is widely used in parity-encoded RAID systems due to its unique feature revealed in formulas 1 and 2 below.

$$P_{1\square n} = D_1 \oplus D_2 \oplus \cdots \oplus D_m \oplus D_{m+1} \oplus \cdots \oplus D_n \qquad [1]$$

$$D_m = D_1 \oplus D_2 \oplus \cdots \oplus D_{m-1} \oplus D_{m+1} \oplus \cdots \oplus D_n \oplus P_{1\square n} \qquad [2]$$

If $P_{1\square n}$ is the XOR sum of *N* data units, D_1 to D_n, formula 2 can be used to regenerate any data unit covered by $P_{1\square n}$. The regeneration entails *N* reading operations and *N-1* XOR calculations. Interestingly, given a supplementary partial parity unit as expressed in formula 3,

$$P_{i\square j} = D_i \oplus D_{i+1} \oplus \cdots \oplus D_{j-1} \oplus D_j, \quad 1 \le i \le j \le n \qquad [3]$$

We can easily get $\overline{P_{i\square j}} = P_{1\square n} \oplus P_{i\square j}$ [4]

Actually, $\overline{P_{i\square j}} = D_1 \oplus \cdots \oplus D_{i-1} \oplus D_{j+1} \oplus \cdots \oplus D_n$ [5]

Then, we can get $D_m = \begin{cases} D_i \oplus \cdots \oplus D_{m-1} \oplus D_{m+1} \oplus \cdots \oplus D_j \oplus P_{i\square j}, \forall i \le m \le j \\ D_1 \oplus \cdots \oplus D_{i-1} \oplus D_{j+1} \oplus \cdots \oplus D_n \oplus P_{1\square n} \oplus P_{i\square j}, otherwise \end{cases}$ [6]

It shows that if a supplementary partial parity unit can be obtained, it incurs approximately only *proportional (if j-i=N/2, then N/2)* data reading operations and only *proportional (if j-i=N/2, then N/2)* XOR calculations, that is, the overhead of regenerating the lost data on the failed disk can be nearly halved.

Motivated by the above important observations, we propose the Supplementary Partial Parity (SPP) mechanism that exploits free disk space and free bandwidth during idle or lightly-loaded periods to enhance the existing recovery mechanisms for parity-encoded RAID systems, thus addressing the reliability and availability problems for ultra-scale data centers.

3. The SPP Architecture

3.1 Design Principles

Since the disk failures are unavoidable in large-scale data centers, a new approach is needed to tolerate frequent disk failures with minimal overhead and complexity. Our proposed new approach, the Supplementary Partial Parity (SPP) redundancy protection architecture, is a two-pronged one 1). to minimize reconstruction time and user response time during recovery to reduce the "window of vulnerability"; and 2). to tolerate against multiple disk failures. As an architectural approach to improving availability of existing parity-encoded RAID systems, SPP aims to achieve flexibility, adaptability and scalability, as follows.

1. Flexibility: Since it is risky and costly to expand an existing disk array online as discussed in Section 2, it is desirable not to require any reconfiguration of data or parity layout in standard RAID levels when incorporating SPP into an existing parity-encoded RAID system.

2. Adaptability: It will be advantageous for SPP to be activated, deactivated and adjusted adaptively and dynamically in order to target an optimal performance/cost ratio in terms of various QoS requirements and resource investments.

3. Scalability: The availability improvement by the SPP approach should be scalable to the size of the redundant disk arrays and the available system resource. In other words, if the scale of the disk arrays grows up, the improvement should increase accordingly. If more free storage space and idle times are available, the SPP approach should improve the availability proportionally to the resource utilization.

SPP exploits free storage space and idle times to store and update a set of partial parity units that cover a subset of disks (or data stripe units) during failure-free and idle/lightly-loaded periods, thus supplementing the existing full parity units for improved availability. By leveraging the XOR operations appropriately among partial parity, full parity and data units (see Section 3.2 for details), SPP can reconstruct data

on the failed disks with only a fraction of the original overhead that is proportional to the partial parity coverage, thus significantly reducing the overhead of data regeneration especially under heavy workload.

3.2 The Basic Idea and Simple Examples of SPP

Here is a general but simple example to illustrate the basic idea of SPP, including its SPP layout, coverage and functionality. In Figure 1, $P_{m\text{-}n}$ denotes a full parity stripe covering data stripes from D_m to D_n. $Q_{i\text{-}j}$ denotes a partial parity stripe covering data stripes from D_i to D_j while $Q_{P,k}$ denotes a partial parity stripe covering the data stripe D_k and the parity stripe P for this parity group. A shadowed rectangle with the dashed line denotes the coverage of Q for this parity group.

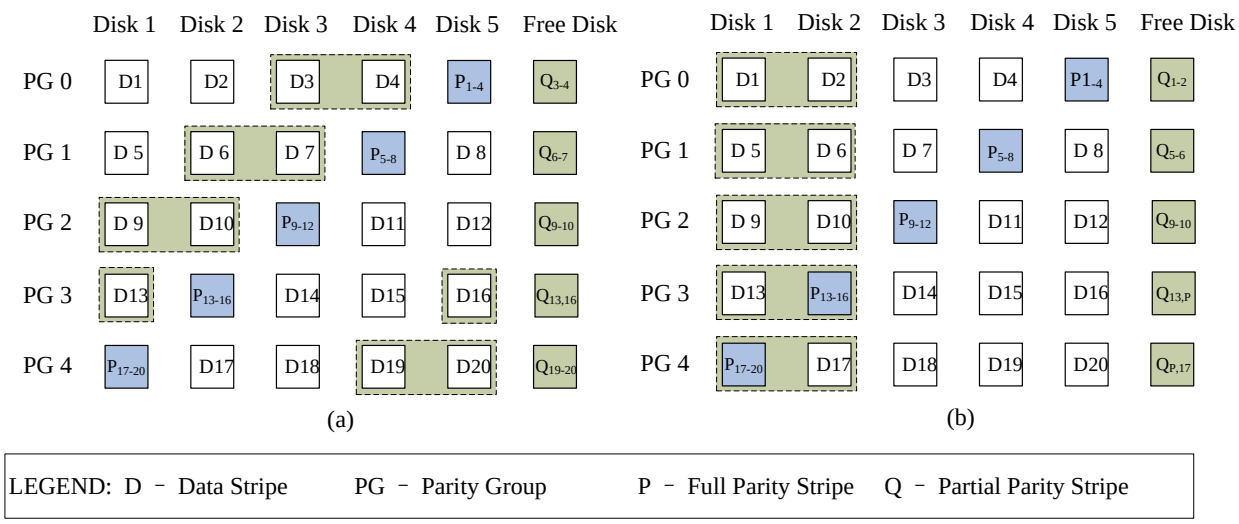


Figure 1. Two examples of SPP coverage and layout. In Figures 1(a) and 1(b), SPP utilizes one free disk to store the partial parity stripes, and each partial parity stripe covers data on half of the disks for each parity group. Figure 1(a) depicts an SPP coverage with *Diagonal Orientation* and Figure 1(b) depicts an SPP coverage with *Vertical Orientation*.

As Figure 1(a) shows, given a RAID5 left-asymmetry disk array consisting of five disks with a free hot-spare disk, SPP uses this spare disk to store the partial parity stripes. More specifically, the SPP stripes of $Q_{3\text{-}4}$, $Q_{6\text{-}7}$, $Q_{9\text{-}10}$, $Q_{13,16}$ and $Q_{19\text{-}20}$ are calculated as follows:

$$Q_{3\text{-}4} = D_3 \oplus D_4;\ \ Q_{6\text{-}7} = D_6 \oplus D_7;\ \ Q_{9\text{-}10} = D_9 \oplus D_{10};\ \ Q_{13,16} = D_{13} \oplus D_{16};\ \text{and}\ \ Q_{19\text{-}20} = D_{19} \oplus D_{20}$$

Assume that one disk, say Disk 4, fails at one time. According to the conventional full parity approach, D_4, $P_{5\text{-}8}$, D_{11}, D_{15} and D_{19} on Disk 4 will be regenerated as follows:

$$D_4 = D_1 \oplus D_2 \oplus D_3 \oplus P_{1\text{-}4};\quad P_{5\text{-}8} = D_5 \oplus D_6 \oplus D_7 \oplus D_8;\quad D_{11} = D_9 \oplus D_{10} \oplus P_{9\text{-}12} \oplus D_{12};$$
$$D_{15} = D_{13} \oplus P_{13\text{-}16} \oplus D_{14} \oplus D_{16};\ \text{and}\ \ D_{19} = P_{17\text{-}20} \oplus D_{17} \oplus D_{18} \oplus D_{20}$$

However, if the SPP approach is deployed, we can regenerate these data or parity stripes in a more efficient way as follows:

$D_4 = D_3 \oplus Q_{3\square 4}$; $P_{5\square 8} = D_5 \oplus D_8 \oplus Q_{6\square 7}$; $D_{11} = P_{9\square 12} \oplus D_{12} \oplus Q_{9\square 10}$; $D_{15} = P_{13\square 16} \oplus D_{14} \oplus Q_{13,16}$; and $D_{19} = D_{20} \oplus Q_{19\square 20}$

Intuitively, compared with the conventional full parity approach, SPP in this example can almost halve the overhead due to data read operations and XOR calculations for data reconstruction on average. Although reads issued to individual disks for data reconstruction are always executed in parallel, the recovery bandwidth of each disk quickly becomes a bottleneck while users' requests are being served at the same time, since a disk array utilizes most of the available bandwidth to guarantee services to users' requests [17]. Moreover, the maximal I/O latency among read operations is bounded by the slowest read operation. Thus, SPP has on average a 50% chance of avoiding the slowest read operation, compared with the full parity approach. On the other hand, because SPP halves the number of the read operations, it also avoids the negative performance impact of data reconstruction (reads) on disks that are spared of the involvement in recovery, which is particularly important for user requests under heavy workloads. As a result, SPP can reduce disk bandwidth utilization due to reconstruction, shorten disk I/O queues, mitigate system bus bottlenecks, and lower CPU utilization during failure recovery. This will be specially pronounced under heavy I/O workloads.

Figure1 (b) illustrates another SPP approach with a different coverage orientation of partial-parity. According to the coverage orientation, there are two forms of partial-parity distribution: partial-parity with *Diagonal Orientation* (as shown in Figure 1(a)) and partial-parity with *Vertical Orientation* (as shown in Figure 1(b)). Diagonal Orientation implies that data stripes covered by SPP are distributed diagonally; while Vertical Orientation signifies that stripes covered by SPP are distributed among a fixed subset of disks.

Assume that one disk, say Disk 4, fails at one time, we can regenerate these data or parity stripes as follows:

$D_4 = D_3 \oplus P_{1\square 4} \oplus Q_{1\square 2}$; $P_{5\square 8} = D_7 \oplus D_8 \oplus Q_{5\square 6}$; $D_{11} = P_{9\square 12} \oplus D_{12} \oplus Q_{9\square 10}$; $D_{15} = D_{14} \oplus D_{16} \oplus Q_{13,\mathrm{P}}$; and $D_{19} = D_{18} \oplus D_{20} \oplus Q_{\mathrm{P},17}$

Assume that another disk, say Disk 2, subsequently fails, we can regenerate these data or parity stripes as follows:

$D_2 = D_1 \oplus Q_{1\square 2}$; $D_6 = D_5 \oplus Q_{5\square 6}$; $D_{10} = D_9 \oplus Q_{9\square 10}$; $D_{13\square 16} = D_{13} \oplus Q_{13,\mathrm{P}}$; and $D_{17} = P_{17\square 20} \oplus Q_{\mathrm{P},17}$

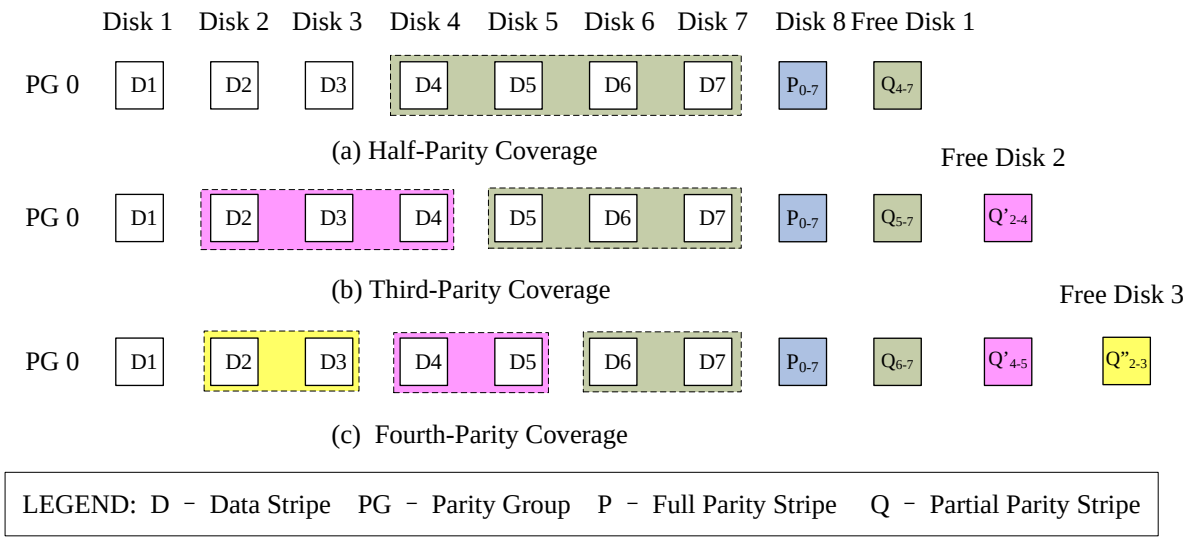
The advantage of Diagonal Orientation is its ability to balance recovery workload among all of the disks, but at the cost of not being able to tolerate a subsequent disk failure during recovery. On the other hand, Vertical Orientation can tolerate another disk failure during recovery if exactly one of the two failed disks is covered by SPP, a fault-tolerant ability that is equivalent to that of RAID50 [18]. The drawback of this distribution lies in the imbalanced recovery workload. An additional advantage of Vertical Orientation is its potential for covering a number of designated disks that may have higher failure rates.

3.3 Design Space of the SPP Architecture

As illustrated in the example of Figure 2, the SPP architecture has a rich design space along a number of dimensions, which we will discuss in more details in this subsection. In general, in addition to the cover-

age orientation (diagonal vs. vertical) discussed in Section 3.2, the main design space of SPP can span along the following dimensions: coverage range, placement strategy, and update policy.

SPP Coverage Range refers to the proportion of the component disks in a parity group in the disk array that are covered by a SPP parity group. In the *Half-Parity* approach (Figure 2(a)), for example, the cov



LEGEND: D – Data Stripe PG – Parity Group P – Full Parity Stripe Q – Partial Parity Stripe

Figure 2. An example of SPP with different coverage ranges. For simplicity, only the first parity group for every coverage range is illustrated.

erage range is ½ since each SPP parity group covers the data on half of the component disks. In terms of the available storage space and system availability requirements, SPP provides a family of optional approaches with different coverage ranges. As Figure 2(b) depicts, the *Third-Parity* approach exploits two free spare disks to store partial parity for two sets of SPP stripes, with each SPP parity group exclusively covering stripes on one third of the component disks. Third-Parity reduces the overhead for data regeneration to nearly one third of that required by the full parity approach. Similarly, the *Fourth-Parity* approach exploits three spare disks to store partial parity for three sets of SPP stripes, with each SPP parity group exclusively covering stripes on one fourth of the component disks. Correspondingly, the overhead of Fourth-Parity is decreased to nearly one fourth of the overhead required by the full parity approach. Additionally and in general, if the Vertical Orientation coverage distribution is applied, an nth*-Parity* approach can tolerate up to *n* simultaneous disk failures.

More attractively, the coverage range can be flexibly setup by utilizing workload characteristics or file system semantic knowledge. For example, a SPP parity group need only cover the infrequently-accessed stripes to reduce SPP update overhead if a monitor of user access pattern is integrated into the SPP approach. Or, if the liveness information of the data stripes is obtained by methods similar to those described in TSD [19] or SDS [20], a SPP parity group need only cover the live stripes to further reduce reconstruction overhead.

SPP Placement Strategy. As Figures 1 and 2 indicate, our SPP approach utilizes one or more dedicated free spare disks to store the SPP parity units. On the other hand, if the liveness information of data stripes in disk arrays can be obtained, SPP can exploit “free blocks”, disk blocks that are deemed “dead” from the perspective of file systems or databases [21], to store the SPP parity units, thus minimizing the extra storage overhead to zero. Since it has been shown that the proportion of the free capacity over the whole

capacity of a storage system deployed in a typical application environment is about 50% [22], there is in general sufficient amount of room for SPP to leverage additional "free blocks" in a disk array to store one or multiple replicas of the SPP stripes to mitigate the negative impact on the validity of SPP parity stripes, in the possible events that free blocks containing the SPP parity units are overwritten with new data by the upper file systems or databases.

SPP Update Policy. When a write request arrives at a disk array with an address that does not fall under the coverage of any SPP parity group, no update is needed to any SPP parity unit. Otherwise, the SPP parity unit of the SPP parity group covering the address of the write request needs to be updated somehow.

SPP provides two update policies: *synchronous update* and *asynchronous update*. The synchronous update policy, which applies update to the corresponding SPP parity stripe at the same time as the write operation, ensures the full validity for each SPP parity group, but incurs performance degradation with write-intensive workloads. However, it may be practically acceptable since write intensity is generally much lower than read intensity and writes tend to congregate around a relatively small proportion of the storage capacity in typical workloads [16]. The advantage of the asynchronous update policy, which postpones updates to parity stripes until idle or lightly-loaded periods, is its ability to minimize performance degradation due to frequent SPP updates. However, the asynchronous update policy may reduce the benefit of the SPP approach during recovery if some SPP parity stripes are invalid (not updated yet) at the time of recovery. In general, the amount of such decrease in SPP benefit will be proportional to the amount of invalid SPP parity stripes. Another drawback of the asynchronous update policy is its need to use some NVRAM or other types of battery-powered memory to store the SPP verification bitmap table (see Section 4 for details), although the cost of such NVRAM can be negligible in modern RAID systems with massive memory capacity.

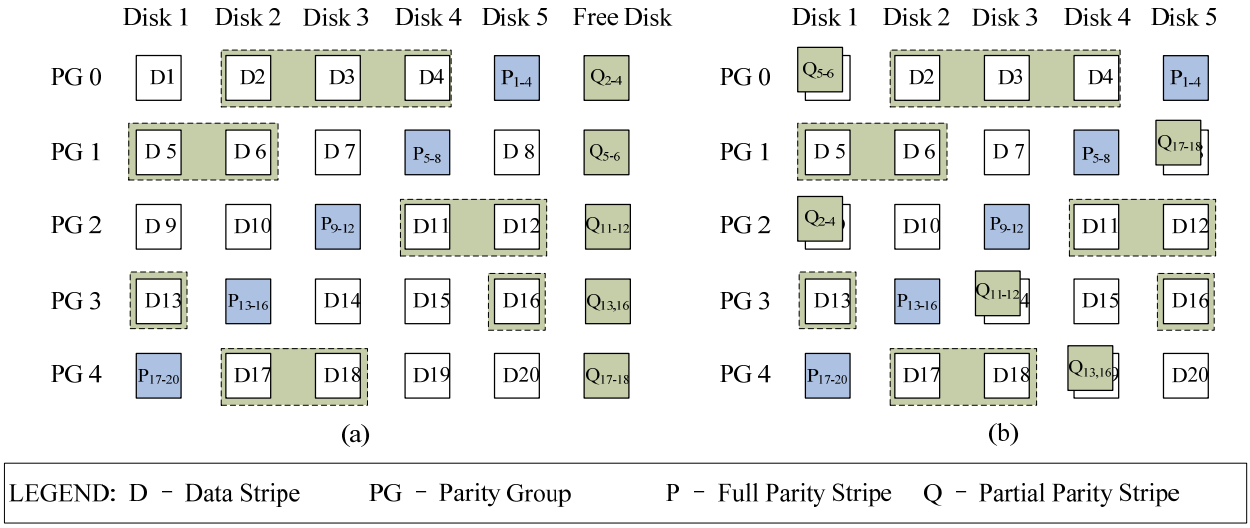


Figure 3. The coverage examples for the SPP Level 2 and SPP Level 3. Figure 3(a) depicts the coverage and layout for the SPP Level 2, in which each partial parity stripe only covers infrequently-accessed data stripes for each parity group. Figure 3(b) depicts the coverage and layout for the SPP Level 3, in which SPP utilizes free blocks on the component disks to store partial parity stripes.

Based on the SPP design space defined and discussed above, we define three SPP levels, analogous to the typical RAID levels, as follows:

SPP Level 1 is the basic level of SPP, whose salient feature is to trade the lowest implementation complexity and minimal overhead for a reasonable availability improvement. SPP Level 1 includes Half-Parity, Third-Parity, Fourth-Parity, etc. These approaches utilize one or multiple dedicated spare disks to store SPP parity stripes. It is easy to determine the coverage of SPP parity groups due to the fixed and simple coverage range algorithm. This level can be easily implemented in most existing parity-encoded disk arrays.

SPP Level 2, as a flexible extension of SPP Level 1, includes those approaches that have flexible coverage range and can thus cover infrequently-accessed stripes according to workload characteristics. The main advantage of SPP Level 2 is its high flexibility, which comes at the expense of the need for an access popularity monitor to keep track of the changing user accesses and a bitmap table to represent the status of whether a stripe is covered by a particular SPP parity group. As shown in Figure 3(a), the coverage range becomes more flexible since the number and locations of data stripes covered by the SPP stripes are no longer fixed as in SPP Level 1.

The main objective of **SPP Level 3** is to incorporate semantic knowledge into the recovery process. By exploiting the liveness information of the data stripes from the perspective of file systems or databases, SPP Level 3 does not need dedicated spare disks to store SPP parity units because it can leverage the free blocks in the component disks to store the SPP parity units and their replicas. Another advantage is that it only needs to generate and update the SPP stripes for live data blocks. However, since this level needs the semantic knowledge of file system or databases running on a RAID system, it incurs more implementation complexity and sacrifices some portability. In Figure 3(b), SPP stores the SPP stripes onto the free blocks onto the component disks needless dedicated spare disks. Table 1, below, summarizes the characteristics of the suggested three optional SPP levels based on the main parameters of the SPP design space.

SPP Level	Coverage Range	Placement Strategy	Update Policy	Flexibility	Adaptability	Scalability	Storage Overhead
Level 1	Fixed	Dedicated Disks	Both	Moderate	Strongest	Strong	Small
Level 2	Flexible	Dedicated Disks	Both	Strong	Strong	Strong	Small
Level 3	Most Flexible	Free Blocks	Both	Strong	Moderate	Strong	Zero

Table 1. The characteristics of three SPP Levels.

4. Organization and Prototype Implementation

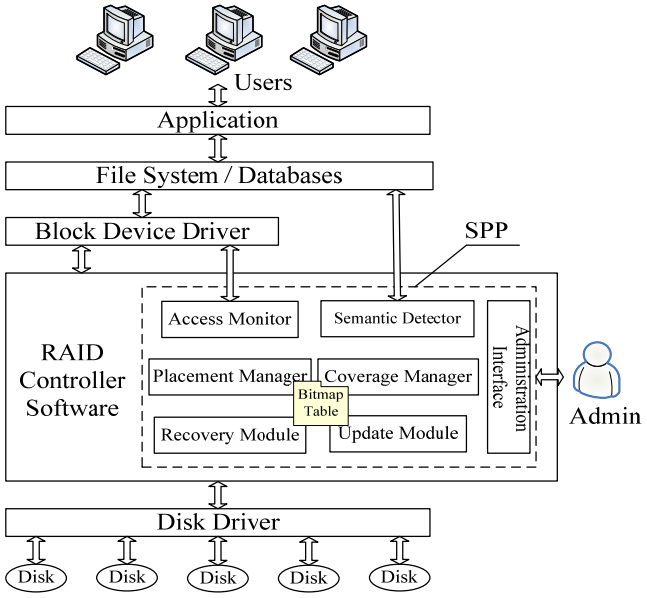


Figure 4. SPP organization and its functional modules.

The entire SPP structure can be embedded into any RAID controller software, so it is independent of the underlying disks or storage media. As a result, our SPP approach can be easily ported onto any of the various platforms based on block-level devices, such as Network Block Device, Solid State Disk, flash disk, etc.

Basically, the SPP organization is composed of eight main functional components: Administration Interface, Access Monitor, Semantic Extractor, Coverage Manager, Placement Manager, Update Module, Recovery Module, and SPP Bitmap Table, as shown in Figure 4. **Administration Interface** provides an interface for storage system administrators to configure the parameters of the SPP design space and monitor operational states. **Access Monitor** is responsible for keeping track of the frequency of I/O requests to help **Update Module** determine the appropriate time to start updating SPP stripes, and keep track of the changing popularity of workloads to help Coverage Manager determine the appropriate coverage range for SPP Level 2. The main functions of **Semantic Extractor** include two parts: (1) to discover which blocks are alive to help Coverage Manager choose an appropriate coverage range covering only the live data stripes for each parity group; and (2) to discover which blocks are free to help the Placement Manager choose the appropriate location to store the SPP parity stripes and their replicas for SPP Level 3. **Placement Manager** is responsible for selecting and executing a placement strategy while **Coverage Manager** is responsible for choosing and executing a coverage strategy for SPP. In the **SPP bitmap table**, one bit is used to denote the validity status of the SPP parity stripe for each parity group, with a value of '0' (false) indicating an invalid status and a value of '1' (true) signifying a valid status. The SPP **Recovery Module** is responsible for leveraging both the partial parity and full parity mechanisms to rebuild the data or parity stripes to the replacement disk upon a disk failure. Once a disk fails, the SPP recovery process will be activated automatically and data reconstruction will be launched in the background. For each parity group, the SPP bitmap table will be checked first. If the corresponding bit is true, it will rebuild the data on the failed disk according to the specific SPP approach. Otherwise, it will rebuild ac-

cording to the full parity approach. The main function of the SPP Update Module is to generate the partial parity when deployed and update them appropriately based on the designated SPP update policy.

As shown in Figure 5, the SPP approach can be activated or de-activated at any given time, even for an online RAID system. An administrator can activate the SPP approach via the Administration Interface and inform the SPP module of the detailed SPP configurations (such as SPP Level, available spare disk number, SPP coverage range and update policy, etc.). Given a specific SPP configuration, other appropriate SPP components will be activated and the bitmap table will be initialized. After the initialization, the Update Module will start to generate the SPP parity stripes for each parity group, and store and update them on the free storage space during idle or lightly-loaded periods under the control of the Access Monitor, Semantic Extractor, Coverage Manager and Placement Manage by default (i.e., asynchronous update). Periodically, the SPP update process checks the SPP bitmap table to identify and update the invalid SPP stripes during the idle periods. Once a disk fails, the SPP Recovery Module will start a recovery process in the background and rebuild the data on the failed disks automatically until all of the stripes have been rebuilt on the replacement disk. Due to the nature of supplementary partial parity, the SPP module can dynamically adjust the appropriate strategies and policies to achieve the target of optimal performance/cost ratio according to available resources and the administrator's requirements.

In our current proof-of-the-concept prototype SPP implementation, we incorporate the basic SPP architecture and organization into the Linux software RAID (MD) and its corresponding management software (mdadm). Due to the time and space constrains, we implement one common case of SPP Level 1, Half-Parity, for the study reported in this paper, while the implementation and performance evaluation of

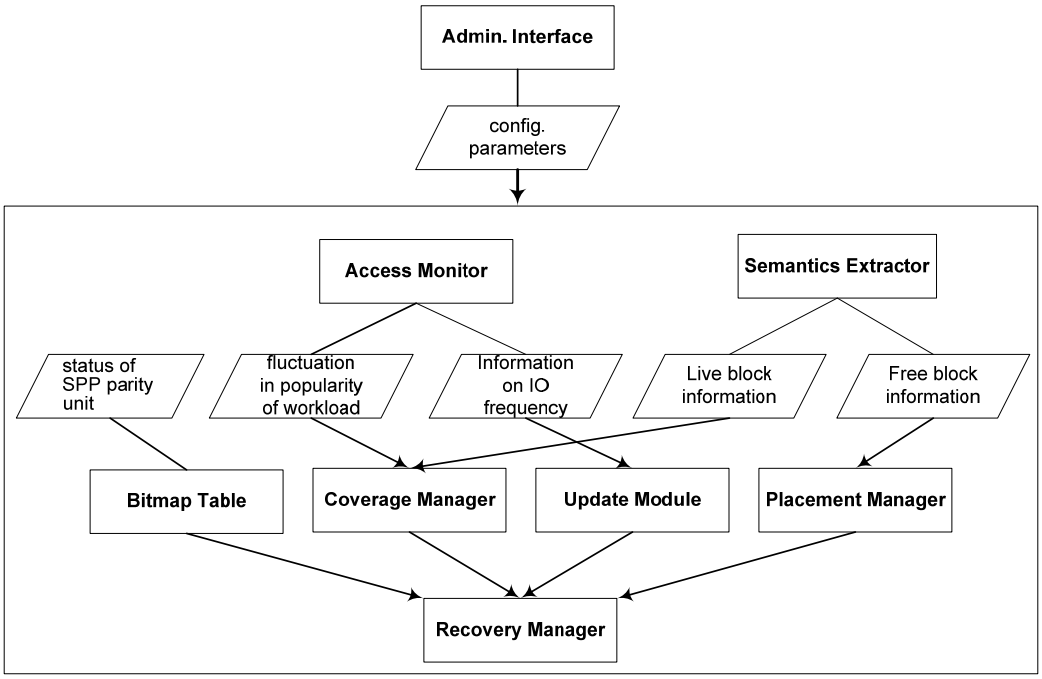


Figure 5. SPP workflow.

SPP Level 2 and 3, which are beyond the scope of this paper, are left for future research. We implement all the SPP components except for the Semantic Extractor because Half-Parity does not need the semantic information of file systems or databases. For the flexibility of our experiments, we integrate an IOCTL function into mdadm to trigger the SPP update process manually.

The source code augmentation on MD mainly concentrates on the SPP coverage, placement, update modules and the key modification made to MD lies in its recovery module. All of the augmented and modified source code amounts to no more than 2000 lines of C code.

5. Performance Evaluations

In this section, we report and analyze the experimental results from our extensive trace-driven experiments conducted on the SPP Level 1 prototype, Half-Parity (HP for short). There are generally two common evaluation metrics for system availability and user performance of a recovery process: reconstruction time and average user response time during recovery. We will present performance evaluation of SPP with regard to the various RAID configurations, such the number of the component disks, RAID level, RAID chunk size, reconstruction bandwidth thresholds, etc., and typical workloads.

5.1 Trace-Driven Experiment Setup

All of the experiments are conducted on two different server-class machines but with the same software configurations. Machine A has an Intel 3GHz Xeon processor, 512 MB DDRAM, and 15 Seagate ST3300831AS 300GB SATA hard drives attached by a Highpoint RocketRAID 2240 SATA adapter. Machine B has an AMD Opteron Dual-Core 2.2GHz Processor, 2GB DDRAM, and 15 Western Digital WD1600AAJS-08PSA 160G SATA hard drives attached by a Marvell 88SX50XX SATA adapter. The majority of the experimental results reported here are obtained from Machine B unless explicitly specified otherwise. We install the Fedora Core 4 Linux (kernel version 2.6.11) on the machines that have the Linux software RAID package (MD and mdadm) embedded by default. It must be noted that there is a mechanism in MD to control the disk bandwidth utilized by the recovery process to guarantee user performance. The default minimal and maximal thresholds are 1MB/s and 200MB/s, respectively, which means that MD will use a minimal of 1MB/s disk bandwidth for recovery upon heavy user loads and any available bandwidth for recovery upon light user loads since the highest bandwidth of off-the-shelf disks is far less than 200MB/s.

We use RAIDmeter [17] as a block-level trace-replay software that issues specific I/O requests to the storage device at the specific times according to the trace records and collects the response time per I/O request during failure recovery. Two typical traces [23] identified by the Storage Performance Council [24], Websearch (Web for short) and Financial (Fin for short), are used in our performance evaluations. The former was obtained from a machine running a web search engine. The latter was captured from OLTP applications run by a large financial institution. In addition, to avoid the time-consuming recovery process, we limit the capacity of each disk to 10GB in the experiments and truncate the beginning 2000000 recorders of both the traces. We note that this limit on disk capacity should not significantly affect the comparative evaluation results, nor the main conclusions from the study. Regarding both captured traces, Web has a 99.98%:0.02% read/write ratio with 331.05 IOPS intensity, while Fin has about a 21.55%:78.45% read/write ratio with 55.36 IOPS intensity. We use Web to represent heavy workload while we intentionally double the intensity of Fin to represent moderate workload since the original intensity of Fin is too light. To scale the workloads to the growing number of disks in the experiments, the coverage of the requests is increased proportionally to the number of disks.

5.2 Numerical Results

We conduct experiments to evaluate the performance of the Half-Parity (HP) SPP configuration with the Diagonal Orientation on a RAID5 left-asymmetric disk array composed of a variable number of component disks and one single hot-spare disk with a fixed chunk size of 64KB. We incorporate one dedicated disk to store the SPP parity stripes, and all SPP parity stripes have been updated and thus valid at the time of recovery. Table 2 and Table 3 respectively show the reconstruction times and average user response times of RAID5 with and without HP under the default MD recovery bandwidth threshold range of 1MB/s to 200MB/s, as a function of the number of disks. From Table 2 one can see that HP improves the reconstruction time consistently by about 50% with the Web trace and by up to 37.78% with the Fin trace. From Table 3 one can see that HP incurs an average of 5.99% and a maximum of 15.20% performance degradation on the average response time compared with the architecture without HP. The reason why the reconstruction time improvement is far greater than the average response time improvement is that MD utilizes a very small preserved disk bandwidth for recovery by default (1 MB/s in this case) while availing the remainder of all available bandwidth to serve user I/O requests. Under heavy workloads, the preserved recovery bandwidth will decease to the minimal threshold to guarantee the user performance. In this scenario, the recovery bandwidth becomes a severe bottleneck. As shown in Table 2, the recovery rate without HP is roughly 10GB / 10000sec = 1MB/s while the recovery rate with HP is roughly 10GB / 5000sec = 2MB/s for the heavy Web trace. In other words, our HP approach succeeds in doubling data regeneration rate of the full parity approach under a heavy workload. The moderate Fin trace, however, only resulted in about 30% reduction in reconstruction time, further validating our intuition that SPP's effectiveness is most pronounced under heavy workloads. The reason why the average user response time of HP slightly suffers is two-fold. First, our current prototype implementation favors reconstruction time improvement over user response time improvement by allocating all the saved bandwidth by SPP back to the recovery process, resulting in user requests not benefiting from any of the reduced data regeneration overhead. For example, the recovery rate for the Web trace is doubled in HP because the recovery process in HP reaps all the benefit of halving the overhead of data regeneration. Second, for write-intensive applications, such as the Fin trace, the replacement disk becomes a performance bottleneck since HP has a higher reconstruction write rate to the replacement disk than the non-HP scheme, which causes a slight slowdown of the user write requests that are issued to the replacement disk (as shown in Table 3). However, the policy of whether to favor reconstruction or user requests can be tunable to strike a good balance between reliability and user performance during recovery. As a result, the user response time can be significantly improved if one chooses to allocate all the benefit from reducing data regeneration overhead to serve user requests without increasing the reconstruction time.

RAID Level	**Number of Disks**	**Reconstruction Time(second)**					
		Web			**Fin**		
		w/o HP	**w/ HP**	**improved**	**w/o HP**	**w/ HP**	**improved**
RAID5	9	10411.90	5193.99	50.11%	2105.82	1715.75	18.52%
	11	10388.83	5199.64	49.95%	3212.91	2199.77	31.53%
	13	10377.48	5210.95	49.79%	3866.23	2405.72	37.78%

Table 2. A comparison of reconstruction time with HP and without HP as a function of the number of disks.

RAID Level	**Number of**	**Average User Response Time during recovery (millisecond)**	
		Web	**Fin**

	Disks	w/o HP	w/ HP	improved	w/o HP	w/ HP	improved
RAID5	9	30.46	30.76	-0.98%	86.24	87.35	-1.29%
	11	30.47	30.96	-1.61%	66.62	76.74	-15.20%
	13	30.49	32.34	-6.07%	72.06	79.85	-10.81%

Table 3. A comparison of user response time with HP and without HP as a function of the number of disks.

It must be noted from the above results that SPP is most effective in alleviating the impact of performance bottleneck. Since MD is an availability-oriented RAID system, the default minimum recovery bandwidth threshold of 1MB/s imposes a performance bottleneck for the recovery process in favor of user requests and our SPP is able to effectively alleviate the bottleneck by reducing reconstruction time significantly. On the other hand, as shown in our results given later in the paper, the user response time can also be significantly reduced by SPP if the performance bottleneck is shifted onto the user request service by limiting the bandwidth available to serving user requests. SPP can be adjusted to provide performance improvement for both reconstruction time and user response time simultaneously.

To examine the impact by the different chunk sizes, we conduct experiments on a RAID5 disk array composed of 13 disks and one single hot spare disk with variable chunk sizes of 4KB, 64KB and 256KB. The experimental results show that the measured reconstruction times and average response times of HP remain almost unchanged, demonstrating that the Half-Parity approach is not sensitive to change in chunk size.

Similarly, we conduct experiments on the platform of a disk array consisting of 11 disks and one spare disk with variable RAID levels of RAID4 and RAID5 to examine the performance impact on HP by the different RAID levels. We plot the measured reconstruction time and average response time during recovery in the Figure 6. From Figure 6, one can see that the reliability and performance impacts of HP on a RAID4 disk array are similar to those on a RAID5 disk array. With HP, reconstruction time has been reduced nearly by half while sacrificing very small user response time degradation under the default MD reconstruction bandwidth threshold of 1MB/s.

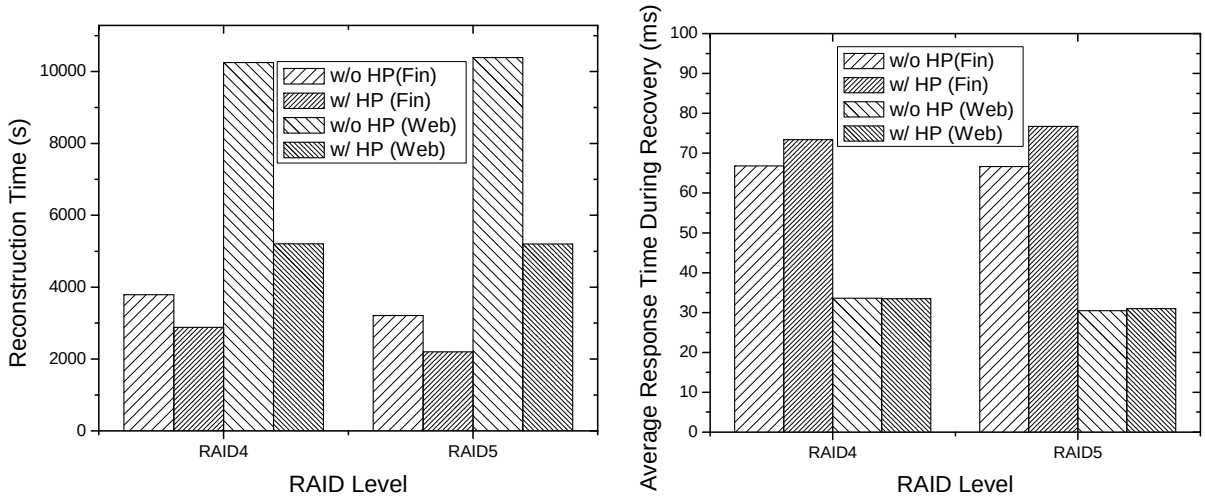


Figure 6. Reconstruction time (in seconds) with two different RAID levels (RAID4 and RAID5), under two traces: Fin and Web, comparing the architecture without and with HP. The RAID4/RAID5 disk array is consisted of 11 disks and one spare disk with 64KB chunk size.

Since the preserved bandwidth for recovery has direct impact on the recovery performance and user performance, we conduct experiments to evaluate performance impacts caused by the different minimal reconstruction bandwidth threshold for recovery. By adjusting the default minimal threshold from 1MB/s to

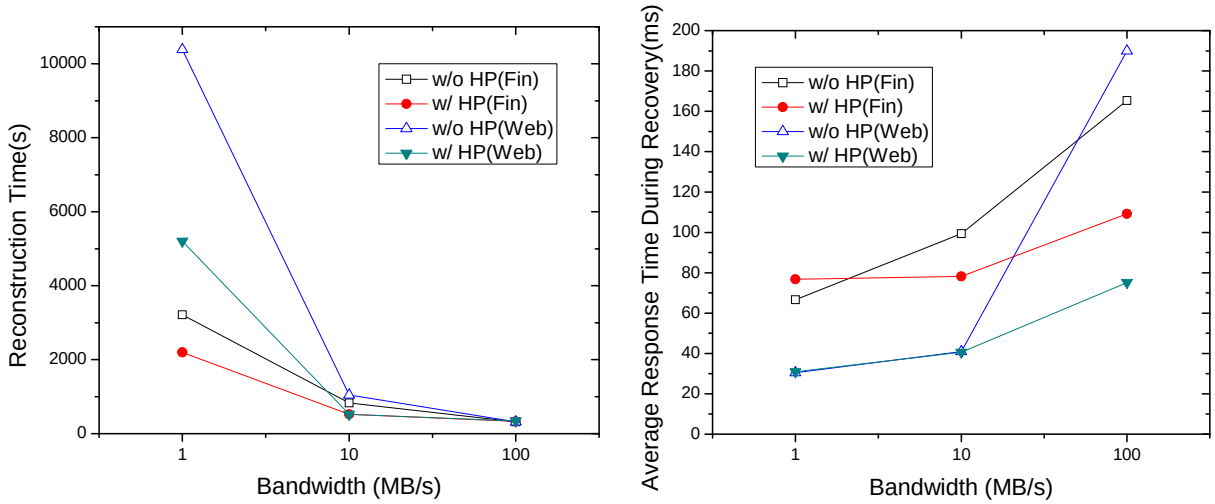


Figure 7. Reconstruction time (in seconds) and average response time (in milliseconds) during recovery under different recovery bandwidth (1MB/s, 10MB/s, 100MB/s), under two traces: Fin and Web, comparing the architecture without and with HP.

10MB/s and then to 100MB/s, respectively, Figure 7 plots the measured reconstruction times and average response times during recovery on a RAID5 disk array consisting of 11 disks and one hot spare disk. It is observed that as the threshold increases, the user response time is consistently lengthened while the reconstruction time is consistently shortened because more disk bandwidth is allocated for the recovery process for both the conventional approach and our SPP approach. However, as the threshold shifts from one extreme (i.e., 1MB/s) to the other (i.e., 100MB/s), the SPP improvement over the original full-parity approach also shifts from reconstruction time (Figure 7(a)) to user response time (Figure 7(b)), reducing the user response time by up to 60.42% for Web and 33.93% for Fin. It demonstrates that our SPP approach can significantly alleviate the bandwidth bottleneck under heavy workloads. HP shows great advantages in reconstruction time with lower recovery bandwidth, which implies that HP is especially desirable under heavy workload. It is noted that setting 10MB/s as a minimal bandwidth threshold for recovery may be a reasonably good reliability/performance tradeoff for the Fin trace because HP improves both reconstruction time and average response time simultaneously.

To examine the performance impact by different coverage orientations, Diagonal Orientation vs. Vertical Orientation, we conduct experiments on an HP-enabled RAID5 disk array with both orientations. Figure 8 illustrates that the reconstruction time and response time improvements with the Vertical Orientation are less than those with the Diagonal Orientation, especially for the Fin trace. This is because that the Vertical Orientation covers a fixed subset of all component disks and distributes all SPP recovery workloads onto this subset, thus imposing imbalanced workloads on the disk array during recovery. As a result, the reconstruction time and user response time during recovery are lengthened compared with the Diagonal

Orientation that distributes SPP recovery work evenly among all component disks. In other words, the Vertical Orientation trades a small performance loss for its multiple-failure-tolerance capability.

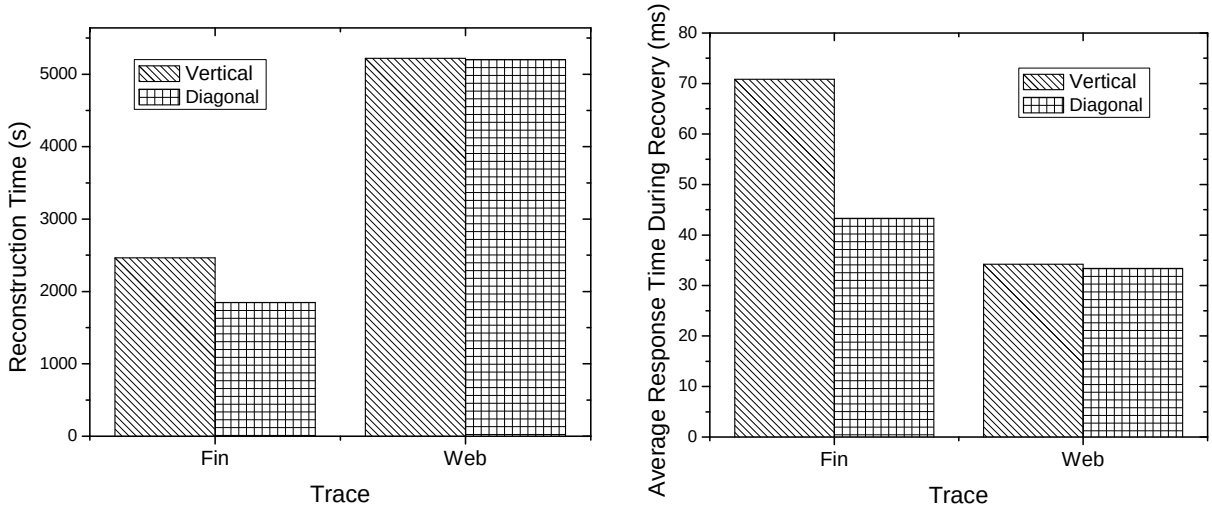


Figure 8. Reconstruction time (in seconds) and average response time (in milliseconds) during recovery with Diagonal and Vertical Orientation of SPP architecture, under Fin and Web traces. This experiment is conducted on Machine A, and the disk array is consisted of 13 disks and one spare disk.

5.3 Discussions

In summary, our trace-driven experiments have shown that the SPP approach can significantly improve the reconstruction time and average response time during recovery, especially under heavy workloads. In addition, we draw the following the additional observations and conclusions from our experimental study:

1) Reconstruction time with our Half-Parity approach has been reduced consistently by about 50% compared with the conventional parity-based approach, under relatively heavy user I/O loads (Web). Also, comparing the results with the two traces used in this study, SPP is more desirable under a heavy IO workload. It indicates that the reconstruction time improvement under heavy workloads is roughly proportional to the coverage range of SPP, so other SPP approaches such as Third-Parity or Fourth-Parity will likely produce far greater performance improvement if more available spare disks are introduced. It shows that our SPP approach is scalable with the intensity of workloads and available resources.

2) With limited recovery bandwidth, SPP shows pronounced effect in reducing reconstruction time. On the other hand, SPP shows pronounced effect in reducing user response time during recovery when the disk bandwidth to serve user requests becomes a bottleneck. More importantly, we can judiciously select a reasonable threshold for the minimal recovery bandwidth to improve the reconstruction time and response time simultaneously, as a good tradeoff between system reliability and user performance. The SPP design provides this flexibility for system administrators to balance system reliability and user performance requirements.

3) The experimental results show that while the SPP approach is insensitive to the RAID level and chunk size, it is very sensitive to the minimal recovery bandwidth threshold and the intensity of workloads. Moreover, our investigation into the performance impacts of different coverage orientations and the

number of component disks has confirmed the efficiency of SPP. The experimental results also demonstrate the adaptability of the SPP approach since SPP can achieve different performance/cost ratios to target different reliability and performance requirements by judiciously tuning a number of design parameters in the SPP design space.

6. Related Work

Because our approach is supplementary to the conventional parity-encoded RAID solution, we will concentrate on the most representative and relevant work on the existing parity-encoded RAID organization and recovery mechanisms.

Among these existing mechanisms, AFRAID [25] and VSP [26] are the most relevant approaches to our SPP approach. The essential idea of AFRAID is to achieve a significantly improved performance (similar to RAID0 in the best case) by sacrificing a small amount of data redundancy. AFRAID eliminates the small update penalty by delaying data update until the idle period of users' workload. AFRAID regulates the parity update policy to allow a smooth tradeoff between performance and availability. By comparison, our SPP approach improves system reliability and availability by supplementing extra yet efficient data redundancy without sacrificing performance. More importantly, our SPP approach provides extra redundancy protection, in addition to the existing full-parity redundancy, instead of the frequent-but-non-guaranteed redundancy provided by AFRAID, meaning that the AFRAID approach may result in data loss because full parity update is postponed to a later time.

On the other hand, VSP presents an extension to the RAID5 architecture by using a variable scope protection, in which full parity units are maintained for arbitrary subsets of the data blocks in each stripe. The VSP parity only protects the in-use data blocks. However VSP does not specify how to differentiate data blocks that are in-use. While SPP also covers variable subsets of data units for each parity group, it differs from VSP in two important ways. First, SPP augments a parity-encoded RAID by providing a supplementary partial parity protection that can be updated asynchronously. Second, the SPP Level-1 and Level-2 (refer to Section 3 for more details) do not need to obtain the liveness information of the data units, so they can be easily integrated with any existing hardware or software RAID systems.

To some extent, the SPP approach leverages the advantages of both AFRAID and VSP while avoiding their drawbacks. By exploiting the unique feature of XOR, SPP greatly improves reconstruction time and user response time during recovery, in addition to achieving higher failure tolerance.

Parity Declustering [27, 28] reduces the additional load on survival disks during recovery by distributing data and parity groups over a larger number of disks, thus balancing cost against reliability and performance during recovery. To address the "read-modify-write" problem in parity encoded disk arrays, the Parity Logging approach [29] applies journaling techniques to log the parity update record, while the Floating Parity method [30] updates data and parity units to free blocks that are near the disk heads, thus improving the small write performance during the fault-free operational mode. Both of these two approaches favor performance during the operational mode over availability, but our SPP favors availability without sacrificing performance during the recovery mode.

Bachmat et al proposed a greedy reconstruction algorithm [31] to exploit the locality of user's accesses in the recovery process in RAID1 disk arrays. Similarly, Tian et al. proposed PRO [7] to deploy a popularity–based multi-threaded scheduling algorithm to rebuild the frequently-accessed areas prior to rebuilding infrequently-accessed areas for both mirroring and parity-encoded disk arrays, to exploit access locality

and disk sequentiality. Sivathanu et al. proposed a live-block recovery [6] approach in their D-GRAID model, which recovers only those data blocks that are live from the perspective of file systems and databases. Because our SPP approach is an architectural approach orthogonal to PRO and D-GRAID, integrating them into our SPP will further improve system reliability and performance of parity-encoded RAID systems during recovery.

Finally, the TRAP-array approach [32] leverages the XOR operations performed upon each block write in RAID4/5 controller, so it can rewind the sequence and history of XORs resulting from writes to recover data to any point-in-time upon data damage, as a means to achieve effective snapshot. Interestingly, to guarantee high system reliability, our SPP exploits the XOR operations among the dataset space (i.e., along the spatial dimension) while TRAP exploits the XOR operations among the timeline space (i.e., along the temporal dimension).

7. Conclusions and Future Work

In this paper we propose a Supplementary Partial Parity (SPP) mechanism, an architectural approach to improving the availability of parity-based RAID systems. The basic idea of the SPP architecture is to incorporate partial parity into the existing parity encoded protection mechanism for redundant disk arrays to improve system reliability and availability. In particular, SPP exploits free storage space and idle or lightly-loaded periods to generate and update the partial parity units during fault-free operations, thus achieving significant reduction in reconstruction overhead and improving failure-resistant capability. Via leveraging the exclusive OR operation appropriately among partial parity, full parity and data units, SPP not only can reconstruct the data on the failed disks with a fraction of the original overhead compared with the conventional full parity, thus significantly reducing the overhead of data regeneration, especially under heavy workload, but also can tolerate more than one disk failure with much better flexibility.

We implement a prototype of the SPP architecture and incorporate it into Linux software RAID. Extensive trace-driven experiments are conducted to evaluate the performance advantages of SPP. The experimental results demonstrate that our SPP approach can significantly improve the reconstruction time and response time performance during recovery with high flexibility, adaptability and scalability.

Since SPP is an on-going project, we feel that supplementary partial parity and related performance/cost tradeoffs is an area rich in research issues that we will address as our future work. For example, we will implement prototypes for SPP Level 2 and Level 3 to examine their performance advantages by exploiting the workload characteristics and semantic knowledge and evaluate implementation complexity and design tradeoffs. Additionally, we will exploit the impacts of the SPP approach on the non-standard parity encoded disk arrays such as systems incorporated with the Parity Declustering and Parity Logging techniques. Finally, we will extend the SPP approach to a distributed storage system or a mobile data center and evaluate the new design issues and challenges in these new environments.

8. Acknowledgement

We thank the Storage Performance Council and UMass Trace Repository for providing us the I/O traces. This work is sponsored by the US NSF under Grant No. CCF-0621526 and the National Basic Research Program of China (973 Program) under Grant No. 2004CB318201.

9. References

[1]. Bianca Schroeder and Garth A. Gibson. Disk Failures in the Real World: What Does an MTTF of 1,000,000 Hours Mean to You? In *FAST '07*, San Jose, CA, February 2007.

[2]. Eduardo Pinheiro, Wolf-Dietrich Weber, and Luiz Andre Barroso. Failure Trends in a Large Disk Drive Population. In *FAST '07*, San Jose, CA, February 2007.

[3]. Jon G. Elerath and Michael Pecht. Enhanced Reliability Modeling of RAID Storage Systems. In *DSN '07*, Edinburgh, UK, June 2007.

[4]. Haryadi S. Gunawi, Vijayan Prabhakaran, Swetha Krishnan, Andrea C. Arpaci-Dusseau, Remzi H. Arpaci-Dusseau. Improving File System Reliability with I/O Shepherding. In *SOSP '07*, Stevenson, WA, Oct, 2007.

[5]. Nikolai Joukov, Arun Krishnakumar, Chaitanya Patti, Abhishek Rai, Sunil Satnur, Avishay Traeger, and Erez Zadok. RAIF: Redundant Array of Independent Filesystems. In *MSST '07*, San Diego, CA, September 2007.

[6]. Muthian Sivathanu, Vijayan Prabhakaran, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Improving Storage System Availability with D-GRAID. In *FAST '04*, San Francisco, CA, March/April 2004.

[7]. Lei Tian, Dan Feng, Hong Jiang, Ke Zhou, Lingfang Zeng, Jianxi Chen, Zhikun Wang, and Zhenlei Song. PRO: A Popularity-based Multi-threaded Recon-struction Optimization for RAID-Structured Storage Systems. In *FAST '07*, San Jose, CA, February 2007.

[8]. Kiron Vijayasankar, Gopalan Sivathanu, Swaminathan Sundararaman and Erez Zadok. Exploiting Type-Awareness in a Self-Recovering Disk. In *StorageSS '07*, Alexandria, Virginia, October 2007.

[9]. Kevin Greenan, Ethan Miller, Thomas Schwarz and Darrell Long. Disaster Recovery Codes: Increasing Reliability with Large-Stripe Error Correction Codes. In *StorageSS '07*, Alexandria, Virginia, October 2007.

[10]. David A. Patterson, Garth Gibson, and Randy H. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD '88*, June 1988.

[11]. Petascale Storage System to Be Built at MIT. http://www.hpcwire.com/hpc/661879.html

[12]. Garth A. Gibson. Reflections on Failure in Post-Terascale Parallel Computing. Keynote in *ICPP '07*, Xi'an, China, September 2007.

[13]. Dave Anderson and Willis Whittington. Hard Drives: Today & Tomorrow. Tutorial in *FAST '07*, San Jose, CA, February 2007.

[14]. Jeffrey S. Chase, Darrell C. Anderson, Prachi N. Thakar, and Amin M. Vahdat. Managing Energy and Server Resources in Hosting Centers. In *SOSP '01*, Banff, Canada, October, 2001.

[15]. Charles Weddle, Mathew Oldham, Jin Qian, and An-I Andy Wang. PARAID: A Gear-Shifting Power-Aware RAID. In *FAST '07*, San Jose, CA, February 2007.

[16]. Chris Ruemmler and John Wilkes. UNIX Disk Access Patterns. In *Winter USENIX '93*, January 1993.

[17]. Lei Tian, Hong Jiang, Dan Feng, Qin Xin, and Xing Shu .Implementation and Evaluation of a Popularity-Based Reconstruction Optimization Algorithm in Availability-Oriented Disk Arrays. In *MSST '07*, San Diego, CA, September 2007.

[18]. ACNC. RAID LEVEL 50: High I/O Rates & Data Transfer Performance. http://www.acnc.com/04_01_50.html

[19]. Gopalan Sivathanu, Swaminathan Sundararaman, and Erez Zadok. Type-Safe Disks. In *OSDI '06*, Seattle, WA, November 2006.

[20]. Muthian Sivathanu, Vijayan Prabhakaran, Florentina I. Popovici, Timothy E. Denehy, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Semantically-Smart Disk Systems. .In *FAST '03*, San Francisco, CA, March/April 2003.

[21]. Muthian Sivathanu, L. N. Bairavasundaram, Andrea C. Arpaci-Dusseau, and Remzi H. Arpaci-Dusseau. Life or Death at Block-Level. In *OSDI '04*, San Francisco, CA, December 2004.

[22]. Nitin Agrawal, William J. Bolosky, John R. Douceur, and Jacob R. Lorch. A Five-Year Study of File-System Metadata. In *FAST '07*, San Jose, CA, February 2007.

[23]. UMass Trace Repository. http://traces.cs.umass.edu/index.php/Storage/Storage.

[24]. Storage Performance Council. http://www.storageperformance.org/home.

[25]. Stefan Savage and John Wilkes. AFRAID--A Frequently Redundant Array of Independent Disks. In *USENIX '06*, San Diego, CA, January 1996.

[26]. Peter A. Franaszek and John T. Robinson. On Variable Scope of Parity Protection in Disk Arrays. *IEEE Transaction on Computer*, VOL. 46, NO. 2, February 1997.

[27]. Richard R. Muntz and John C. S. Lui. Performance Analysis of Disk Arrays under Failure. In *VLDB '90*, Brisbane, Australia, August 1990.

[28]. Mark Holland and Garth A. Gibson. Parity Declustering for Continuous Operation in Redundant Disk Arrays. In *ASPLOS '92*, Boston, MA, October 1992.

[29]. Daniel Stodolsky, Garth Gibson, and Mark Holland. Parity Logging: Overcoming the Small Write Problem in Redundant Disk Arrays. In *ISCA '93*, San Diego, CA, May 1993.

[30]. Jai Menon, James Roche and Jim Kasson, Floating Parity and Data Disk Arrays. *Journal of Parallel and Distributed Computing*, 17,129-139, 1993.

[31]. Eitan Bachmat and Jiri Schindler. Analysis of Methods for Scheduling Low Priority Disk Drive Tasks. In *SIGMETRICS '02*, Marina Del Rey, CA, June 2002.

[32]. Qing Yang, Weijun Xiao, and Jin Ren. TRAP-Array: A Disk Array Architecture Providing Timely Recovery to Any Point-in-time". In *ISCA '06*, Boston, MA, June 2006.