Implementation and Evaluation of a Popularity-Based Reconstruction Optimization Algorithm in Availability-Oriented Disk Arrays

Lei Tian[†] Hong Jiang[‡] Dan Feng[†] Qin Xin[§] Xing Shu[†]
ltian@hust.edu.cn *jiang@cse.unl.edu* *dfeng@hust.edu.cn* *qxin@ieee.org* *xshu2006@gmail.com*
[†] *Huazhong University of Science and Technology / Wuhan National Laboratory for Optoelectronics*
[‡] *University of Nebraska-Lincoln*
[§] *Symantec Corporation*

Abstract

In this paper, we implement the incorporation of a Popularity-based multi-threaded Reconstruction Optimization algorithm, PRO, into the recovery mechanism of the Linux software RAID (MD), which is a well-known and widely-used availability-oriented disk array scheme. To evaluate the impact of PRO on RAID-structured storage systems such as MD, we conduct extensive trace-driven experiments. Our results demonstrate PRO's significant performance advantage over the existing reconstruction schemes, especially on a RAID-5 disk array, in terms of the measured reconstruction time and response time.

1. Introduction

With the advent of RAID [1] or RAID-like storage systems, a large body of research work has been proposed in the literature to improve the conventional data recovery mechanisms from disk failures, and the practicability and applicability of these improved mechanisms have been demonstrated. Generally, we can divide the previously published work in this area into the three categories of erasure code-based approaches, parity/spare layout-based approaches, and recovery workflow-based approaches.

Since the mechanism evaluated in this paper belongs to the third category of workflow-based approaches, we will focus our discussion of recent and new approaches in this category. Tian et al. [2] proposed a popularity-based multi-threaded reconstruction optimization algorithm (PRO) to optimize the recovery process deployed in disk arrays by integrating the popularity and locality of workloads into the recovery process.

The PRO algorithm is demonstrated to improve the reconstruction time and average I/O response time during recovery by optimizing the original Disk-Oriented Reconstruction (DOR) [3] approach in RAIDframe [4] with the read-only workload of a machine running a web-search engine. However, there are a number of limitations and weaknesses of PRO, discussed below, which must be addressed:

1). The PRO approach has been incorporated and evaluated only in one of the existing common reconstruction schemes, Disk-Oriented Reconstruction. However, the effect of incorporating PRO in another typical reconstruction approach, such as Pipeline Reconstruction (PR) [5] in the Linux software RAID (MD) has not been investigated and evaluated in practice or analyzed quantitatively. We believe that it is imperative to incorporate PRO into PR because, importantly, PR in MD is based on a significantly different design philosophy than DOR in RAIDframe, e.g., MD favors availability [6] while RAIDframe favors reliability. We summarize the main differences between MD's PR and RAIDframe's DOR as follows: First, DOR associates lower priority with the reconstruction I/O requests than with requests of user accesses, while PR associates the same priority to the reconstruction requests. Secondly, DOR allocates most of the available disk bandwidth for fast recovery, while PR preserves a reasonable amount of disk bandwidth (the default range is from 1MB/s to 20MB/s) to avoid imposing negative performance impact on user accesses. Thirdly, DOR integrates the redirection of reads mechanism [3] while PR performs on-the-fly reconstruction to re-generate data once users' reads arrive on the failed disk. Besides, many product-level storage systems consist of underlying MD implementations. EMC Centera [7], for example, is built from a cluster of commodity machines and manages the underlying disks by MD.

2). The PRO approach has been evaluated only in terms of the read-only workload of a web search engine application [8, 9]. However, with respect to other representative read/write workloads, such as mail server applications and Online Transaction Processing (OLTP) applications, the impact of PRO has not been evaluated. It is necessary to conduct trace-driven experiments to assess

24th IEEE Conference on Mass Storage Systems and Technologies (MSST 2007)
0-7695-3025-7/07 $25.00 © 2007 IEEE

Table 1. A Description of the PRO's APIs

API Name	Function Description	Function invoked
pro_init_desp	Initiaze the description of PRO	raidx_run
pro_init_bitmap	Initiaze the reconstruction bitmap table(RBT)	raidx_run
pro_find_available_thread	Find an available thread to allocate a new hot zone	raidx_make_request
pro_init_thread	Initiaze a thread with its corresponding hot zone	raidx_make_request
pro_start_thread	Switch the thread to the running or suspended state	raidx_make_request
pro_find_hotzone	Find whether the stripe is in any hot zone or not	raidx_make_request
pro_sort_threads	Sort all of the threads according to their priority(popularity)	md_do_sync
pro_schedule_stripe	Select the highest-priority thread and issue a stripe to recon	md_do_sync
pro_free_bitmap	Free the memory allocated for RBT	raidx_stop
pro_free_desp	Free the memory allocated for the description of PRO	raidx_stop

PRO's impact on both availability and reliability during the recovery process.

To address the issues above, we incorporate the PRO approach into PR in the Linux software RAID, and conduct a series of experiments and measurements to evaluate the impacts of PRO on an availability-oriented platform.

The main contributions of our paper are:

1). we implement the incorporation of PRO into the PR scheme in the Linux software RAID.

2). we conduct extensive trace-driven experiments in a real implementation, to investigate the performance and reliability improvements of PRO over PR. The results show the significant advantage of PRO over PR in availability-oriented disk arrays, especially in RAID-5 disk arrays.

2. Design and Implementation of PRO

In this section, the PRO architecture and its design are presented, along with its detailed implementation issues and current limitations.

2.1. Design and Architecture of PRO

As a reconstruction optimization algorithm, PRO's goal is to optimize the existing recovery approaches to generate an optimal reconstruction sequence for such parallel reconstruction algorithms as Pipeline Reconstruction (PR) and Disk-Oriented Reconstruction (DOR). Due to the access popularity that is ubiquitous in real I/O workloads, the main function of PRO is to integrate workload characteristics into the existing recovery approaches, rendering redirection of reads [10] and head-following [3] much more efficient, thus achieving the goals of improving the reconstruction time and response time simultaneously.

The key idea of PRO is to allow the reconstruction process in a RAID-structured storage system to rebuild the frequently accessed areas prior to rebuilding infrequently accessed areas.

More specifically, PRO firstly divides the storage space of the replacement disk into contiguous and non-overlapping areas, called "hot zones", and initializes multiple independent reconstruction threads with each thread being responsible for rebuilding its corresponding hot zone. Furthermore, the priority of a thread is correlated to the popularity of its hot zone; Secondly, after the successful initialization of the reconstruction threads, PRO selects a reconstruction thread that has the highest priority, allocates a time slice to it and activates it to rebuild the remaining data units of its hot zone until the time slice is used up. If the time slice is ran out of by this thread, PRO suspends it, re-selects a reconstruction thread with the currently highest priority, and allocates one new time slice to it. This process repeats until all of the data units in the replacement disk have been rebuilt.

From the architectural viewpoint, PRO is divided into three modules: Access Monitor (AM), Reconstruction Scheduler (RS) and Reconstruction Executer (RE). AM is responsible for monitoring the users' accesses and adjusting the corresponding hot zones or initializing new hot zones and new reconstruction threads. RS is responsible for sorting the threads by their priority, selecting the highest-priority thread, allocating a time slice, and switching it to start rebuilding its remaining data stripes. RE is responsible for rebuilding the corresponding stripes issued by RS on the replacement disks one by one. (See [2] for details)

2.2. Implementation Issues

To adapt PRO in PR of MD, we define ten main APIs. Table 1 details information about the APIs of PRO on MD.

Table 1 lists the main APIs of PRO that need to be augmented in PR of MD. Among them, the function of *pro_schedule_stripe* is the core part of PRO, responsible for the scheduling algorithm and connecting the AM module and RE module.

The entire PRO source code is around 1000 lines of C code, and it is easy to be shared with the RAID-1/RAID-4/RAID-5/RAID-10 kernel modules of MD.

Table 2. The Characteristics of the WebSearch and Financial Traces.

Trace Name	Num. of Requests	RDs/WRs ratio	I/O Intensity (IOPS)
WebSearch	200,000	99.98% / 0.02%	323.75
Financial	200,000	35.91% / 64.09%	64.06

Different from the DOR implementation in RAIDframe, where the data structures of a bitmap array pointer (*RF_ReconMap_s*) and an item pointer of the bitmap array (*RF_ReconMapListElem_s*) are deployed to indicate that a reconstruction unit has been either totally reconstructed or not at all, the PR implementation in MD does not provide a bitmap table to keep track of whether every stripe unit has been rebuilt on the failed disk.

As a result, PRO augments a similar data structure, Reconstruction Bitmap Table (RBT), into the original PR implementation to indicate whether a stripe unit has been rebuilt or not. We use one bit to represent each stripe unit, and a true value ('1') denotes that the corresponding stripe unit has been reconstructed; a false value ('0') denotes that the corresponding stripe unit needs reconstruction. In the current MD implementation, the size of a stripe unit is the same as the page size (always 4KB). Denoting all the stripe states of a 320 GB hard-disk will consume 10MB memory. However, the memory for the storage of reconstruction bitmap table can be reduced proportionally if a bit is designed to cover two or more consecutive stripe units.

Another implementation issue that must be addressed is that the data structure of RBT will be lost if a power supply failure occurs during recovery. Once we plan to re-start the recovery process to rebuild the remaining units, we must reconstruct all of the units without RBT. One of the available solutions for this problem is to utilize NVRAM to store RBT, or flush the content of RBT on a hard-disk temporarily just like the bitmap mechanism embedded into the current MD version. We believe that it is worthwhile to add extra memory or hardware considering the benefits gained from PRO.

3. Performance Evaluations

In this section, we present a trace-driven evaluation of an implementation of PRO embedded in the Linux MD kernel module. This performance study concentrates on the reconstruction performance in terms of average I/O response time during recovery and reconstruction time.

3.1. Setup Details

All the experiments were performed on a server-class PC with an Intel 3GHz Pentium4 Xeon processor and 512MB DDR memory. There is a Highpoint RocketRAID 2240 SATA card in the system to house 10 Seagate ST3300831AS SATA2 disks. Each disk, with 300GB capacity and 8MB cache, spins at 7200RPM, with a sustained transfer rate of up to 76MB/s. We limit the capacity of every disk to 5GB in experiments to avoid the redundant time-consuming recovery process for an entire disk. The MD software including its configuration tool (mdadm) is shipped with the Fedora Core 4 Linux (kernel version: 2.6.11). Based on the PR approach of MD, we incoporate PRO into it. For the same reason as above, we scale the default bandwidth range to between 10MB/s and 30MB/s.

3.2. Benchmark and Workloads

We have evaluated our implementation through extensive trace-driven experiments and have conducted performance evaluations by using RAIDmeter [2], which is a block-level trace replay software tool with functions of replaying traces and evaluating the I/O response time of the storage device.

We run our evaluations over two traces identified from the Storage Performance Council [8, 9]. The first one, Financial, was collected from OLTP applications running at a large financial institution, and the other one, WebSearch, was collected from a system running a popular search engine. Because of the relatively short recovery time, we only use the beginning part of these two traces that are sufficient for our evaluations. Table 2 shows the relevant workload characteristics of these two traces.

3.3. Numerical Results

To evaluate the performance of two reconstruction schemes: PR and PRO-powered PR (PRO for short), we conduct the first set of experiments on a RAID-5 disk array and a RAID-10 disk array composed of a variable number of hard disks and one single hot-spare disk, with the same chunk size of 64KB.

Table 3 depicts the measured average response time of the two approaches respectively. On a RAID-5 disk array, PRO outperforms PR by up to 30.82% and 18.74% in user response time for the Financial and WebSearch traces, respectively. It clearly shows the efficacy of PRO. One can see that the response time improvement of PRO for a RAID-5 disk array is more significant than that for a RAID-10 disk array. On the other hand, PRO consistently

Table 3. A comparison of PRO and PR user response time as a function of the number of disks.

RAID Level	Number of Disks	Average User Response Time during recovery (millisecond)					
		WebSearch			Financial		
		PR	PRO	improved	PR	PRO	improved
RAID-5	3	225.67	183.37	18.74%	92.44	63.95	30.82%
	5	252.98	210.67	16.72%	68.34	54.46	20.31%
	7	258.06	230.25	7.85%	71.36	55.93	21.62%
	9	257.86	236.55	8.26%	62.62	48.55	22.47%
RAID-10	4	235.61	233.91	0.72%	50.75	48.81	3.82%
	6	238.45	239.66	-0.51%	59.88	51.50	13.99%

Table 4. A comparison of reconstruction time in PRO and PR as a function of the number of disks.

RAID Level	Number of Disks	Reconstruction Time (second)					
		WebSearch			Financial		
		PR	PRO	improved	PR	PRO	improved
RAID-5	3	488.23	487.95	0.06%	247.73	236.57	4.5%
	5	483.20	484.22	-0.21%	313.63	244.23	22.13%
	7	487.29	489.41	-0.44%	315.89	251.66	20.33%
	9	488.36	487.66	1.43%	315.75	284.63	9.86%
RAID-10	4	487.53	487.98	-0.09%	418.11	418.74	-0.15%
	6	489.31	486.86	5.03%	421.33	442.52	-5.03%

exhibits scalable performance improvement as the number of disk drives increases.

Figure 1 illustrates the noticeable improvement of the PRO algorithm over PR in user response time on a RAID-5 disk array. One can see that with respect to the WebSearch trace, the onset of the PRO performance improvement is much earlier than that of PR during recovery. On the other hand, it is demonstrated that PRO rapidly reduces user response time to a lower level prior to the PR algorithm, and preserves this level steadily until recovery is ended. However, the performance of PRO in user response time is similar to that of PR on a RAID-10 disk array.

Table 4 depicts the reconstruction time results of the two approaches. The improvement of PRO over PR on a RAID-5 disk array is negligible under the WebSearch trace while significant, under the Financial trace, with up to 22.13% improvement. It is noted that because MD is availability-oriented, its recovery mechanism allocates a nominal bandwidth range to rebuild data units while preserving a significant portion of the bandwidth for user requests. Due to the heavier intensity of the WebSearch workload, no matter which approach we deploy in MD, it is observed that the reconstruction speed persists at around 10~11MB/s, the pre-specified minimal bandwidth threshold by our experiments. It demonstrates that MD has to guarantee a minimal recovery bandwidth with the heavy workload. However, because of the lower intensity of the Financial workload, MD leverages the available bandwidth to rebuild data stripes; enabling PRO to effectively exploit popularity and locality to reduce seek latency much more than PR. However, the improvement or degradation by PRO over PR on a RAID-10 disk array is relatively slight. It is also noted that the overhead of reconstructing data on the fly is lower for RAID-10 than for RAID-5. Another reason is that a RAID-10 disk array deploys a read-balance method to service users' accesses. The two reasons cause the less significant improvement on a RAID-10 disk array than on a RAID-5 disk array.

Figure 2(a) and Figure 2(b) shows the reconstruction time, response time and overall improvement of PRO over PR on a RAID-5 disk array consisting of 3 disks and 1 hot-spare disk, with chunk sizes of 64KB if we raise the minimal and maximal bandwidth thresholds to 100MB/s, which is not obtainable for the hard-disks we used. As a result, PR prefers to utilize any hard-disk bandwidth available to rebuild data blocks, which is similar to DOR in RAIDframe. In Figure 2(a) and Figure 2(b), one can see that PRO outperforms PR in both reconstruction time and user response time by up to 4.44% and 27.65%-47.17% respectively. From Figure 2, one can see obviously that allocating more bandwidth helps both PR and PRO, but PRO benefits significantly more.

We conducted performance evaluations on the platform of a RAID-5 disk array consisting of 3 disks and 1 hot-spare disk, with chunk sizes of 16KB and 64KB to investigate the impact of chunk sizes. Figure 3(a) and Figure 3(b) illustrates the measured reconstruction time and average response time. One can see that for the three

stripe sizes we examined, the PRO algorithm outperforms the PR in average response time.

The results illustrate that PRO outperforms PR both in average response time during recovery and reconstruction time for MD, an availability-oriented disk array, especially on a RAID-5 disk array. It is noted that PRO outperforms PR by a large margin in average response time, and is slightly better in reconstruction time than PR.

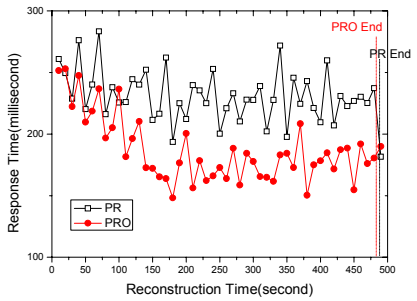


Figure 1(a) WebSearch, RAID-5

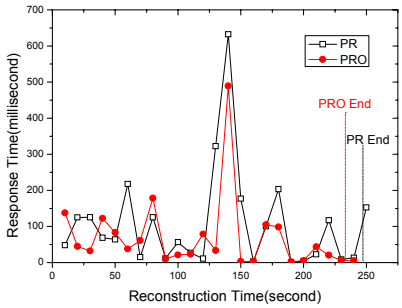


Figure 1(b) Financial, RAID-5

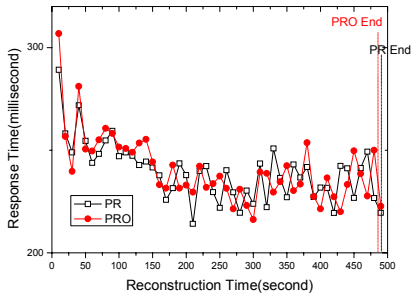


Figure 1(c) WebSearch, RAID-10

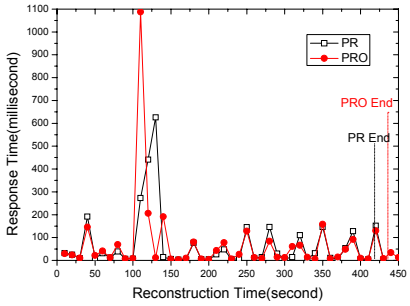


Figure 1(d) Financial, RAID-10

Figure 1(a), (b), (c), (d). A comparison of PRO and PR user response time on a RAID-5 and RAID-10 disk array as a function of the respective traces: WebSearch and Financial. In all of these figures, the two curves show the PRO and DOR user response time trend during recovery. Figure 1(a) and Figure 1(b) depict the response time trend on a RAID-5 disk array consisting of 3 disks and 1 hot-spare disk, with a stripe unit size of 64KB. Figure 1(c) and Figure 1(d) depict the response time trend on a RAID-10 disk array consisting of 6 disks and 1 hot-spare disk, with a stripe unit size of 64KB.

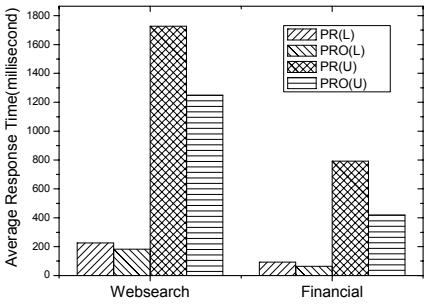


Figure 2 (a) avg. response time

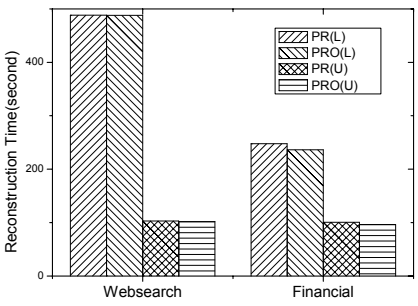
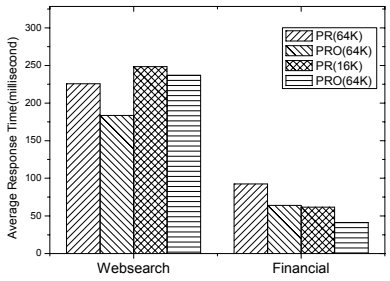


Figure 2 (b) reconstruction time

Figure 2(a), (b). A comparison of PRO and PR user response time and reconstruction time on a RAID-5 disk array consisting of 3 disks and 1 hot-spare disk with a stripe unit size of 64KB as a function of the respective traces: WebSearch and Financial. In all of these figures, PR(L) and PRO(L) signify that MD preserves the bandwidth between 10MB/s and 30MB/s while PR(U) and PRO(U) indicates that MD utilizes any available bandwidth to for the fastest recovery.



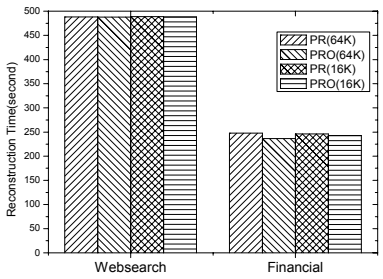


Figure 3 (a) avg. response time **Figure 3 (b) reconstruction time**

Figure 3(a), (b): A comparison of PRO and PR average response time during recovery and reconstruction time as a function of the respective stripe unit size: 16KB and 64KB.

4. Conclusions and Future Work

In this paper, we present the incorporation of a Popularity-based multi-threaded Reconstruction Optimization algorithm (PRO) into the Pipeline Reconstruction (PR) recovery approach of Linux software RAID (MD). PRO optimizes the reconstruction sequence for the existing parallel recovery algorithms. We implement PRO in the Linux software RAID kernel module and evaluate the performance impact of PRO by conducting extensive trace-driven experiments. Our experimental results have demonstrated that PRO can greatly improve reconstruction performance in availability-driven disk arrays. Compared with PR, PRO results in up to 30.82% improvement in average response time and up to 22.13% improvement in reconstruction time in a RAID-5 system.

We believe that there are still many directions for future research on PRO. One potential direction is to incorporate the preemptive scheduling algorithm instead of the original unitary priority-based time-sharing scheduling algorithm to exploit the rapidly changing popularity. Another idea is to investigate the impacts of PRO in the recovery mechanisms in distributed storage systems with appropriate benchmarks and real workloads.

Acknowledgments

We thank the Storage Performance Council and UMass Trace Repository for providing us the I/O traces, the anonymous reviewers for their valuable comments in reviewing this paper. This work is sponsored by the National Basic Research Program of China (973 Program) under Grant No. 2004CB318201, the Program for New Century Excellent Talents in University NCET-04-0693 and NCET-06-0650, and the US NSF under Grant No. CCF-0621526.

References

[1] D. Patterson, G. Gibson, and R. Katz. A Case for Redundant Arrays of Inexpensive Disks (RAID). In *SIGMOD '88*, 1988.

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

[3] M. Holland. On-Line Data Reconstruction in Redundant Disk Arrays. Carnegie Mellon Ph.D. Dissertation CMU-CS-94-164, April 1994.

[4] W.V. Courtright II, G.A. Gibson, M. Holland and J. Zelenka. RAIDframe: Rapid Prototyping for Disk Arrays. In *SIGMETRICS '96* Vol. 24 No. 1, May 1996.

[5] Jack Y.B. Lee and John C.S. Lui. Automatic Recovery from Disk Failure in Continuous-Media Servers. IEEE Transaction On Parallel and Distributed Systems, Vol. 13, No. 5, May 2002.

[6] A. Brown and D. A. Patterson. Towards Maintainability, Availability, and Growth Benchmarks: A Case Study of Software RAID Systems. In *USENIX '00*, pages 263-276, San Diego, CA, June 2000.

[7] EMC. EMC Centera: Content Addressed Storage System. http://www.emc.com/.

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

[9] OLTP Application I/O and Search Engine I/O. UMass Trace Repository. http://traces.cs.umass.edu/index.php/Storage/Storage.

[10] R. Hou, J. Menon, and Y. Patt. Balancing I/O Response Time and Disk Rebuild Time in a RAID5 Disk Array. In *Proceedings of the Hawaii International Conference on Systems Sciences*, pages 70-79, 1993.