Power-aware Remote Replication for Enterprise-level Disaster Recovery Systems

Kazuo Goda Masaru Kitsuregawa
Institute of Industrial Science, the University of Tokyo, Japan
{*kgoda,kitsure*}*@tkl.iis.u-tokyo.ac.jp*

Abstract

Electric energy consumed in data centers is rapidly growing. Power-aware IT, recently called 'green IT', is widely recognized as a significant challenge. Disk storage is a non-negligible energy consumer. Rather, in light of recent data-intensive systems where a number of disk drives are incorporated, the disk storage may be what we must consider primarily. Yet, all of the disk drives are not used for primary datasets, but rather larger portion of them are utilized for storing a variety of *copies* such as backups and snapshots. Saving the energy of such storage resources that manage copies is a promising approach. The paper presents a power-aware disaster recovery system, in which the reflection of transferred updated information can be deferred through eager compaction technique. Great energy saving of storage systems is expected in the remote secondary site. Our experiments using a commercial database system show that 80-85% energy of the secondary-site disk storage can be saved with small penalties of possible service breakdown time.

1 Introduction

Many attentions are paid on the energy consumption of IT systems, which has been grown up by 25% every year [9]. The recent analysis [2] reports that the annual electricity cost paid by the system owner will go up to twice higher than the annual server expense in 2009. More and more powerful cooling systems and power-supply equipments are being installed into data centers for accommodating the increase of energy consumption; accordingly, the electric energy and the related equipments account for 44% of TCO in a typical system [1]. In addition to the cost issue, energy and heat management has become a key of data center design and operation. The exploding energy consumption might strictly constrain the design space of modern IT systems [32]. Hence, energy saving is a grand challenge for IT research and development.

Storage systems are non-negligible energy consumer in IT systems. Storage systems present 27% of total energy consumption in a typical data center [23]. As the digital data volume is explosively increasing [16], extremely many disk drives are being incorporated into an enterprise system to improve the throughput. Thus, much larger portion of the total energy may be consumed by the storage system in high-performance data-intensive systems. Q. Zhu et al. in paper [34] points out an interesting example, where disk drives account for 71% of the total energy consumption in a large-scale OLTP system. Therefore, energy saving of the disk storage is rather essential as well as server processors and network devices.

Interestingly, all the disk drives of recent enterprise disk storage are not necessarily used for primary datasets. Rather, larger portion of the disk drives are utilized for storing a variety of *copies* to improve the system performance and availability. Suppose a simple IT system, which holds a single snapshot in a local data center and a backup copy in a remote data center. Two thirds of all the disk drives equipped in the total system are used for copies. Modern enterprise systems may use much more disk drives for copies [18]. Saving the energy of such storage resources should be a natural idea.

The paper proposes a power-aware disaster recovery system. It has been widely recognized that the business breakdown due to unpredictable disasters such as terrors and hurricanes provides a nation and a society with terrible damage [7, 25]. Business continuity is being enforced by nation-level legal systems as well as by enterprise-level internal disciplines [3, 26, 29]. The disaster recovery system [8, 15, 17] is a practical solution which places a remote secondary site and, in case of disaster, continues the business on the secondary site. By concentrating transferred update information through eager compaction technique, our proposed system can derive longer idle time of the data volume in order to reduce significantly the energy consumption of disk storage of the secondary site with small penalties of service break-

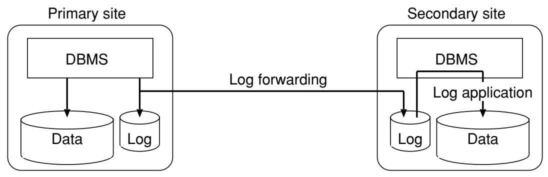


Figure 1: A disaster recovery system based on database log forwarding.

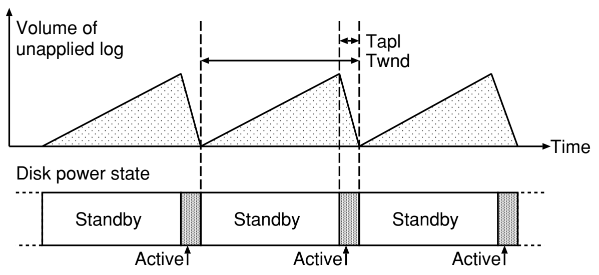


Figure 2: Deferred log application.

down time. In the real disaster recovery system, many resources of the secondary site may not be fully utilized when the primary site is normally operating. Great energy saving is expected in practice. To the best of our knowledge, similar researches have not been published.

This paper is organized as follows. Section 2 concisely describes disaster recovery systems that are currently deployed in many enterprise systems. Section 3 proposes novel techniques for energy saving of disaster recovery systems and Section 4 evaluates the proposal through the experiments using a commercial database system. Section 5 briefly summarizes related works and finally Section 6 concludes the paper.

2 Disaster recovery system

A disaster recovery system comprises two or more sites. Business is usually operated in the primary site and, once a disaster damages the primary site, the business is continued in the remote secondary site. For enabling such disaster recovery, up-to-date data of the primary site must be always copied into the secondary site. Many solutions have been proposed in papers and deployed into real systems, but the basic idea is similar in that they are composed of the following two steps: (1) transferring only updated information of the primary site to the secondary site and (2) reflecting it to the storage in the secondary site. Here the updated information means queries or transactions for the conventional logical database replication, changed blocks for the storage-level physical block forwarding [8, 17], and database log entries for the log forwarding [15, 27].

The recovery capability of such a disaster recovery system can be defined by two metrics: recovery point objective (RPO) and recovery time objective (RTO). RPO denotes possibility of data loss, i.e. how latest data can be recovered in case of disaster. RTO means inter-site takeover overhead, i.e. how soon the business can start again in the secondary site in case of disaster. It is preferable that RPO and RTO would be small. This paper focuses on enterprise-level systems such as brokerage and e-commerce, which accept only small service breakdown time and do never allow any data loss even in case of disaster. Thus, we assume here that inter-site data transfer is done in the synchronous fashion; specifically, RPO is always zero. Needless to say, our contribution can be directly applied to more relaxed asynchronous modes.

3 Power-aware disaster recovery system

To save the energy consumption of disaster recovery systems, we focus on the disk storage in the secondary site, in which most of the storage resources are used only for storing backup copies.

Let us describe a scenario based on database log forwarding, which is recently deployed in high-end disaster recovery systems. Figure 1 illustrates a disaster recovery system based on database log forwarding, where physical database log is shipped from the primary site to the secondary site and the forwarded log is applied in the secondary site.

3.1 Deferred log application

Our idea is to *defer* the application of transferred database log to derive longer idle time of the data volume in the secondary site. Figure 2 illustrates a batch application scheme for realizing such deferred log application. In the secondary site, the transferred log is stored in the log volume and is not immediately applied to the data volume. While the database log is not being applied, the data volume is idle, so that the energy consumption of the data volume can be saved by spinning down the volume. Longer standby time gives greater energy saving, but provides larger amount of unapplied log stored in the log volume. In case of disaster, the secondary site must apply all the unapplied log before starting the business again. Therefore, deferability of log application is mainly determined by RTO requirements.

Let us discuss the relationship between RTO requirements and deferability. Let R_{gen} and R_{apl} be the log generation rate in the primary site and the maximum log application rate in the secondary site respectively. We assume that the recovery time of the secondary site is proportional to the amount of unapplied log[1]. The fol-

[1]Fast log application is a key to quick recovery [19]. Other marginal

Table 1: Typical OLTP systems

Rank	Vendor	System	tpmC	Database	# of disks (data volume)	# of disks (log volume)
1	IBM	System p5 595	4,033,378	IBM DB2 9	6400	360
2	IBM	eServer p5 595	3,210,540	IBM DB2 UDB 8.2	6400	140
3	IBM	eServer p5 595	1,601,784	Oracle Database 10g	3200	96
4	Fujitsu	PRIMEQUEST 540 16p/32c	1,238,579	Oracle Database 10g	1920	224
5	HP	Integrity Superdome 64p/64c	1,231,433	Microsoft SQL Server 2005	1680	56

Quoted from *Top Ten TPC-C by Performance Version 5 Results* disclosed at `http://www.tpc.org/` as of December 11, 2006.

lowing formulae give the optimal batch configuration: a batch interval T_{wnd} and a necessary log application time in the interval T_{apl}, the combination of which can gain maximum energy conservation. For the page limitation, we have to omit the mathematical proof.

$$\begin{aligned} T_{wnd} &= \frac{R_{apl}^2}{(R_{apl} - R_{gen}) \cdot R_{gen}} T_{RTO} \\ T_{apl} &= \frac{R_{apl}}{R_{apl} - R_{gen}} T_{RTO} \end{aligned}$$

Here T_{RTO} denotes a RTO requirement, i.e. given allowance of service breakdown time. Implicitly, $R_{apl} > R_{gen}$ and $T_{RTO} > T_{up} + T_{down}$, where T_{up} and T_{down} denote time penalties of spinning up and down respectively. The secondary site can concentrate log application based on the above batch configuration and proactively spin up and down the data volume in order to save energy consumption of the disk storage.

3.2 Eager log compaction

Obviously, larger $\frac{R_{apl}}{R_{gen}}$ leads to greater energy saving. In this section, we introduce *eager compaction* technique to improve the log application throughput significantly.

In a disaster recovery system based on log forwarding, the transferred log entries are applied to the data volume in a way similar to database redo operation. In the normal redo operations, log entries are applied strictly in log sequence number (LSN) order. On the contrary, our proposed eager techniques can compact the log sequence in a window buffer and apply the compacted sequence to the data volume. The compaction process comprises *log folding* and *log sorting*. Log folding is a technique to reduce the number of log entries to be applied. The log entries which manipulate the identical record are coalesced in the window buffer. For example, assuming that three log entries, `insert(data1)`, `update(data1` $\rightarrow$ `data2)` and `update(data2` $\rightarrow$ `data3)`, are given in the sequence, these three entries, manipulating the same record, can be logically folded into a single entry, `insert(data3)`. On the other hand, log sorting reorders log entries in the window buffer to improve disk access sequentiality. In many cases, an entry of database log has a physical reference to the target record. Log sorting leverages such physical information. Both the methods together can improve the log application throughput. Accordingly, larger energy saving can be expected.

recovery overheads are out of the scope of this paper.

3.3 Discussion

Here we would like to briefly discuss our contribution to the total energy consumed by the secondary-site storage. Table 1 shows top-five systems quoted from "Top Ten TPC-C by Performance Version 5 Results". Four systems used only 2.2-5.6% of all the disk drives for database log, and the other system used only 11.6% for log. That is, in the secondary site, only very few disk drives must be always spinning actively and the other disk drives can be spun down by the combination of deferred log application and eager log compaction. The contribution of our idea is still significant on the whole storage system in the secondary site.

Although this section discusses the problem mainly based on the database log forwarding, the proposed method can be easily extended to other remote replication methods such as logical database replication and physical block forwarding. Specifically, eager compaction technique can be directly applied to physical block forwarding. Forwarded blocks can be folded and sorted similarly based on physical block address. On the other hand, slight modification is necessary for logical database replication, since queries and transactions described in SQLs are not aware of physical addresses. Queries and transactions should be scheduled with assistance of batch query scheduling techniques [21, 24]. So far, such compaction techniques of updated information were intended for reducing inter-site traffic [17]. In contrast, our attempt is focused on deriving long idle period for energy saving.

4 Evaluation

This section presents validating experiments using TPC-C benchmark, showing that disk power consumption of the secondary site can be saved with slight degradation of the quality of business continuity. Online transactions

Table 2: Basic parameters of disk drive models.

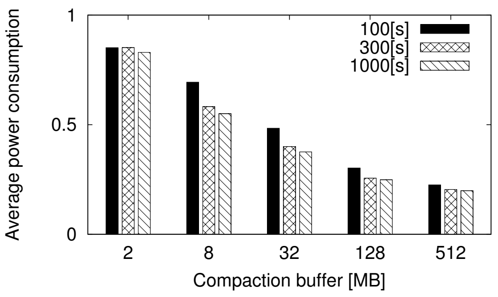
Model	IBM Ultrastar 36Z15	HGST Deskstar T7K250
Capacity	18.4 GB	250 GB
Rotational speed	15000 rpm	7200 rpm
Avg. seek time	3.4 ms	8.5 ms
Transfer rate	55 MB/s	61 MB/s
Active power	39.0 W	9.7 W
Idle power	22.3 W	5.24 W
Standby power	4.15 W	(U) 4.04 W (L) 2.72 W (N) 0.93 W
Spin-down penalties (time and energy)	15.0 s 62.25 J	(U) 0.7 s, 3.5 J (L) 17.0 s, 19.0 J (N) 0.7 s, 3.5 J
Spin-up penalties (time and energy)	26.0 s 904.8 J	(U) 0.7 s, 3.5 J (L) 17.0 s, 19.0 J (N) 0.7 s, 3.5 J

(U): unloaded mode, (L): low-rpm mode, (N): non-spinning mode

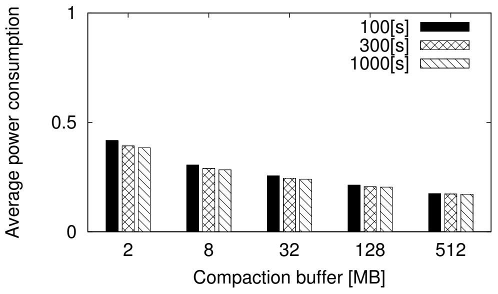
are typical workloads that are seen in enterprise-level disaster recovery systems.

We prepared a hybrid simulation environment for measuring the potential energy saving due to the proposed system. In the experiment, we used a disk drive simulator which can calculate energy consumption based on a disk drive model. We implemented deferred log application and eager log compaction on the top of the disk drive simulator. The developed log applier can apply database log generated by HiRDB [15], a commercial database system.

The experiments were done on a Linux server with dual Xeon processors and 2GB main memory. We set up TPC-C benchmark with 16 and 160 warehouses respectively, and we generated database log on each configuration by executing one million transactions using HiRDB plus 512MB database buffer with no think time. At this execution, we also traced IO behavior by using a kernel-level IO tracer. Then, by replaying the traced IOs using a disk drive model in the simulation environment, we simulated log generation at the primary site. Here, we assumed that the primary site processed transactions at the maximum rate on the specified disk drive model. Next, we applied the generated database log by the log applier, and measured the power reduction effect at the secondary site. Throughout this experiment, we followed the storage system configuration of "IBM System p5 595" in Table 1. That is, we assumed that 94.4% of disk drives were used for the data volume and the same type of disk storage was used both in the primary and secondary sites. The experiments were conducted for different RTO requirements and different window buffer lengths. For validation, we compared the energy saving of the proposed power-aware system with the conventional system in which the transferred update information is immediately reflected.



(a) 16 warehouses



(b) 160 warehouses

Figure 3: Power saving of secondary-site disk storage with high-end disk drives.

Table 3: Batch interval for 160 warehouses and high-end disks under 100 seconds of RTO requirement.

Window buffer	32 MB	128 MB	512 MB
Batch interval	536 s	1070 s	2150 s

Figure 3 summarizes the results obtained with a high-end disk drive model. Basic parameters of the model are presented in Table 2. This model, which is based on IBM Ultrastar 36Z15, may not be new, but has been used in many previous papers [4, 20, 33, 34]. In the graphs, each bar, denoting the average power consumption of the disk storage in the secondary-site storage, is normalized by that of the conventional system. Larger window buffer could accelerate the log application throughput more, and accordingly, greater power saving was gained. Note that, by using 512 MB window buffer, which is as large as the database buffer of the primary site, 85% of the power could be conserved totally in the secondary-site storage. Such great saving was supported by accelerated log application; $\frac{R_{apl}}{R_{gen}}$ could speed up to 20.5 (W=16) and 49.3 (W=160) at maximum by eager log compaction. On the other hand, more tolerant RTO requirements could lead to more energy saving, but its contribution was slight. In our experiments, only short RTOs (such as 30 seconds) failed because RTOs were shorter than time penalties of spinning up and down the volume, but moderate RTOs (100 seconds and more) could conserve the energy so much.

Let us consider the batch interval T_{wnd}. Frequent transition of energy modes affects the life time of disk drives. Table 3 summaries sampled values of T_{wnd}. With small buffer, the data volume had to change its modes frequently, but with as large buffer as 512 MB, the disk mode changes only 40 times a day. Note that this frequency was given when the primary site generates database log at top speed. Thus, it looks almost acceptable, since many high-end disk drives support at least 50,000 cycles of starts/stops. Of course, more tolerant RTO gives larger intervals. This analysis reveals that eager compaction is a key technique to the longevity of disk drives.

We conducted the experiments using a recent mid-range disk drive, which has new energy-efficient features [14]. Basic parameters of the model are presented in Table 2 too. The disk drive, which is based on HGST Deskstar T7K250, has three standby states: unload, low-rpm and non-spinning (equal to conventional standby mode). Basically, the proposed disaster recovery system could work with mid-range disk drives similarly. But, since recent mid-range disk drives can change the energy modes with much smaller time penalties than high-end disk drives, the proposed system could work for such small RTOs as 30 seconds. Figure 4 compares these three standby modes with 160 warehouses. By using the non-spinning standby mode, 80% saving was gained at maximum in comparison with the conventional system. However, we cannot observe the substantial benefit of using new energy-saving functions such as unload and low-rpm.

In summary, the proposed power-aware disaster recovery system can achieve great energy saving of the secondary-site disk storage without little harm to the recovery capability. Only 100 seconds and 30 seconds of RTO allowance were needed for high-end disks and mid-range disks respectively. This observation is surprising, since strict high-availability systems, as known as five nines (99.999%), allow only 315 seconds of breakdown per year. Our proposal can be promising in such top-drawn disaster recovery systems.

5 Related works

Research communities have presented various attempts for energy-efficient disk storage.

The simplest approach is to transition disk drives to a low-power mode after the predetermined time period has elapsed after the last disk access. This technique is widely deployed in commercial disk drives. More sophisticated techniques that try to tune the threshold adaptively have been also studied [6, 11]. Such threshold based techniques work effectively for battery-operated mobile and laptop computers. However, it looks difficult to directly apply these techniques to enterprise systems.

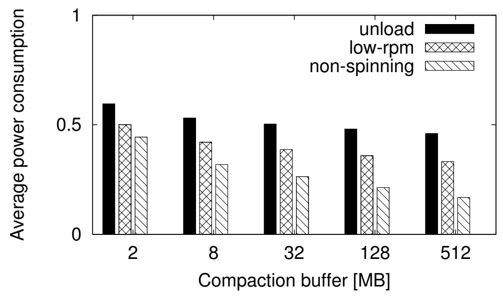


Figure 4: Power saving of secondary-site disk storage with mid-range disk drives under 30 seconds of RTO requirement.

Massive Array of Idle Disks (MAID) [5] and Popular Data Concentration (PDC) [4] are alternative approaches that migrate/replicate popular blocks on specific disk drives to create long idle period of the other disk drives. These techniques leveraging access locality are deployed in real archival storage systems.

Exploiting redundancy information and large cache space that RAID capability holds seems a reasonable approach. Energy Efficient RAID (EERAID) [20] and RIMAC [33] can arrange IO requests at RAID controllers so as to avoid evicting out blocks that are originally stored in spun-down disk drives as much as possible. Power-Aware RAID (PARAID) [30] introduces an asymmetric parity placement on the legacy RAID-5 so that the system can dynamically change the number of actively spinning disk drives.

Other researchers [12, 34] have actively studied on multi-speed disk drives which have the capability of changing the rotational speeds. These attempts look very effective. However, to our knowledge, such multi-speed disk drives are still limited in experimental prototypes and not yet seen in the market.

Recently several application-assisted approaches for storage energy conservation have been reported. Cooperative IO [22, 31] is a set of power-aware IO system calls, by which the user can specify deferability and abortability to each IO. Compiler-based application transformation [10, 13, 28] tries to arrange IO commands in source code levels in order to concentrate IO requests.

Our work differs from these previous works in that we are trying to fully leverage the characteristics of the secondary-site disk storage. That is, the disk storage there manages only copies and its resources are not necessarily busy when the primary site is alive. Our eager strategy of concentrating database log can provide long idleness to many disk drives, accordingly obtaining substantial energy saving opportunities.

6 Conclusion

The paper proposes a power-aware disaster recovery system, in which the reflection of transferred updated information can be deferred through eager compaction technique, so as to gain great energy saving of storage systems in the remote secondary site. Experiments with a commercial database system showed that 80-85% energy consumption can be conserved in the secondary-site disk storage with small penalties of possible service breakdown time.

In this paper, we focus on the energy consumption of disk drives which are main components of recent disk storage. Further, we would like to extend our approach so as to provide a system-wide analysis considering RAID controllers and cache memory.

Acknowledgement

This work has been supported in part by *Development of Out-of-Order Database Engine*, a joint research project between the University of Tokyo and Hitachi Ltd., which started in 2007 as a part of the Next-generation IT Program of the Ministry of Education, Culture, Sports, Science and Technology (MEXT) of Japan.

References

[1] APC. Determining Total Cost of Ownership for Data Center and Network Room Infrastructure. White paper, 2002.

[2] B. Rudolph. Storage In an age of Inconvenient Truths. Storage Network World Spring 2007, 2007.

[3] British Standards Institution. BS25999: Business Continuity Managemenet, 2006.

[4] Carrera, E. V., Pinheiro, E., and Bianchini, R. Conserving Disk Energy in Network Servers. In *Proc. Int'l Conf. on Supercomputing* (2003), pp. 86–97.

[5] Colarelli, D., and Grunwald, D. Massive Arrays of Idle Disks for Storage Archive. In *Proc. ACM/IEEE Conf. on Supercomputing* (2002), pp. 1–11.

[6] Douglis, F., Krishnan, P., and Bershad, B. Adaptive disk spin-down policies for mobile computers. In *Proc. USENIX Symp. on Mobile and Location-Independent Computing* (1995), pp. 121–137.

[7] Eagle Rock Alliance. Online Survey Results: 2001 Cost of Downtime, Contingency Planning Research, 1996.

[8] EMC Corp. Symmetrix Remote Data Facility product description guide, 2000.

[9] F. Moore. More power needed. Energy User News, 2002.

[10] Gniady, C., Hu, Y. C., and Lu, Y.-H. Program Counter-Based Prediction Techniques for Dynamic Power Management. *IEEE Trans. Comput. 55*, 6 (2006), 641–658.

[11] Golding, R. A., Bosch, P., Staelin, C., Sullivan, T., and J.Wilkes. Idleness is not sloth. In *Proc. USENIX Tech. Conf.* (1995), pp. 201–212.

[12] Gurumurthi, S., Sivasubramaniam, A., Kandemir, M., and Franke, H. Reducing Disk Power Consumption in Servers with DRPM. *IEEE Computer 36*, 12 (2003), 59–66.

[13] Heath, T., Pinheiro, E., Hom, J., Kremer, U., and Bianchini, R. Application Transformations for Energy and Power-Aware Device Management. In *Proc. Int'l Conf. on Parallel Arch. and Compilation Tech.* (2002), pp. 121–130.

[14] HGST Inc. Quietly cool. White Paper, HGST, 2004.

[15] Hitachi Ltd. Hitachi Relational Database Management System Solutions for Disaster Recovery to Support Business Continuity. Review Special Issue, Hitachi Technology, 2004.

[16] IDC. The Expanding Digital Universe: A Forecast of Worldwide Information Growth Through 2010. An IDC White Paper sponsored by EMC.

[17] Ji, M., Veitch, A., and Wikes, J. Seneca: remote mirroring done write. In *Proc. USENIX Conf. on File and Storage Tech.* (2003), pp. 253–268.

[18] Kleiman, S. Trends in Managing Data at the Petabyte Scale. Invited talk, USENIX Conf. on File and Storage Tech., 2007.

[19] Lahiri, T., Ganesh, A., Weiss, R., and Joshi, A. Fast-Start: Quick Fault Recovery in Oracle. White paper, 2001.

[20] Li, D., Wang, J., and Varman, P. Conserving Energy in Conventional Disk based RAID Systems. In *Proc. Int'l Workshop on Storage Network Arch. and Parallel I/Os* (2005), pp. 65–72.

[21] Lu, H., and Lee Tan, K. Batch Query Processing in Shared-Nothing Multiprocessors. In *Proc. Int'l. Conf. on Database Syst. for Advanced Applications* (1995), pp. 238 – 245.

[22] Lu, Y., Benini, L., and Micheli, G. Power-aware operating systems for interactive systems. *IEEE Trans. Very Large Scale Integration Syst. 10*, 2 (2002), 119–134.

[23] Maximum Throughput Inc. Power, heat, and sledgehammer. White paper, 2002.

[24] Mehta, M., Soloviev, V., and DeWitt, D. J. Batch Scheduling in Parallel Database Systems. In *Proc. IEEE Int'l. Conf. on Data. Eng.* (1993), pp. 400 – 410.

[25] National Climate Data Center, U.S. DOC. Climate of 2005 Atlamtic Hurricane Season. Online Report available at `http://www.ncdc.noaa.gov/oa/climate/research/2005/hurricanes05.html`, 2005.

[26] National Fire Protection Association. NFPA1600: Standard on Disaster/Emergency Management and Business Continuity Programs (2004 Edition), 2004.

[27] Oracle Corp. Oracle Data Guard 11g. The Next Era in Data Protection and Availability. White paper, 2007.

[28] Son, S. W., Mandemir, M., and Choudhary, A. Software-Directed Disk Power Management for Scientific Applications. In *Proc. IEEE Parallel and Distributed Processing Symp.* (2005), p. 4b.

[29] U.S. SEC. General Rules and Regulations promulgated under the Securities Exchange Act of 1934, 2005.

[30] Weddle, C., Oldham, M., Qian, J., Wang, A. A., Reiher, P., and Kuenning, G. PARAID: A Gear-Shifting Power-Aware RAID. In *Proc. USENIX Conf. on File and Storage Tech.* (2007), pp. 245–260.

[31] Weissel, A., Beutel, B., and Bellosa, F. Cooperative I/O – A Novel I/O Semantics for Energy-Aware Applications. In *Proc. USENIX Symp. on Operating Syst. Design and Imple.* (2002), pp. 117–130.

[32] Worth., S. Green Storage. SNIA Education, 2006.

[33] Yao, X., and Wang, J. RIMAC: A Novel Redundancy-based Hierarchical Cache Architecture for Energy Efficient, High Performance Storage System. In *Proc. EuroSys* (2006), pp. 249–262.

[34] Zhu, Q., Chen, Z., Tan, L., Zhou, Y., Keeton, K., and Wikes, J. Hibernator: Helping Disk Arrays Sleep through the Winter. In *Proc. ACM Symp. on Operating Syst. Principles* (2004), pp. 177–190.

A Linux Implementation Validation of Track-Aligned Extents and Track-Aligned RAIDs

Jin Qian, Christopher Meyers, and An-I Andy Wang
Florida State University, {qian, meyers, awang@cs.fsu.edu}

Abstract

Through clean-slate implementation of two storage optimizations—track-aligned extents and track-aligned RAIDs—this paper shows the values of independent validations. The experience revealed many unanticipated disk and storage data path behaviors as potential roadblocks for wide deployment of these optimizations, and also identified implementation issues to retrofit these concepts to legacy data paths.

1 Introduction

Validation studies are common in science, but less emphasized in computer science, because a rapidly moving field tends to focus on advancing the frontier.

Through a clean-slate Linux implementation of two storage optimization techniques, we aim to demonstrate the values of validations. (1) Existing validations are often implicit when the original contributors extend their work. Therefore, subtle assumptions on the OS platforms, system configurations, and hardware constraints can become obscure over time. Independent validations help identify these roadblocks, to ease the technology transfer for wide adoptions. (2) Independent validations can explore design alternatives to verify the resiliency of a concept to different platforms and hardware generations.

This paper presents a validation study of track-aligned extents [9] and track-aligned RAIDs [10]. Both showed significant performance gains. Our experience shows many unanticipated disk features and interactions along the storage data path, and identifies implementation issues to retrofit these concepts to the legacy data path.

2 Track-aligned Extents

The basic idea of track-aligned extents is that an OS typically accesses disks in blocks, each containing multiple sectors. Therefore, accessing a block can potentially cross a track boundary and incur additional head positioning time to switch tracks. By exploiting track boundaries, the performance of accessing a track size of data can improve substantially [9].

2.1 Original Implementation

Track-aligned extents [9] was built under FreeBSD by modifying FFS [7]. Two methods were proposed to extract disk track boundaries, one from the user space and one via SCSI commands. The track boundaries are extracted once, stored, and imported to FFS at mount times. The FFS free block bitmaps are modified to exclude blocks that cross track boundaries. The FFS prefetching mechanism was modified to stop at track boundaries, so that speculative disk I/Os made for sequential accesses would respect track alignments.

Track-aligned extents rely on disks that support zero-latency access, which allows the tail-end of a requested track to be accessed before the beginning of the requested track content [13]. This feature allows an aligned track of data to be transferred without rotational overhead.

With Quantum Atlas 10K II disks, the measured results showed 50% improvement in read efficiency. Simulated and computed results also demonstrated improved disk response times and support for 56% higher concurrency under video-on-demand workloads.

2.2 Recreating Track-aligned Extents

Recreating track-aligned extents involves (1) finding the track boundaries and the zero-latency access disk characteristics, (2) making use of such information, and (3) verifying its benefits. The hardware and software experimental settings are summarized in Table 1.

Hardware/software	Configurations
Processor	Pentium D 830, 3GHz, 16KB L1 cache, 2x1MB L2 cache
Memory	128 MB or 2GB
RAID controller	Adaptec 4805SAS
Disks tested	Maxtor SCSI 10K5 Atlas, 73GB, 10K RPM, 8MB on-disk cache [6]
	Seagate CheetahR 15K.4 Ultra320 SCSI, 36GB, 8MB on-disk cache [12]
	Fujitsu MAP3367NC, 10K RPM, 37GB, with 8MB on-disk cache [5]
Operating system	Linux 2.6.16.9
File system	Ext2 [4]

Table 1: Experimental settings.

2.3 Extracting Disk Characteristics

User-level scanning: Since the reported performance gains for track alignments are high, conceivably a user-level program can observe timing variations to identify track boundaries. A program can incrementally issue reads, requesting one more sector than before, starting from the 0th sector. As the request size grows, the disk bandwidth should first increase and then drop as the request size exceeds the size of the first track (due to track switching overhead). The process can then repeat, starting from the first sector of the previously found track. Binary search can improve the scheme.

To reduce disturbances caused by various disk data path components, we used the `DIRECT_IO` flag to bypass the Linux page cache, and we accessed the disk as a raw device to bypass the file system. We used a modified `aacraid` driver code to bypass the SCSI controller, and we used `sdparm` to disable the read cache (`RCD=1`) and prefetch (`DPTL=0`) of the disk.

As a sanity check, we repeated this experiment with an arbitrary starting position at the 256th sector (instead of the 0th sector). Additionally, we repeated this experiment with a random starting sector between 0 and 512, with each succeeding request size increasing by 1 sector (512 bytes).

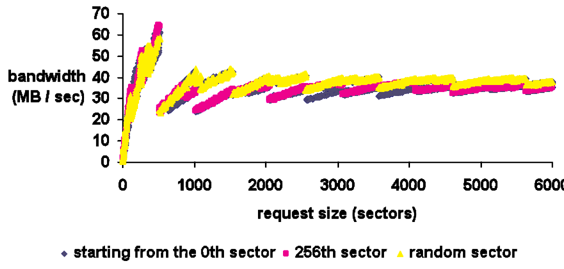


Figure 1: Bandwidth for various read request sizes from varying starting sectors on a Maxtor disk.

Surprisingly, although Figure 1 exhibits bandwidth "cliffs," the characteristic trends are not sensitive to the starting location of requests, suggesting that those cliffs are caused by sources of data misalignments other than tracks. Some possibilities are transfer granularity of the DMA and the management granularity of IO buffers. The graph also suggests the presence of other optimizations that are not disabled. For example, the high bandwidth before the first cliff far exceeds our expected performance gain. [2] conjectures that the DEC prefetch scheme implemented in Maxtor may override specified disk settings at times and proceed with prefetching. Additionally, for certain ranges of request sizes (e.g., between 1,000 and 1,500 sectors), the average bandwidth shows multimodal behaviors.

To verify that those cliff locations are not track boundaries, we wrote a program to access random cliff locations with the access size of 512 sectors (256KB), as indicated by the first cliff location. We ran multiple instances of this program concurrently and perceived no noticeable performance difference compared to the cases where the accesses started with random sectors.

SCSI diagnostic commands: Unable to extract track boundaries from a naive user-level program, we resorted to SCSI `SEND/RECEIVE DIAGNOSTIC` commands to map a logical block address (LBA) to a physical track, surface, and sector number.[1] However, this translation for large drives is very slow, and it took days to analyze a 73-GB drive. We modified the `sg_senddiag` program in the Linux `sg3_utils` package to speed up the extraction process, according to the following pseudocode:

[1] We did not use DIXtrac [8] for the purpose of clean-slate implementation and validation.

```
1. Extract from LBA 0 sector-by-sector until either
   track number or surface number changes. Record
   LBA and the physical address of this track boundary.
   Store the track size S.
2. Add S to the last known track boundary T and
   translate S + T and S + T – 1.
   a. If we detect a track change between S + T and S
      + T – 1, then S + T is a new boundary. Record
      the boundary. Go to step 2.
   b. If there is no change between S + T and S + T –
      1, the track size has changed. Extract sector-
      by-sector from the previous boundary until we
      detect a new track boundary. Record the
      boundary, update S, and go to step 2.
3. If sector reaches the end of the disk in step 2, exit.
```

Through this scheme, we extracted the layout mapping specifics that are not always published in vendors' datasheets and manuals [5, 6, 12] in about 7 minutes.

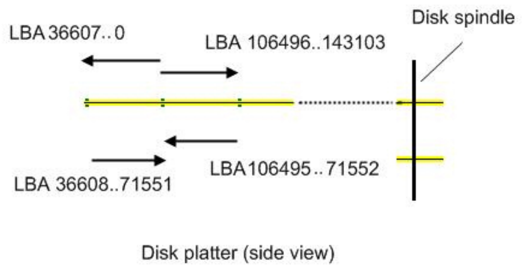


Figure 2: Non-monotonic mapping between LBA and track numbers.

First, the LBA mapping to the physical track number is not monotonic (Figure 2). For the Maxtor drive, LBA 0 starts on track 31 of the top surface and increases outward (from the disk spindle) to track 0, and then the LBA continues from the bottom surface of track 0 inward to track 31. Next, the LBA jumps to track 63 of the bottom surface growing outward to track 32, and then switches back to the top surface's track 32 and continues inward to track 63. The pattern repeats.

Variants of this serpentine numbering scheme [1, 11] are observed in Seagate [12] and Fujitsu [5] drives as well. At the first glace, one might conjecture this numbering scheme relates to the elevator and scanning-based IO schedulers, but this scheme is attributed to the faster timing when switching a head track-to-track on the same surface than when switching to a head on a different surface [11].

Second, the track size differs even for the same disk model from the same vendor, due to the manufacturing process of the disks. After assembly, the disk head frequency response is tested. Disk heads with a lower frequency response are formatted with fewer sectors per track [2]. We purchased 6 Maxtor 10K V drives at the same time and found 4 different LBA numbering schemes (Table 2). The implication is that track extraction needs to be performed on every disk, even those from the same model. Track size may differ in the same zone on the same surface due to defects. Thus, we are no longer able to calculate the track boundary with zone information but have to extract all tracks.

Serial number	Surface 0, outer most track	Surface 1, outer most track
J20 Q3 CZK	1144 sectors	1092 sectors
J20 Q3 C0K/J20 Q3 C9K	1092 sectors	1144 sectors
J20 TK 7GK	1025 sectors	1196 sectors
J20 TF S0K/J20 TF MKK	1060 sectors	1170 sectors

Table 2: Track sizes of Maxtor 10K V drives.

Verifying track boundaries: To verify track boundaries, we wrote a program to measure the elapsed time to access 64 sectors with shifting offsets from random track boundaries. The use of 64 sectors eases the visual identifications of boundaries. We measured tracks only from the top surface within the first zone of a Maxtor disk, so we could simplify our experiment by accessing a mostly uniform track size of 1,144 sectors.

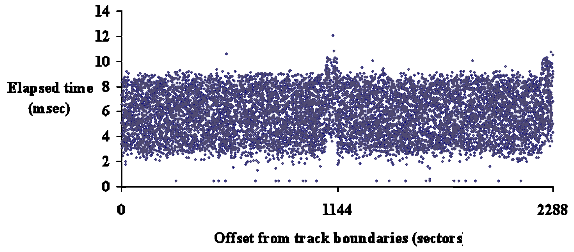


Figure 3: Elapsed time to access 64 sectors, starting from different offsets from various track boundaries on a Maxtor drive (the track size is 1,144 sectors).

Figure 3 confirms extracted track boundaries. Each data point represents the time to access a 64-sector request starting from a randomly chosen sector offset from a track boundary. The 6-msec timing variation reflects the rotation delay for a 10,000 RPM drive. The average elapsed time for accessing 64 sectors across a track boundary is 7.3 msec, compared to 5.7 msec for not crossing the track boundaries. Interestingly, the difference of 1.6 msec is much higher than the track switching time of 0.3 to 0.5 msec [6]. We also verified this extraction method with other vendor drives. The findings were largely consistent.

Zero-latency feature verification: Since the effectiveness of track-aligned extents relies on whether a disk can access the data within a track out-of-order, we performed the tests suggested in [13]. Basically, we randomly picked two consecutive sectors, read those sectors in reverse LBA order, and observed the timing characteristics. We performed the test with various caching options on.

As shown in Figure 4, with a Maxtor drive, 50% of the time the second request is served from the on-disk cache, indicating the zero-latency capability. (We did not observe correlations between the chosen sectors and whether the zero-latency feature is triggered.) In contrast, the other two drives always need to wait for a 3- to 6-msec rotational delay before serving the second sector request. For the remainder of the paper, we will use the Maxtor drives.

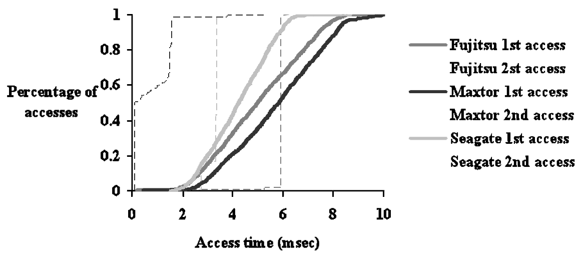


Figure 4: CDF of disk access times for accessing random sets of two consecutive LBAs in the reverse order.

2.4 Exploiting Track Boundaries

The track boundary information can be exploited at different levels.

User level: One possibility is to create a user program to make use of this track information. Similar to the disk defragmentation, instead of moving file blocks to reduce the level of fragmentation, we can move blocks to align with track boundaries. This approach avoids kernel changes and can make files smaller than a track not cross track boundaries, and files larger than a track aligned to track boundaries.

However, this approach needs to overcome many tricky design points. For example, certain blocks are referenced from many places (e.g., hardlinks). Moving those blocks requires tracking down and updating all references to the block being moved. Such information might not be readily available.

File system level: We can mark certain sectors as bad so a file system cannot allocate blocks that consist of sectors across track boundaries. However, this method does not prevent a track-size file from being allocated across two tracks. This approach also anticipates some bandwidth loss when a single IO stream accesses multi-track files due to unused sectors. However, when a system is under multiple concurrent IO streams, the performance benefits of accessing fewer tracks when multiplexing among streams can outweigh the performance loss.

Implementation: We implemented track-aligned extents in ext2 [4] under Linux. First, we used the track boundary list extracted by the SCSI diagnostic commands as the bad-block list input for the `mke2fs` program, which marks all of these blocks, so that they will not be allocated to files. We also put this list in a kernel module along with two functions. One initializes and reads the list from user space. The other is used by different kernel components to find a track boundary after a given position.

We then modified the ext2 pre-allocation routine to allocate in tracks (or up to a track boundary). One disadvantage of this approach is over-allocation, but the unused space can later be returned to the system. However, should the system anticipate mostly track-size accesses, we are less concerned with the wasted space. For instance, database and multimedia applications can adjust their access granularity accordingly. With the aid of this list, we can also change the read-ahead to perform prefetches with respect to track boundaries.

Our experience suggests that individual file systems only need to make minor changes to benefit from track alignments.

2.5 Verification of the Performance Benefits

We used the sequential read and write phases of the Bonnie benchmark [3], which is unaware of the track alignments. The write phase creates a 1-GB file, which exceeds our 128-MB memory limit. We enabled SCSI cache, disk caching, and prefetch to reflect normal usage. Each experiment was repeated 10 times, analyzed at a 90% confidence interval.

Figure 5 shows the expected 3% slowdown for a single stream of sequential disk accesses, where skipped blocks that cross track boundaries can no longer contribute to the bandwidth.

We also ran `diff` from GNU `diffutils` 2.8.1 to compare two 512-MB large files via interleaved reads between two files, with the `–speed-large-files` option. Without this option, `diff` will try to read one entire file into the memory and then the other file and compare them if memory permits, which nullifies our intent of testing interleaved reads. Figure 6 shows that track-aligned accesses are almost twice as fast as the normal case. In addition, we observed that disk firmware prefetch has no regard for track boundaries. Disabling on-disk prefetch further speeds up track-aligned access by another 8%. Therefore, for subsequent experiments, we disabled disk firmware prefetch for track-aligned accesses.

Additionally, we conducted an experiment that involves concurrent processes issuing multimedia-like traffic streams at around 500KB/sec. We used 2GB for our memory size. We wrote a script that increases the number of streams by one after each second, and the script records the startup latency of each new stream. Each emulated multimedia streaming process first randomly selects a disk position and sequentially accesses the subsequent blocks at the specified streaming rate. We assumed that the acceptable startup latency is around 3 seconds, and the program terminates once the latency reaches 3 seconds.

Figure 7 shows that the original disk can support up to 130 streams with a startup latency within 3 seconds. A track-size readahead window can reduce the latency at 130 streams by 30%, while track-aligned access can reduce the latency by 55%.

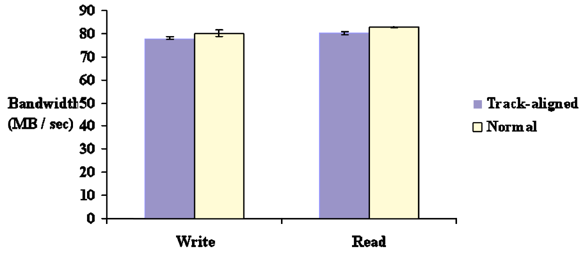


Figure 5: Bandwidth comparisons between conventional and track-aligned accesses to a single disk, when running the Bonnie benchmark.

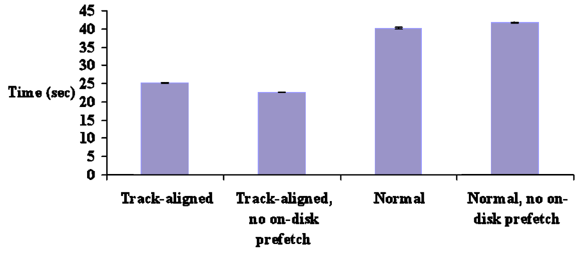


Figure 6: Speed comparisons between conventional and track-aligned accesses to a single disk, `diffing` two 512MB files with 128MB of RAM.

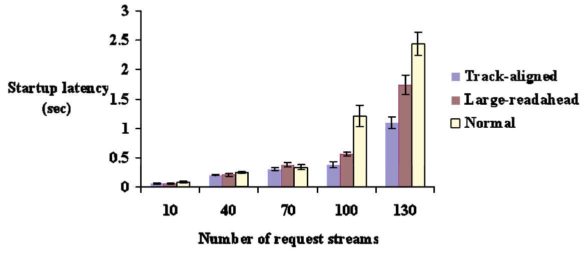


Figure 7: Startup latency comparisons of conventional I/O requests, requests with a one-track prefetch window, and track-aligned requests on a single disk, with a varying number of multimedia-like request streams.

3 Track-aligned RAIDs

Original implementation: Schindler et al [10] proposed Atropos, a track-aligned RAID. The implementation was through a user-level logical volume manager process. The process bypasses conventional storage data paths and issues raw IOs. An application needs to be linked with a stub library to issue reads and writes. The library uses shared memory to avoid data copies and communicates with Atropos through a socket.

Without the conventional storage data path, Atropos is responsible for scheduling requests with the help of a detailed disk model. Atropos also needs to duplicate logics provided by conventional RAID levels. As a proof of concept, the measured prototype implemented RAID-0 (no redundancy) and RAID-1 (mirroring), although issues relevant to other RAID levels are addressed in the design.

To handle different track sizes due to disk defects, for simplicity Atropos skips tracks that contain more than a threshold number of defects, which translates to about 5% of storage overhead.

The performance for track-aligned RAIDs matches the efficiency expectation of track-aligned extents.

Recreating Track-aligned RAIDs: Our clean-slate validation implements track-aligned RAIDs via modifying RAID-5 (distributed parity), retrofitting the conventional storage data path. Thus, unmodified applications can enjoy the performance benefit as well. However, we had to overcome a number of implementation constraints.

Recall from Section 2.3 that the track sizes can differ even from the same disk model. This difference was much more than that caused by defects. Therefore, we need measures beyond skipping tracks. For one, we can construct stripes with tracks of different sizes. Although this scheme can work with RAID-0, it does not balance load well or work well with other RAID levels. For example, RAID-5 parity is generated via XORing chunks (units of data striping) of the same size. Suppose we want the chunk unit to be set to the size of a track. If we use the largest track size as the chunk unit, some disks need to use 1+ tracks to form a chunk. Or we can use the smallest track size as the chunk unit, leading to about 10% of unused sectors for disks with larger track sizes.

Additionally, we observed that parity in RAIDs can interact poorly with prefetching in the following way. Take RAID-5 as an example. At the file system level, prefetching one track from each non-parity disk involves a prefetching window that is the size of a track multiplied by the number of disks that do not contain the parity information. However, as a RAID redirects the contiguous prefetching requests from the file system level, the actual forwarded track-size prefetching requests to individual disks are fragmented, since reads in RAIDs do not need to access the parity information.

Another poor interaction is the Linux plug and unplug mechanisms associated with disk queues and multi-device queues. These mechanisms are designed to increase the opportunities for data reordering by introducing artificial forwarding delays at times (e.g., 3 msec), and do not respect track boundaries. Therefore, by making these mechanisms aware of track boundaries, we were finally able to make individual disks in a RAID-5 access in a track-aligned manner.

Implementation: We modified Linux software RAID-5 to implement the track-aligned accesses. We altered the `make_request` function, which is responsible for translating the RAID virtual disk address into individual disk addresses. If the translated requests crossed track boundaries, the unplug functions for individual disk queues were explicitly invoked to issue track-aligned requests.

To prevent the parity mechanisms from fragmenting track-size prefetching requests, we modified RAID-5. Whenever the parity holding disk in a stripe was the only one not requested for that stripe, we filled in the read request for that disk and passed it down with all others. When this dummy request was completed, we simply discarded the data. The data buffer in Linux software RAID-5 is pre-allocated at initialization, so this implementation does not cause additional memory overhead.

Verification of performance benefits: We compared the base case RAID-5 with a track-aligned RAID-5 with five disks, and a chunk size of 4KB. For the Bonnie benchmark, we used a 1-GB working set with 128MB of RAM. Figure 8 shows that the write bandwidth for the three system settings falls within a similar range due to buffered writes. However, for read bandwidth, the track-aligned RAID-5 outperforms the conventional one by 57%.

The `diff` experiment compared two 512-MB files with 128MB of RAM. Figure 9 shows that the track-aligned RAID-5 can achieve a 3x factor speedup compared to the original RAID-5.

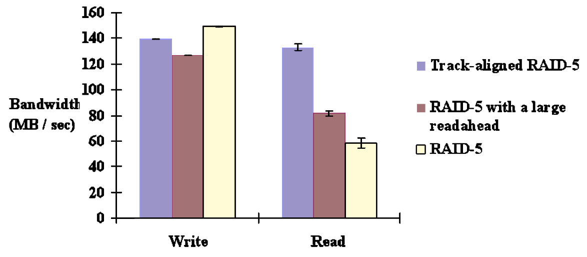


Figure 8: Bandwidth comparisons of the track-aligned RAID-5, a RAID-5 with a prefetch window of four tracks, and the original RAID-5, running Bonnie with 1GB working set and 128MB of RAM.

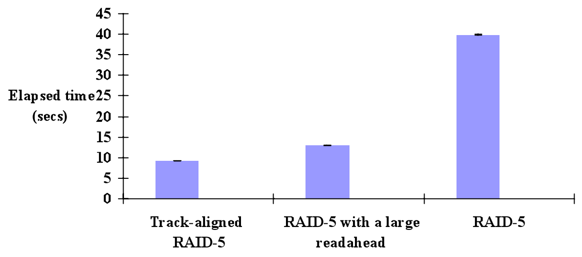


Figure 9: Elapsed time comparisons of the track-aligned RAID-5, a RAID-5 with a prefetch window of four tracks, and the original RAID-5, when running `diff` comparing two 512MB files.

For the multimedia-like workload with 2GB of RAM, the track-aligned RAID-5 demonstrates a 3.3x better scaling in concurrency than the conventional RAID-5 (Figure 10), where a RAID-5 with a readahead window comparable to the track-aligned RAID-5 contributes only less than half of the scaling improvement. The latency improvement of track-aligned RAID-5 is impressive considering that the RAID-5 was expected to degrade in latency when compared to the single-disk case, due to the need to wait for the slowest disk for striped requests. Track-aligned accesses reduce the worst-case rotational timing variance and can realize more benefits of parallelism.

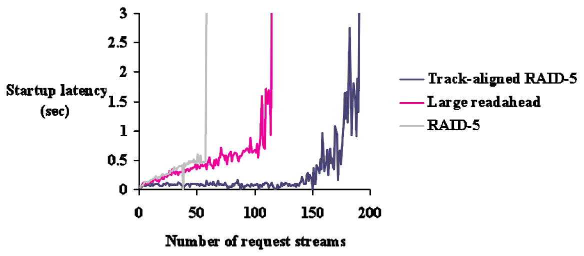


Figure 10: Startup latency comparisons of the track-aligned RAID-5, a RAID-5 with a prefetch window of four tracks, and the original RAID-5, with a varying number of multimedia-like request streams.

4 Lessons Learned and Conclusions

Through clean-slate implementations of track-aligned extents and track-aligned RAIDs, we have demonstrated important values of independent validations. First, the validation of research results obtained five years ago shows the relative resiliency and applicability of these concepts to different platforms and generations of disks. On the other hand, as the behaviors of disks and the legacy storage data path become increasingly complex, extracting physical disk geometries will likely become increasingly more difficult. Also, as disks become less homogeneous even within the same model, techniques such as track-aligned RAIDs need to devise additional measures to prevent a RAID from being limited by the slowest disk.

Second, through exploring design and implementation alternatives, we revealed many unanticipated interactions among layers of data path optimizations. On-disk prefetching, IO scheduling and aggregation, RAID parity, file system allocation, and file system prefetching—all have side effects on IO access alignment and profound performance implications. Unfortunately, the interfaces among data path layers are lacking in expressiveness and control, leading to modifications of many locations to retrofit the concepts of access alignment into the legacy storage data path, the remedy for which is another fruitful area of research to explore.

Acknowledgements

We thank Mark Stanovich and Adaptec for helping us bypass some RAID controller features. We also thank Peter Reiher and Geoff Kuenning for reviewing this paper. This research is sponsored by NSF CNS-0410896 and CNS-0509131. Opinions, findings, and conclusions or recommendations expressed in this document do not necessarily reflect the views of the NSF, FSU, or the U.S. government.

References

[1] Anderson D. You Don't Know Jack about Disks. *Storage*. 1(4), 2003.

[2] Anonymous Reviewer, reviewer comments, *the 6th USENIX Conf. on File and Storage Technologies*, 2007.

[3] Bray T. Bonnie benchmark. http://www.textuality.com/bonnie/download.html, 1996.

[4] Card R, Ts'o T, Tweedie S. Design and Implementation of the Second Extended Filesystem. *The HyperNews Linux KHG Discussion*. http://www.linuxdoc.org, 1999.

[5] Fujitsu MAP3147NC/NP MAP3735NC/NP MAP3367NC/NP Disk Drives Product/Maintenance Manual. http://www.fujitsu.com/downloads/COMP/fcpa/hdd/discontinued/map-10k-rpm_prod-manual.pdf, 2007.

[6] Maxtor Atlas 10K V Ultra320 SCSI Hard Drive. http://www.darklab.rutgers.edu/MERCURY/t15/disk.pdf, 2004.

[7] McKusick MK, Joy WN, Leffler SJ, Fabry RS. A Fast File System for UNIX, *Computer Systems*, 2(3), pp. 181-197, 1984.

[8] Schindler J, Ganger GR. Automated Disk Drive Characterization. CMU SCS Technical Report CMU-CS-99-176, December 1999.

[9] Schindler J, Griffin JL, Lumb CR, Ganger GR. Track-Aligned Extents: Matching Access Patterns to Disk Drive Characteristics. *Proc. of the 1st USENIX Conf. on File and Storage Technologies*, 2002.

[10] Schindler J, Schlosser SW, Shao M, Ailamaki A, Ganger GR. Atropos: A Disk Array Volume Manager for Orchestrated Use of Disks. *Proc. of the 3rd USENIX Conf. on File and Storage Technologies*, 2004.

[11] Schlosser SW, Schindler J, Papadomanolakis S, Shao M, Ailamaki A, Faloutsos C, Ganger GR. On Multidimensional Data and Modern Disks. *Proc. of the 4th USENIX Conf. on File and Storage Technology*, 2005.

[12] Seagate Product Manual: CheetahR 15K.4 SCSI. http://www.seagate.com/staticfiles/support/disc/manuals/enterprise/cheetah/15K.4/SCSI/100220456d.pdf, 2007.

[13] Worthington BL, Ganger GR, Patt YN, Wilkes J. On-line Extraction of SCSI Disk Drive Parameters. *ACM Sigmetrics*, 1