Reducing Read Latency of Shingled Magnetic Recording With Severe Intertrack Interference Using Transparent Lossless Data Compression

Kalyana Sundaram Venkataraman[1], Guiqiang Dong[1], Ningde Xie[2], and Tong Zhang[1]

[1]Department of Electrical and Computer Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180 USA
[2]Storage Technology Group, Intel, Hillsboro, OR 97124 USA

With the distinct advantage of retaining conventional head and media, the emerging shingled recording technology improves areal storage density through intentional track overlapping that nevertheless introduces severe intertrack interference (ITI). An economically tenable option for shingled drives is to utilize a single read head. As we continue to increase its areal storage density, there will be a higher probability that a read operation demands reading multiple adjacent tracks for explicit ITI compensation. This directly results in a significant read latency penalty when using a single read head. In this work, we propose a simple design strategy to reduce such ITI-induced read latency penalty. If a sector of user data can be compressed to a certain extent, it will leave more storage space for coding redundancy and, hence, opportunistically enable the use of a stronger-than-normal error correction code (ECC). The stronger ECC can accordingly reduce the probability of reading multiple adjacent tracks for explicit ITI compensation. Beyond a simple intrasector compression, the absence of the update-in-place feature in shingled recording makes it feasible to apply lossless compression across multiple consecutive sectors. This can further improve the compression efficiency and, hence, reduce the probability of reading multiple adjacent tracks. We carried out simulations that successfully demonstrate the effectiveness of the proposed design strategies on reducing the read latency penalty caused by severe ITI in shingled recording.

Index Terms—**Lossless compression, read latency, shingled magnetic recording.**

I. Introduction

SHINGLED recording technology [1], [2] has recently emerged as one promising near-term option to continue the historical scaling of magnetic recording storage areal density. Different from other alternative technologies, such as energy-assisted magnetic recording and bit-patterned media recording, shingled recording uses the conventional head and media for recording and relies on a well-controlled track overlap to increase the storage areal density. With a much tighter track pitch, shingled recording is subject to significant intertrack interference (ITI). Although a true 2-D read channel can effectively compensate severe ITI, it could result in higher costs due to the use of multiple read heads [3], [4]. Therefore, this work is only interested in shingled recording with a single read head. Shingled recording with a single read head is fundamentally subject to a storage density versus read latency tradeoff: To read one sector, we first read only the sector itself by treating ITI from its adjacent tracks as random noise, if this first attempt fails, we progressively read one or more adjacent tracks to explicitly compensate ITI in order to correctly recover the target sector. As we use a more aggressive track overlap to increase the areal storage density, ITI will become more significant and, hence, result in a higher probability to read one or more adjacent tracks for explicit ITI compensation. This clearly leads to a longer read latency.

In this work, we are interested in reducing the read latency degradation caused by significant ITI in shingled recording, whereas Venkataraman *et al.* [5] consider techniques primarily addressing the update-in-place problem present in shingled drives. The basic idea is very intuitive and can be briefly described as follows: First, we note that, except for multimedia files, most files are typically stored on disks without compression, even though they may be losslessly compressible. If one sector of user data can be losslessly compressed to a certain extent, extra storage space will become available to store error-correction code (ECC) redundancy, which makes it possible to use a stronger-than-normal ECC for the present sector and, hence, reduce the probability of reading one or more adjacent tracks for explicit ITI compensation. In contrast to a conventional file or disk compression, such an intrasector lossless data compression and decompression are hidden within the disk drive and thereby completely transparent to the user and operating systems. Clearly, it does not change the effective disk storage capacity.

Moreover, since lossless data compression efficiency tends to improve as data size increases, we further investigate the scenario of applying lossless data compression across multiple physically consecutive sectors along the same track in order to opportunistically enable an even stronger ECC and, hence, more effectively reduce the read latency overhead. This is referred to as *virtual sector compression*. Ironically, the loss of update-in-place feature in shingled recording makes such a virtual sector compression strategy practically appealing. For conventional magnetic recording that can naturally support update-in-place, such virtual sector compression is not pragmatic, since we have to read and write back multiple sectors instead of simply writing one sector, leading to an update latency penalty. In sharp contrast, shingled recording loses the update-in-place feature due to the overlapped write. As a result, when one sector is to be updated, we always have to read and write back many consecutive sectors on several adjacent tracks, which naturally enables the use of virtual sector compression without causing any extra update latency. In the context of virtual sector compression, we further discuss and evaluate two different implementation

Manuscript received February 28, 2012; revised June 21, 2012, October 04, 2012, and December 11, 2012; accepted January 16, 2013. Date of publication January 23, 2013; date of current version July 23, 2013. Corresponding author: K. S. Venkataraman (e-mail: venkak2@rpi.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMAG.2013.2242086

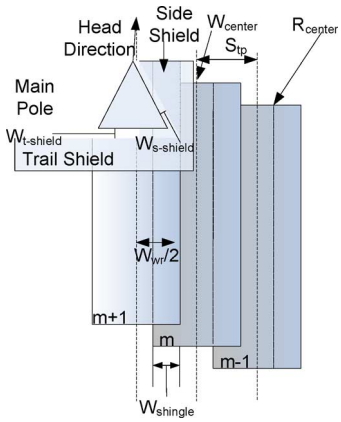

Fig. 1. Illustration of shingled magnetic recording with corner writer. Tracks m-1, m, and m+1 are sequentially written to the disk in the overlapped manner.

strategies, including: 1) fixed-size virtual sector, where we always apply lossless data compression across the same number of a sectors; and 2) content-aware virtual sector, where we dynamically adjust the virtual sector size according to the type of files (hence, their content characteristics).

Finally, we note that a recent related work [6] combines endurance coding with lossless compression to improve the NAND flash memory program/erase cycling endurance. The common feature of this work and [6] is to exploit lossless compressibility to create extra redundant space for improving data coding, whereas this work uses the extra redundancy for stronger ECC and [6] uses the extra redundancy for better endurance coding (notice that endurance coding aims to manipulate the data pattern written to flash memory cells in order to reduce the physical memory cell wear-out). In addition, this work proposes both intrasector and intersector compression for shingled recording, while only intrapage compression is feasible for NAND flash memory (since interpage compression will cause significant read latency overhead in NAND flash memory).

II. Shingled Write and Random Read Latency

Shingled magnetic recording works on the principle of overlapped writing and requires a corner writer to achieve high field gradients [7]. Fig. 1 illustrates the basic principle of shingled writing, where W_{Shingle} denotes the adjacent track width overwritten by shingling, W_{center} and R_{center} are the track centers for writing and reading, S_{tp} denotes the shingled track pitch, and $W_{s-\mathrm{shield}}$ and $W_{t-\mathrm{shield}}$ are the shield gap widths. Although a true 2-D read channel signal processing enabled by multiple read heads can naturally mitigate the severe ITI in shingled magnetic recording [7]–[9], it apparently suffers from higher cost. Hence, this work only considers shingled recording with a single read head.

As we aggressively push the shingled recording areal density at the cost of increasingly severe ITI, a natural progressive three-step read scheme for random read is needed to explicitly compensate ITI, as illustrated in Fig. 2: Using the single read head, we first read the main track and carry out a traditional 1-D read channel signal processing by treating ITI as random noise; if it fails, we read its adjacent track on one side and carry

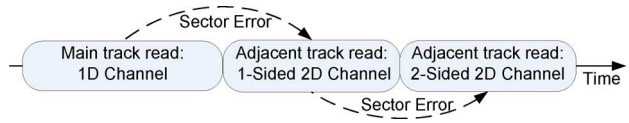

Fig. 2. Illustration of the progressive three-step read scheme to provide just-enough read channel signal processing.

out a 2-D read channel signal processing by explicitly incorporating one-sided ITI information; if it fails again, we read its adjacent track on the other side and carry out a two-sided 2-D read channel signal processing. To quantitatively demonstrate the progressive performance improvement of this three-step read scheme, we carried out the following simulations. According to [10] and [11], we set the Gaussian read sensitivity function for shingled magnetic recording channel response as

$$h(t,\lambda)=\exp\left[\left(\frac{c}{2}\right)\left(\left(\frac{t}{PW_{50}^{a}}\right)^{2}+\left(\frac{\lambda}{PW_{50}^{c}}\right)^{2}\right)\right] \quad (1)$$

where $c=-1.8197$, t represents the down track location, λ is the cross-track location, and PW_{50}^{a} and PW_{50}^{c} denote the half-power width along track and across track, which are set to 5.2 and 4.8 nm, respectively. We set the shingled track pitch S_{tp} as 6.4 nm with W_{Shingle} corresponding to an ITI of 20% for a 70-nm write head, where we consider the adjacent track ITI as the scaled version of the target track response. Fig. 3 shows the read channel model being used in this work. We assume that sectors on different tracks are perfectly aligned, and we do not take into account edge noise and transition noise. As illustrated in Fig. 3, $a_{i,k}$ denotes the bits and $n_i(t)$ denotes additive white Gaussian noise (AWGN) where $i=0,-1,1$ corresponds to the target track and its two adjacent tracks. The impulse response is given by h_i, where i corresponds to the pass number, and G and F denote the generalized partial response (GPR) target and the equalizer whose coefficients are obtained through standard Lagrange constrained minimization [12], [13] for the error $\hat{e}_k$. The filter $h_i(-t,\lambda_0)$ matches the impulse response on the target track. In addition, we set that equalizer has 11 taps. We note that, although Fig. 3 shows the complete 2-D read channel model, it is straightforward to obtain 1-D or one-sided 2-D read channel by removing the responding components. After the equalization, 1-D Viterbi detector is used for signal detection, and we assume the use of Reed–Solomon (RS) code as ECC. We set the codeword length of 4 kB and, thereby, the RS code is constructed over GF(2^{12}).

Fig. 4 shows the sector error rate (SER) versus signal-to-noise ratio (SNR) curves of the three steps, where the SER is defined as the rate at which the ECC decoding of one sector fails: 1-D read channel processing, one-sided 2-D read channel processing, and two-sided 2-D read channel processing. The SER, i.e., RS code decoding failure rate in this study, is estimated using the analytical method presented in [14].

As ITI becomes more significant, the second and even the third step in the three-step progressive read process will more likely occur, leading to a much longer average read latency due to the disk rotational latency in comparison to the seek time. This clearly suggests that it is highly desirable to minimize the

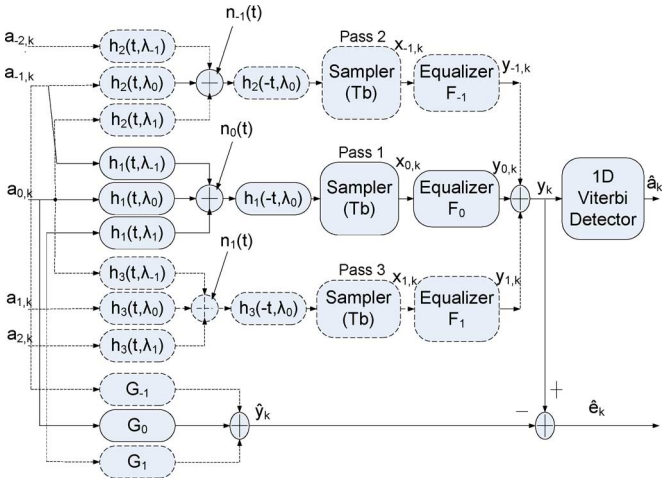

Fig. 3. Illustration of 2-D channel model used for the model.

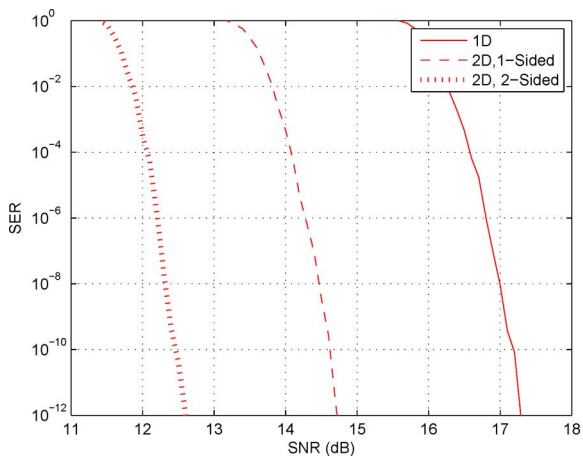

Fig. 4. Simulated SER versus SNR performance of shingled recording with 20% ITI.

occurrence of the second and third steps in this progressive three-step read.

III. Using Transparent Intrasector Data Compression to Reduce Read Latency

In the context of the above three-step progressive read process, let $P_e^{(1)}$ and $P_e^{(2)}$ denote the probability that the first and second read steps fail. We can then express the average read latency as

$$\tau_{\text{lat}} = \tau_{\text{seek}} + P_{\text{seek}} \cdot \tau_{\text{rot}} + \tau_{\text{data}} + (P_e^{(1)} + P_e^{(1)} \cdot P_e^{(2)}) \cdot (\tau_{\text{rot}} + \tau_{\text{data}} + \tau_{\text{sq-seek}}) \quad (2)$$

where τ_{seek} is the seek time, P_{seek} is the probability of rotational delay upon seek, τ_{rot} is the rotational delay for read, τ_{data} is data transfer time, and $\tau_{\text{sq-seek}}$ is the sequential seek time when read head switches from one track to its adjacent track. Given the disk rotation speed, we must reduce $P_e^{(1)}$ and $P_e^{(2)}$ in order to reduce the read latency overhead. Straightforwardly, we can allocate more redundant bits in each sector and, hence, employ a stronger ECC to reduce $P_e^{(1)}$ and $P_e^{(2)}$, which nevertheless suffer areal storage density loss and, hence, bit cost increase.

TABLE I
Simulated Code Rate Reduction Factors for Different Lossless Compressibility

Compressibility β_i	100%	98%	96%	94%
Code rate reduction factor η_i	1	0.9671	0.9434	0.9210

This work is motivated by the fact that, except for the multimedia files, many files stored on hard disk drives are not compressed at all, although they are losslessly compressible. We propose to apply an intrasector transparent lossless compression to opportunistically reduce read latency without incurring areal storage density loss. Given one sector of user data that is losslessly compressible, we apply lossless compression to leave more storage space for ECC coding redundancy within the sector and, hence, enable the use of a stronger ECC to reduce $P_e^{(1)}$ and $P_e^{(2)}$. Since the lossless data compression is applied to each individual sector independently, it is completely transparent to the operating systems and users. We performed the following case studies to quantitatively demonstrate the effectiveness of this simple design strategy.

In this work, we chose the popular Lempel–Ziv–Storer–Symanski (LZSS) lossless data compression algorithm [15], which is applied to compress various user data, executable and system files on one of our department servers with Red Hat enterprise edition Linux. Besides OS and personal files, the server hosts several commercial electronic design automation (EDA) software packages. Let $\eta \in (0, 1]$ denote the code rate reduction factor, i.e., the use of lossless data compression can scale down the ECC code rate by a factor of η. Let β denote the compressibility corresponding to η, i.e., the probability that one sector can be protected by an ECC with the code rate reduction factor of η after the use of LZSS-based lossless data compression. Ideally, in order to fully utilize the available storage space within each sector, we may expect that the ECC code rate reduction can be gracefully adjusted with a fine granularity. However, such an ideal scenario with fine-grained configurability and adaptability may largely complicate the practical implementation. Therefore, we only consider four different compressibility, including $\beta_1 = 100\%$, $\beta_2 = 98\%$, $\beta_3 = 96\%$, and $\beta_4 = 94\%$, based upon which we can obtain the corresponding four code rate reduction factor, as shown in Table I. In this case study, we consider the 4-kB sector format and each sector is protected by an RS code, and a (2797, 2731, 33) RS code is used when lossless compression is not used. According to these code rate reduction factors, we have that the parameters of the other three different RS codes are (2797, 2633, 82), (2797, 2577, 110), and (2797, 2515, 141).

Fig. 5 shows the simulated SER versus SNR performance under the four different code rate reduction factors and different read channel configurations. Based on the simulation results in Fig. 5 and (2), we estimate the read latency versus SNR, as shown in Fig. 6, under different scenarios, which clearly demonstrates the effectiveness of using intrasector data lossless compression strategy to reduce read latency in shingled recording. In

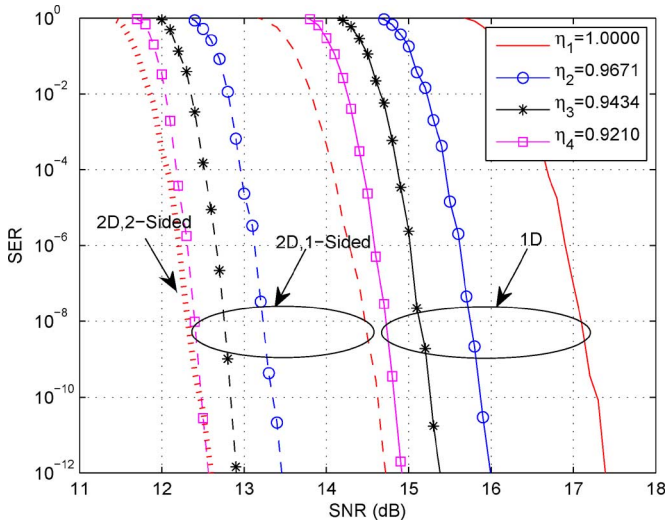

Fig. 5. Simulated SER versus SNR performance when intrasector compression enables different code rate reduction factors.

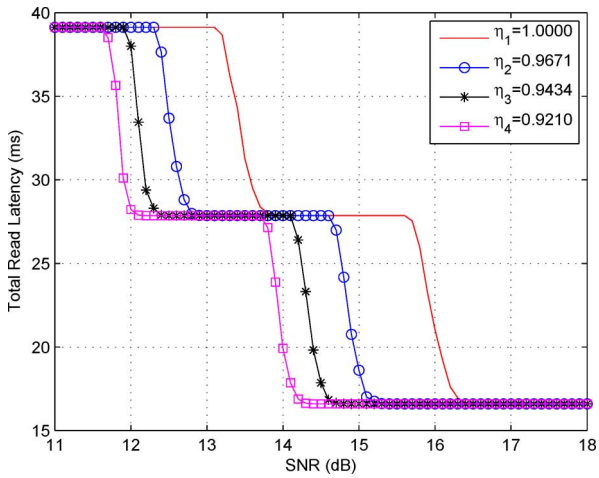

Fig. 6. Estimated read latency versus SNR when intrasector compression enables different code rate reduction factors.

this study, we consider a 3.5-in drive with 5400 revolutions per minute (RPM) and an average seek time of 11 ms. We set a 4-kB sector size, a mean read data rate of 110 MB/s, 250 servosectors, and assume negligible command processing time. Meanwhile, due to the discrete nature of the progressive three-step read scheme, there are certain SNR regions in which different code rate reduction factors result in the same read latency, as shown in Fig. 6.

IV. Virtual Sector Compression

In the above discussion, lossless data compression is applied independently to each individual sector. Being transparent to the firmware, users, and operating systems, such an intrasector compression can maximally simplify the practical implementation. Nevertheless, since lossless data compression efficiency

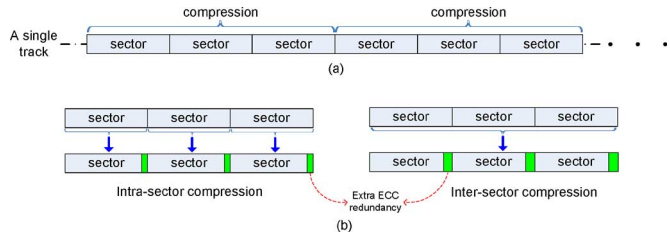

Fig. 7. Illustration of (a) intersector compression, and (b) comparison between intrasector compression and intersector compression.

strongly depends on the data chunk[1] size (i.e., a larger data chunk leads to a higher compression efficiency), such an intrasector compression suffers from relatively poor compression efficiency due to the limited sector size (e.g., 4 kB). Intuitively, we can apply lossless data compression across several physically consecutive sectors on the same track to improve compression efficiency, as shown in Fig. 7. Nevertheless, such intersector compression may be subject to two issues: 1) we have to read and write several sectors in order to update any one sector, leading to an extra update latency overhead; and 2) we have to read several sectors in order to read any one sector, leading to an extra read latency overhead.

In shingled recording, a certain number of tracks form a shingled region, and all the tracks within the same region must be written sequentially due to the overlapped writing. As a result, the update-in-place feature is lost. Ironically, such a fundamental limitation of shingled recording eliminates the first issue mentioned above, i.e., applying lossless data compression across several physically consecutive sectors on one track does not induce any extra update latency overhead. The second issue of the extra read latency overhead involves a tradeoff, which is described as follows. Let $P_e^{(1)}(n)$ and $P_e^{(2)}(n)$ denote the probability that the first and second read steps fail when one virtual sector contains n consecutive sectors. We can express the read latency as

$$\begin{aligned}\tau_{\rm lat}(n) = {} & \tau_{\rm seek} + P_{\rm seek} \cdot \tau_{\rm rot} + \tau_{\rm data}(n) \\ & + (P_e^{(1)}(n) + P_e^{(1)}(n) \cdot P_e^{(2)}(n)) \\ & \cdot \tau_{\rm rot} + \tau_{\rm data}(n) + \tau_{\rm sq-seek} \end{aligned} \tag{3}$$

where $\tau_{\rm seek}$ is the seek time, $P_{\rm seek}$ is the probability of rotational delay upon seek, $\tau_{\rm rot}$ is the rotational delay for read, and $\tau_{\rm data}(n)$ is data transfer time for the n sectors. Given the ECC being used to protect each sector, we can estimate $P_e^{(1)}(1)$ and $P_e^{(2)}(1)$, based upon which we can then estimate $P_e^{(1)}(n)$ and $P_e^{(2)}(n)$ as

$$\begin{aligned} P_e^{(1)}(n) &= 1 - (1 - P_e^{(1)}(1))^n \\ P_e^{(2)}(n) &= 1 - (1 - P_e^{(2)}(1))^n. \end{aligned} \tag{4}$$

As n increases, we can improve the lossless data compression efficiency and, hence, use a stronger ECC with a lower code rate, which can reduce the failure probabilities $P_e^{(1)}(1)$ and $P_e^{(1)}(1)$.

[1]Here *data chunk* is used to represent a group of one or more physical sectors, upon which lossless compression is applied.

TABLE II
SIMULATED CODE RATE REDUCTION FACTOR UNDER DIFFERENT COMPRESSIBILITY AND VIRTUAL SECTOR SIZE

	$n=3$	$n=8$	$n=16$	$n=32$
$\beta_2=98\%$	0.9620	0.9612	0.9591	0.9525
$\beta_3=96\%$	0.9262	0.9254	0.9210	0.9159
$\beta_4=94\%$	0.9019	0.8939	0.8873	0.8770

Therefore, according to (4), the virtual sector read failure probabilities $P_e^{(1)}(n)$ and $P_e^{(2)}(n)$ tend to first reduce and then increase as we keep increasing n. Clearly, we should keep n small enough to ensure the monotonic dependence of $P_e^{(1)}(n)$ and $P_e^{(2)}(n)$ on n. Meanwhile, a larger n will directly result in a longer virtual sector data transfer latency $\tau_{\text{data}}(n)$. Therefore, when using the virtual sector design strategy to apply lossless data compression across multiple sectors, the average random read latency essentially involves a tradeoff that depends on the virtual sector size n.

In the remainder of this section, we present case studies to demonstrate two possible options for implementing the virtual sector concept. The first option is to simply group a fixed number consecutive sectors into a virtual sector, and the second option is to dynamically adjust the size of the virtual sector based upon the data content characteristics.

A. Case Study I: Fixed-Size Virtual Sector

Similar to the above study on intrasector compression in Section III, we consider four different compressibility values, i.e., $\beta_1=100\%$, $\beta_2=98\%$, $\beta_3=96\%$, and $\beta_4=94\%$. Again, applying the popular LZSS lossless compression algorithm to files stored on one of our department servers, we estimate the achievable code rate reduction factor η_i under different virtual sector size, as listed in Table II. We note that n denotes the number of physical sectors contained in each virtual sector.

Given each set of virtual sector size n and compressibility β, we can apply the corresponding code rate reduction factor to estimate the probabilities $P_e^{(1)}(n)$ and $P_e^{(2)}(n)$ under different SNR, and further estimate the read latency according to (3). Fig. 8 shows the estimated read latency under different virtual sector size for compressibility $\beta_4=94\%$. As we increase the virtual sector size, we can improve the lossless compression efficiency, i.e., the code rate reduction factor can be smaller, as shown in Table II. Therefore, the read latency curves with larger virtual sector sizes exhibit high-to-low latency transition at a lower SNR, as shown in Fig. 8. On the other hand, a larger virtual sector size results in a larger data transfer time $\tau_{\text{data}}(n)$ in (3). Hence, the read latency curves with larger virtual sector sizes have larger read latency after the transition. For example, within 12–13 dB of SNR, virtual sector size of $n=32$ corresponds to the longest read latency (around 30 ms) and virtual sector size of $n=1$ corresponds to the shortest read latency (around 26 ms), as shown in Fig. 8.

Using the current design practice (i.e., without using any lossless compression) as the baseline, Fig. 9 shows the average read latency saving under different SNR when using different virtual sector sizes. Results clearly show the impact of virtual sector size on the read latency savings, and the obvious advantage of using virtual sector with $n=3$ over intrasector compression (i.e., $n=1$).

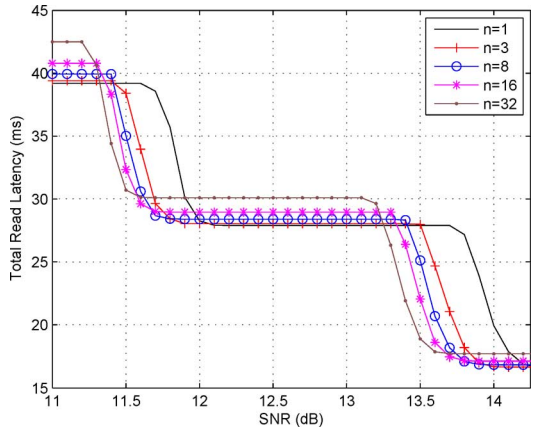

Fig. 8. Read latency under different various sector sizes for $\beta_4=94\%$.

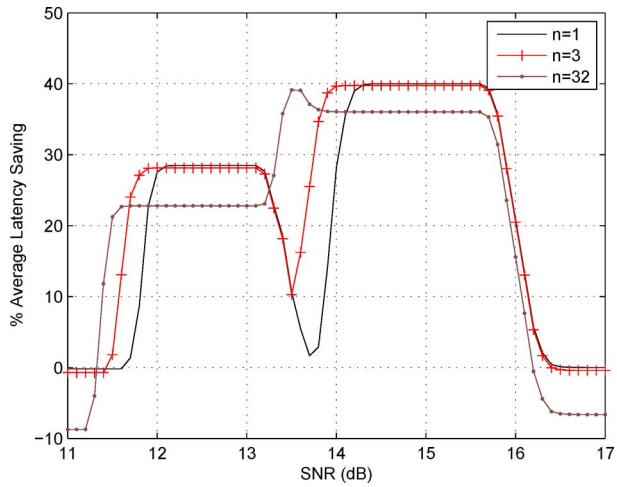

Fig. 9. Average read latency saving when using different virtual sector sizes.

B. Case Study II: Content-Aware Variable-Size Virtual Sector

As discussed above, when using the virtual sector design strategy, the read latency is subject to a tradeoff between compression efficiency and data transfer latency overhead. This inherent tradeoff depends on the virtual sector size n. Intuitively, different types of data (e.g., text files and executable files) can have (largely) different compressibility, hence they may prefer different values of n. This simple intuition inspired us to investigate the potential of content-aware variable-size virtual sector. First, we studied the difference of compression efficiency among different typical file types. In particular, using the same data set in [16] and [17], we studied five different file types, including dll, exe, html, txt, and xml files. Fig. 10 shows the compressibility versus compression ratio when using intrasector lossless compression. The results clearly reveal the significant difference among different types of files. We note that *all sectors* curve in Fig. 10 includes all the sectors we tested on the server including those already compressed data sectors. In addition, given the compressibility $\beta_2=98\%$, Fig. 11 shows

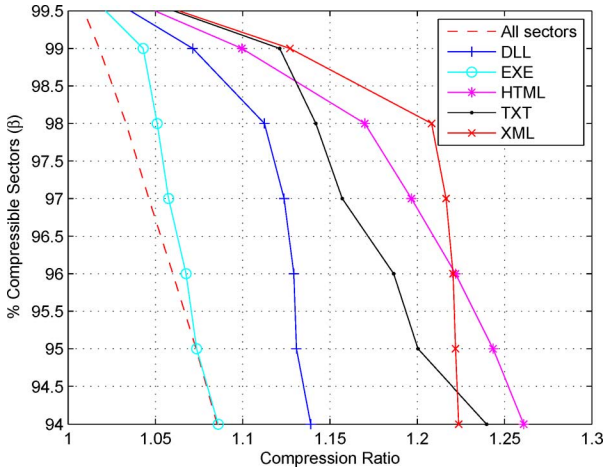

Fig. 10. Compressibility versus compression ratio for different types of file when using intrasector compression.

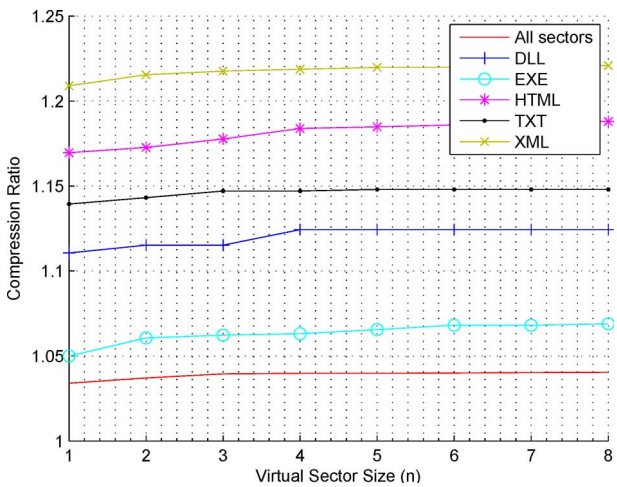

Fig. 11. Performance for different sector types with respect to different sector sizes for compressibility $\beta_2 = 98\%$.

TABLE III
VIRTUAL SECTOR SIZE BASED ON SECTOR TYPE

Sector type	DLL	EXE	HTML	TXT	XML
Virtual Sector Size (n)	4	2	4	3	2

the impact of virtual sector size on the achievable compression ratio for different types of files.

Based upon the results shown in Fig. 11, we can intuitively set the virtual sector size for different types of files, as listed in Table III. The reasoning for such content-aware virtual sector size configuration essentially corresponds to the minimum virtual sector size beyond which the compression ratio tends to saturate.

Assuming that different types of files contain the same amount of sectors and following the content-aware virtual sector size configuration listed in Table III, we can estimate the code rate reduction factors for four different compressibility, as listed in Table IV.

Using the above parameters and results, we estimate the average read latency under different SNR values, as shown in Fig. 12, when using such a content-aware virtual sector design strategy. For the purpose of comparison, we also show the average read latency when fixing the virtual sector size as $n = 1$ and $n = 3$, respectively. Furthermore, to illustrate the advantages of content-aware virtual sector over fixed-size virtual sector, Fig. 13 compares the read latency savings against the baseline scenario without using data compression. The results clearly demonstrate the advantage of incorporating content awareness in virtual sector design strategy.

The above results quantitatively demonstrate the potential effectiveness of the proposed design strategy. Nevertheless, we

TABLE IV
SIMULATED CODE RATE REDUCTION FACTORS FOR DIFFERENT COMPRESSIBILITY

Compressibility β_i	100%	98%	96%	94%
Code rate reduction factor η_i	1	0.8797	0.8583	0.8404

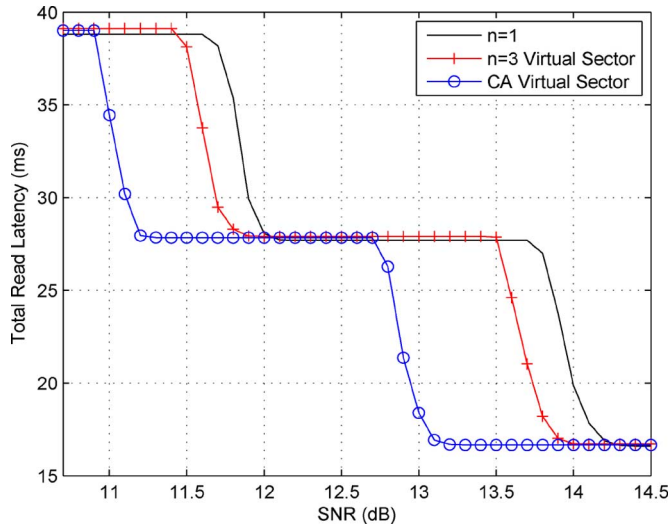

Fig. 12. Estimated read latency when using content-aware and fixed-size virtual sector design strategies.

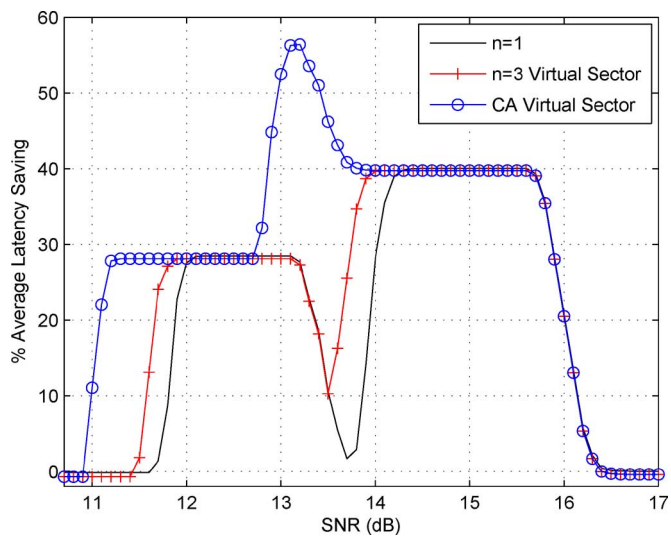

Fig. 13. Average read latency savings when using content-aware and fixed-size virtual sector design strategies.

would like to emphasize that the proposed design strategy is only applicable to compressible user data, hence should not be used for data that are already compressed such as multimedia data. In addition, the actual benefit apparently depends upon the specific design of the systems.

V. Conclusion

This paper demonstrates the potential of using lossless data compression to reduce the read latency penalty caused by ITI in the emerging shingled recording magnetic storage. Leveraging the fact that, except for the multimedia files, most files are typically stored on disks without compression, we propose to apply intrasector lossless compression to opportunistically enable the use of a stronger ECC that can obviate an explicit compensation for ITI. We carried out simulations and the results demonstrate that such a simple technique can reduce the disk read latency by up to 40% compared with conventional design practice without compression. In addition, we propose to extend intrasector lossless compression into intersector (or virtual sector) compression that applies compression across multiple consecutive sectors. We further discuss both fixed-size and content-aware variable-size virtual sector compression. Simulation results suggest that, compared with simple intrasector compression, virtual sector compression can reduce the read latency by up to 39%. These results demonstrate the promising potential of applying the proposed design strategy to address the read latency issue in future highly scaled shingled recording drives.

References

[1] K. Miura, E. Yamamoto, H. Aoi, and H. Muraoka, "Estimation of maximum track density in shingles writing," *IEEE Trans. Magn.*, vol. 45, no. 10, pp. 3722–3725, Oct. 2009.

[2] F. Lim, B. Wilson, and R. Wood, "Analysis of shingle-write readback using magnetic-force microscopy," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1548–1551, Jun. 2010.

[3] L. Barbosa, "Simultaneous detection of readback signals from magnetic recording tracks using array heads," *IEEE Trans. Magn.*, vol. 26, no. 5, pp. 2163–2165, Sep. 1990.

[4] P. Voois and J. Cioffi, "Multichannel signal processing for multiple-head digital magnetic recording," *IEEE Trans. Magn.*, vol. 30, no. 6, pp. 5100–5114, Nov. 1994.

[5] K. S. Venkataraman, G. Dong, and T. Zhang, "Techniques mitigating update-induced latency overhead in shingled magnetic recording," *IEEE Trans. Magn.*, vol. 48, no. 5, pp. 1899–1905, May 2012.

[6] A. Jagmohan, M. Franceschini, L. Lastras-Montano, and J. Karidis, "Adaptive endurance coding for NAND flash," in *Proc. IEEE GLOBECOM Workshops*, Dec. 2010, pp. 1841–1845.

[7] R. Wood, M. Williams, A. Kavčić, and J. Miles, "The feasibility of magnetic recording at 10 Terabits per square inch on conventional media," *IEEE Trans. Magn.*, vol. 45, no. 2, pp. 917–923, Feb. 2009.

[8] K. S. Chan, R. Radhakrishnan, K. Eason, M. R. Elidrissi, J. J. Miles, B. Vasic, and A. R. Krishnan, "Channel models and detectors for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 46, no. 3, pp. 804–811, Mar. 2010.

[9] A. Kavčić, X. Huang, B. Vasic, W. Ryan, and M. F. Erden, "Channel modeling and capacity bounds for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 46, no. 3, pp. 812–818, Mar. 2010.

[10] E. Hwang, R. Negi, and B. V. K. V. Kumar, "Signal processing for near 10 Tbit/in^2 density in two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 46, no. 6, pp. 1813–1816, Jun. 2010.

[11] M. Mallary, A. Torabi, and M. Benakli, "One terabit per square inch perpendicular recording conceptual design," *IEEE Trans. Magn.*, vol. 38, no. 4, pp. 1719–1724, Jul. 2002.

[12] S. Nabavi and B. V. K. V. Kumar, "Two-dimensional generalized partial response equalizer for bit-patterned media," in *Proc. IEEE Int. Conf. Commun.*, Glasgow, Scotland, Jun. 2007, pp. 6249–6252.

[13] W. Chang and J. R. Cruz, "Inter-track interference mitigation for bit-patterned magnetic recording," *IEEE Trans. Magn.*, vol. 46, no. 11, pp. 3899–3908, Nov. 2010.

[14] Z. Keirn, V. Krachkovsky, E. Haratsch, and H. Burger, "Use of redundant bits for magnetic recording: Single-parity codes and Reed-Solomon error correcting code," *IEEE Trans. Magn.*, vol. 40, no. 1, pp. 225–230, Jan. 2004.

[15] J. Storer and T. Szymanski, "Data compression via textual substitution," *J. ACM*, vol. 29, no. 4, pp. 928–951, Oct. 1982.

[16] W.-J. Li, K. Wang, S. Stolfo, and B. Herzog, "Fileprints: Identifying file types by n-gram analysis," in *Proc. 6th Annu. Inf. Assurance Workshop*, Jun. 2005, pp. 64–71.

[17] S. Moody and R. Erbacher, "Statistical analysis for data type identification," in *Proc. Systematic Approaches Digital Forensic Eng.*, Oakland, CA, USA, May 2008, pp. 41–70.