Coding-based Energy Minimization for Phase Change Memory

Azalia Mirhoseini
Electrical and Computer Engineering Department, Rice University
azalia@rice.edu

Miodrag Potkonjak
Computer Science Department, University of California, Los Angeles
miodrag@cs.ucla.edu

Farinaz Koushanfar
Electrical and Computer Engineering Department, Rice University
farinaz@rice.edu

ABSTRACT

We devise new coding methods to minimize Phase Change Memory write energy. Our method minimizes the energy required for memory rewrites by utilizing the differences between PCM read, set, and reset energies. We develop an integer linear programming method and employ dynamic programming to produce codes for uniformly distributed data. We also introduce data-aware coding schemes to efficiently address the energy minimization problem for stochastic data. Our evaluations show that the proposed methods result in up to 32% and 44% reduction in memory energy consumption for uniform and stochastic data respectively.

Categories and Subject Descriptors

B.7.1 [**Integrated Circuits**]: Memory Technologies

General Terms

Algorithms, Performance, Design

Keywords

Phase Change Memory, Energy Efficient Coding

1. INTRODUCTION

The demand for data and information storage has been upsurging at an unprecedented rate, continually fueling improvements for underlying memory technologies. However, it has become clear that alternative technologies are necessary to fulfill the requirements of developing devices and applications beyond the near future. In addition to these technology developments, redesigned architectures, tools, and system-level methodologies are needed to take advantage of the properties of the latest digital storage media.

One very promising emerging non-volatile storage technology is *Phase-Change Memory (PCM)*. PCM data storage exploits the large electrical resistance difference between two states of the phase-change material. In one state, the material is amorphous with a high resistance; in another, the material is crystalline and highly-conductive. After more than a decade of dedicated research into new forms of phase change media, PCM technology is finally available on the market. Recent announcements indicate advances towards multi-level phase change memory with improved integration, retention, endurance, and yield characteristics [19].

This paper aims at minimizing the energy cost of rewriting to PCM by creating low overhead data encoding methods. The proposed encoding scheme utilizes PCM bitwise manipulation ability during the word overwrites; only the bits that are changing for the new word compared to the existing word in the memory location would require overwriting. Our new encoding scheme ensures that the energy cost for the required overwrites is minimized at the expense of adding a small number of additional bits for encoding. Our formulation and solutions incorporate the fact that the PCM set and reset energy costs are not equal (see Appendix A). This asymmetric model captures the inherent physical differences between the organized crystalline and amorphous state transitions. To the best of our knowledge, this is the first work that utilizes PCM asymmetric set and reset energy behavior to minimize energy consumption. Our optimization is easily integrable within the processor architecture and memory interface with a very low complexity and overhead.

A special case of data encoding for minimizing the unidirectional transitions in the memory is the Rivest and Shamir Write-Once Memory (WOM) coding which assumed a memory model where the bits could only be set (and could not be reset) [17]. The goal was to increase the number of effective cycles for memory rewrites. Subsequent work followed, mostly in information theory and coding with the goal of estimating the capacity and finding more efficient WOM codes. Applications and extension of this model for addressing the flash memory device lifetime improvements were studied [8, 20]. Unfortunately, the assumptions made in the earlier theoretical work limits their applicability to PCM because they do not capture PCM bi-directionality and bit-level access properties.

The large space of possibilities provided by the freedom in both setting and resetting transitions and bit-programmability motivate the development of new type of codes that can be applied for improving PCM write energy. The complicating factors are the new degrees of freedom and the curse-of-dimensionality resulting from the exponential number of plausible code combinations. To address the challenges, this paper presents a novel formal handling of the

DAC 2012, June 3-7, 2012, San Francisco, California, USA.

energy minimization that is appropriate for PCM and other storage technologies with bit-level access while simultaneously considering the asymmetric set and reset transition energy costs. The paper's contributions are:

- We introduce a formal treatment and formulation of PCM coding, with the goal of minimizing the energy. We show that the problem is NP-complete.
- A methodology for deriving the optimal bounds for minimum energy data encoding problem is developed.
- We devise an Integer Linear Programming (ILP) formulation that can find the optimal codes. Our ILP framework can integrate both symmetric and asymmetric set/reset costs for different code sizes.
- For runtime and efficiency reasons, we develop an alternative rapid and efficient algorithm for addressing the problem. The method builds upon the smaller optimal codes using a Dynamic Programming (DP) approach.
- We introduce an efficient distribution-aware data encoding method for non-uniformly distributed data.
- Our evaluations on a diverse set of benchmark data show significant gains in PCM energy performance.

The remainder of the manuscript is organized as follows. The relevant literature is surveyed in Section 2. The architecture of the method is presented in Section 3. Section 4 formally defines the energy saving data encoding problem and discusses its complexity. Our method for finding the optimum bounds on the codes is presented in the same section. In Section 5 we introduce the coding algorithms. Evaluations of the methods on several benchmark data sets are presented in Section 6. We conclude in Section 7. Acknowledgements are presented in Section 7.1. We provide complementary methods and discussions in the Appendix.

2. RELATED WORK

The field of resistive memory material has been rapidly growing in recent years, both in research and in terms of industrial prototypes, making PCM the most viable emerging technology for the next generation storage devices [13, 19]. Recent work has shown significant efficiency and improvements in memory structures by integrating the PCM within the storage hierarchy [10, 14, 21].

Previous PCM research has demonstrated that the PCM endurance, reliability, and energy consumption would greatly improve if redundant writes are avoided, i.e., by reading the existing contents of the bits and only programming those bits that must be changed [21]. *Flip-N-Write* is a protocol that adds an indicator bit to each word to determine if the word is inverted or not, [9]. PCM controller can write the data in an inverted form if it requires less number of bit changes. No optimality proof was provided.

Our paper formalizes, provides proofs and generalizes the Flip-N-Write method by devising codes of length $N + K$ for words of length N, where $K \geq 1$. Our approach, for the first time in the literature, considers the asymmetric set and reset energy costs. We will show that significant improvements in energy are achieved at the expense of memory overhead.

Write-Once Memory (WOM) encoding was introduced in [17] to increase the number of writes to uni-directional memories. The NAND flash memory has been modeled as a one-way transitional memory and generalizations of the WOM codes have been applied to it [8, 20]. However, the WOM model and the flash encoding methods do not capture PCM properties including bit-level access and asymmetric energy costs. The bit-level operations for PCM have been used earlier for error correcting codes [18].

The WOM model has also been naturally extended to the family of Write-Efficient Memory [7], with the objective of minimizing the overall number of transitions. However, to the best of our knowledge, the few papers available on WEM have mainly focused on developing loose bounds without providing an optimality guarantee, or they centered on constructing suitable error correcting codes, e.g., [12, 15].

3. ENCODING ARCHITECTURE

Figure 1 presents an abstract view of the placement of the data encoding/decoding module for our method. Our algorithms for devising the codes are run off-line, so their runtime complexity does not affect the realtime chip performance. The energy saving codes resulting from our algorithms are then saved in the memory controller which interfaces to the PCM on one side and to processing units on the other side. The memory controller may also be interfaced to other storage devices in the memory hierarchy. The complexity of runtime encoding and decoding will be discussed in Section 5.3. Our consistent assumption is that the read energy consumption is negligible compared to that of set and reset [21]. More details of the PCM operation and energy characteristics can be found in Appendix A.

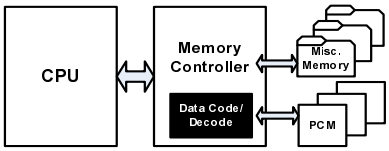

Figure 1: Data encoding/decoding module is a part of memory controller and is interfaced to the PCM.

4. PROBLEM FORMULATION, COMPLEXITY AND BOUNDS

Our goal is to minimize the energy cost associated with writing words to the memory. Each word consists of a fixed number of bits and the energy cost of writing the word is equal to the total cost of the required bit flips (sets/resets).

We provide an optimal encoding scheme that assigns multiple representations (or codes) to each word in the data set. The objective of encoding is to minimize the energy cost for writing the next word of data. The method trades-off the encoding data overhead with resulting energy improvements.

We have shown that assigning the best codes to each word is equivalent to clustering the vertices of a graph where each cluster represents a word (See Appendix B for an example). Clustering should be done such that it yields the minimum distance between the vertices of different clusters. We can formally define our problem as follows:

Problem. Minimize the energy cost of PCM rewrites.

Given. The word and the codeword (symbol) lengths in bits denoted by N and $N+K$ respectively, where $K \geq 1$. Each word is represented by 2^K symbols. The read, set and reset energies are denoted by E_{read} and E_S and E_R respectively.

Objective. Find the best codes for each word so as to minimize the average energy cost of overwrites. We refer to this problem as $\mathcal{P}(N, K)$.

4.1 Problem Formulation

We denote the words by $W_1, W_2, \ldots, W_{2^N}$ and denote the codes corresponding to word W_i by Z_{li}, where $1 \leq l \leq 2^K$.

The cost function C measures the amount of energy consumed to overwrite a symbol by another one. To overwrite Z_{li} with $Z_{l'i'}$, if N_S number of bit sets and N_R number of bit resets are needed, then C would be:

$$C(Z_{li}, Z_{l'i'}) = (N+K).E_{read} + (N_S).E_S + (N_R).E_R. \quad (1)$$

The first term on left shows the energy for reading Z_{li}. This cost is negligible due to PCM high read efficiency. The next two terms show the energy for the overwrite process (setting and resetting) so as to get $Z_{l'i'}$. Similar bits in the two symbols remain untouched. Function Φ gives the energy required to overwrite a currently written symbol Z_{li} by a symbol of the next word $W_{l'}$ that incurs the minimum cost:

$$\phi(Z_{li}, W_{l'}) = \min\{C(Z_{li}, Z_{l'i'}), \quad \forall 1 \leq i' \leq 2^K\}. \quad (2)$$

The Objective Function (OF) can be written as follows:

$$\textbf{OF}: \min\{\mathcal{C}(N,K) = \frac{1}{2^{2N+K}} \sum_{1 \leq l, l' \leq 2^N} \sum_{1 \leq i \leq 2^K} \phi(Z_{li}, W_{l'})\}. \quad (3)$$

The minimization is over all possible partitioning of the symbols to the words. Function $\mathcal{C}(N, K)$ represents the average energy cost of code overwrites for all possible rewrites.

4.2 Problem Complexity

We construct a transformation of the energy minimizing coding problem to a distance-based graph clustering problem where each cluster corresponds to a word (Appendix B). The goal is to minimize the inter-cluster distances. In our problem, the inter-cluster distance is the mean distance between the code symbols in one cluster and the closest code symbol in every other cluster. Extensive prior work on the class of distance-based graph clustering have shown that this problem is NP-complete. The proof was given by a reduction from the set covering problem [11].

4.3 Optimal Bounds on the OF

We develop a method for finding a lower bound for the OF. The average cost of overwriting each symbol Z_{li} with the other words is determined by the following formulation: $\frac{1}{2^N-1} \sum_{l'} \phi(Z_{li}, W_{l'})$ for $l' \neq l$ and $1 \leq l' \leq 2^N$. An optimal code assignment is the one that assigns each of the closest $2^N - 1$ symbols to Z_{li} to one of the words $W_{l'} \neq W_l$.

We construct a lower bound for the OF as follows. First, we calculate the distances from each code Z_{li} to all the other $2^{N+K} - 1$ possible codes. Next, the resulting distances are sorted and the average sum of the smallest $2^N - 1$ distances are calculated for each node. The computational complexity of this method is $O(2^{N+K}N + K)$. Based on the fact that the practical values for the memory word and code lengths are chosen relatively small (as we discuss in the evaluation results) and that the procedure is performed off-line, this method is applicable and gives a lower bound for the OF.

5. ENERGY MINIMIZATION ENCODING

We propose two different approaches for solving the coding problem. Our first solution is based on mapping the problem to an instance of an Integer Linear Programming (ILP). An ILP formulation requires linear objective function and constraints. The variables take integer values. There is a combinatorial complexity associated with assigning values to the variables of our NP-complete problem.

The OF represented in Equation 3 is non-linear since the function ϕ is a distance minimization function. To formulate the OF in a linear form, we define new indicator variables for the distance of each symbol in a cluster to the closest symbol in every other cluster. The OF is equivalent to the average of all these variables. Certain linear constrains are applied to ensure the variables meet the minimum distance criteria. The ILP method finds the optimal solution at the expense of runtimes exponentially increasing with the code size. Due to space limitation, details of the ILP formulation is discussed in Appendix C.

The second solution is based on Dynamic Programming (DP) paradigm for uniformly distributed data. We also develop codings for other data distributions that can further minimize the energy.

5.1 Coding for Uniform Data Distributions

First, we show the optimal coding for solving $\mathcal{P}(N, 1)$. Next, we show how to devise the codes for any $\mathcal{P}(N, K)$ based on the coding solutions for smaller N and K values.

5.1.1 *Optimal Coding for* $\mathcal{P}(N, 1)$

Claim: Optimal coding of $\mathcal{P}(N, 1)$, for any $N \geq 1$ is achieved by assigning the complement pairs of symbols to the words. The complement of a symbol is derived by flipping all its bits.

Proof: The optimal coding finds $2^K = 2$ symbols, each of size $N+1$, for each word. For now, we assume that set and reset cost equally. Then the overwrite cost is proportional to the number of bitwise differences for the codes, $E_R = E_S = E$. The average transition cost from each code Z_{li} to all the other words satisfies the following inequality:

$\frac{1}{2^N-1} \sum_{l'} \phi(Z_{li}, W_{l'}) \leq 0.\binom{N+1}{0} + E.\binom{N+1}{1} + \cdots +$
$\frac{N-1}{2} E.\binom{N+1}{[\frac{N-1}{2}]} + i_o.\frac{N+1}{2} E.\binom{N+1}{[\frac{N+1}{2}]}$, *for* $1 \leq l' \leq 2^N$.

Where $i_o = 1$ if N is odd and $i_o = 0$ otherwise. The right side of the inequality equals $E.(N+1)2^{N-1}$. The proof of the inequality is as follows. The nearest 2^N codes to Z_{li} should contain all the codes that have zero distance from it (that is Z_{li} itself); the number of such codes is $\binom{N+1}{0}$. It should also include all the codes that are just one bit different from Z_{li}; the number of such codes is $\binom{N+1}{1}$. The next closest set of codes are the ones that are different from Z_{li} in 2 bits and so on. We continue until we reach to the first closest 2^N codes to Z_{li}. In that case, the number of bit differences reach to $\frac{N}{2}$ when N is even and $\frac{N+1}{2}$ when N is odd. This is because the following equation holds:

$\binom{N+1}{0} + \binom{N+1}{1} + \ldots + \binom{N+1}{[\frac{N-1}{2}]} + i_o\binom{N+1}{[\frac{N+1}{2}]} = 2^N$, where i_o is the same as defined before.

Now, we show that the complement-pair coding assigns all the above 2^N codes to different words. In this case, the average transition cost for each code Z_{li} will be equal to its

Algorithm 1. DP-based method for energy-aware coding

Inputs: Word and code lengths: N, N+K; $\mathcal{C}(N,1)$ and optimal coding for $\mathcal{P}(N,1)$ from Section 5.1.1.

⋆ **Finding $\mathcal{C}(n,k)$ and the partitioning index $index(n,k,1:2)$:**

```
1   for (n=1 to n=N)
2     for (k=1 to k=K)
3       if (k==1)
4         C(n,k) = C(n,1);
5       else
6         for (i=1 to i=n-1)
7           for (j=1 to j=k-1)
8             if (C(n,k) ≥ C(n-i,k-j))
9               C(n,k) = C(n-i,k-j)+C(ij);
10              index(N,K,1:2)=(i,j);
```

⋆ **Building the codes for $\mathcal{P}(N,K)$:**

```
11  for (n=1 to n=N)
12    for (k=1 to k=K)
13      if (k==1)
14        P(n,k) = P(n,1) from Section 5.1.1;
15      else
16        P(n,k) = all code combinations from
            P(n−index(n,k,1), k−index(n,k,2))
            and P(index(n,k,1), index(n,k,2))
```

optimal value and thus the optimal OF is achieved. The sum of bitwise differences of Z_{li} from any complement pair $(Z_{l'1}, Z_{l'2})$, is equal to $N+1$. This is because each bit of Z_{li} is equal to exactly one of the bits of the complement pair. Thus, one symbol of each word has a distance of less than $\frac{N+1}{2}$ bits and the other symbol has a distance of more than $\frac{N+1}{2}$ bits from Z_{li}. This means that all the $2^N - 1$ closest codes to Z_{li} belong to different words. ■

Note that our complement results for the $K = 1$ case also apply to the asymmetric set/reset costs. The number of sets and resets for traversing from a code to its complement is not symmetric for most of the code words. Recall that our objective is to minimize the average costs over all possible transitions. It can be readily shown that for achieving the mean cost, the average inter-complement distance can replace the two disparate transition costs between the complements. The results of the claim then directly follows.

5.1.2 DP-based approach to $\mathcal{P}(N,K)$

We introduce a DP-based algorithm for solving the general $\mathcal{P}(N,K)$ problem. Our algorithm uses the coding results for $\mathcal{P}(p,q)$ and $\mathcal{P}(r,s)$ to construct the codes for $P(p+r,q+s)$ such that the following bounds can be achieved:

$$\mathcal{C}(p+r,q+s) = \mathcal{C}(p,q) + \mathcal{C}(r,s). \quad (4)$$

The code construction is as follows. The word W_i of length $p+r$ is partitioned into 2 words, W_i^1 and W_i^2. The first word is the first p bits and the second word is the last r bits of W_i. There are 2^q, $p+q$-bit symbols for W_i^1 and 2^s, $r+s$-bit symbols for W_i^2 that are obtained from solving $\mathcal{P}(p,q)$ and $\mathcal{P}(r,s)$ respectively. We construct the codes for W_i by concatenating all the possible combinations of these two set of symbols which provides a total of $2^q \times 2^s = 2^{q+s}$ codes (of length $p+q+r+s$) for W_i. It can be easily seen that the codes satisfy Equation 4. Based on the above code construction, the DP method breaks N into smaller values and selects the best partitioning to minimize:

$$\mathcal{C}(N,K) = \min_{i \le N} \{\min_{j \le i} \mathcal{C}(N-i, K-j) + \mathcal{C}(i,j)\}. \quad (5)$$

Algorithm 1 provides the details of the DP method. The optimal coding for $\mathcal{P}(N,1)$ is given from the previous part and the algorithm iteratively traverses over all the possible partitions to improve the energy minimization objective (Lines 1-10). The index vector $index(n,k,1:2)$ is used to store the optimal partitioning of (n,k). After finding all the indices, the algorithm builds the codes (Lines 11-16). The complexity of the algorithm is $O(N^2K^2)$, but recall that this algorithm is run off-line.

5.2 Coding for Stochastic Data

OF 3 minimizes the average energy cost for all the possible word overwrites. Here, we discuss how the inherent stochastic properties for real data scenarios can be exploited for further energy improvements. An important feature is that different words have differing frequencies. To benefit from this fact, instead of weighting all the rewrite energy costs equally, we aggressively optimize our encoding for the rewrites that are more prevalent by assigning different number of codes to the words based on their frequency.

Variable-length and fixed-length coding are two statistical compression techniques. In the variable-length method, shorter codes are assigned to the more frequent words to better improve the compression. However, this adds to decoding complexity and since our main goal is to minimize the energy, decoding efficiency is very important. Thus, we use a fixed-length coding method. We describe our method on text files. The method can be generalized to other data sets with nonuniform frequencies. Our data consists of the lower-case alphabet letters: $W_1 = a$, $W_2 = b$, ..., $W_{26} = z$. Since there are 26 letter, W_i's are 5-bit words.

Let us consider the first 7 most frequent letters of the table, e, t, a, o, i, n and s. The probability that an overwrite occurs on any of these letters (by any other letter) plus the probability that these letters overwrite any other letter accounts for almost 60% of all probable overwrites. Thus, we can benefit a lot by optimizing our coding for these seven letters. To do so, we assign a different prefix to each of these letters such that only the prefixes determine the letter. Since there are 7 letters, the prefixes are 3-bit each and are shown in Figure 2. The prefixes can be interpreted as dictionary indices. The remaining $N+K$ bits of these letters take all the possible 2^{N+K} states. Thus, an overwrite to/by any of these letters requires only adjusting the prefix that is of length 3. The other 19 letters have the prefix (111) as shown in the figure. The remaining $N+K$ bits for the less frequent letters are filled with the codes obtained by solving $\mathcal{P}(N,K)$ as described in Subsection 5. Thus, an overwrite between the letters costs as much as for a regular $\mathcal{P}(N,K)$. By this coding, we assign 2^{N+K} symbols to the highly frequent letters and 2^K codes to the rest of the letters. All the symbols are of length *length of prefix-length*$+N+K$.

5.3 Runtime Coding/Decoding Complexity

As mentioned in Section 3, our algorithms for developing code words are run off-line. The results of our algorithms are then stored in the memory controller as a look-up table. When writing a new word to the memory, there are 2^K options for the word, where K is a small constant number. In our evaluations we used K in range 1-4. The coding

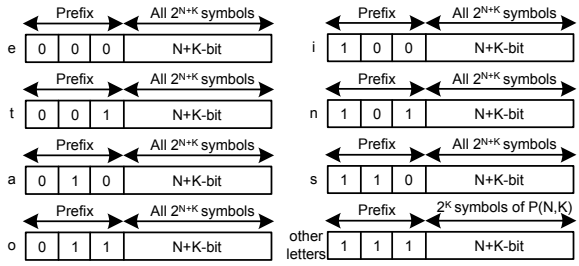

Figure 2: Data-aware alphabet letter codings.

complexity is in the order $\Omega(2^K)$. To do each decoding, the code words can be placed in a binary tree with a depth $K+N$. Searching for a symbol on this tree has an $\Omega(K+N)$ complexity. Thus, both our coding and decoding operations have a very low overhead.

6. EVALUATION RESULTS

We evaluate our energy-aware encoding methods on a variety of benchmark data sets. We perform our evaluations for different relative set and reset energy ratios $\frac{E_R}{E_S}$ and discuss their impact on the energy efficiency. To have a fair comparison, we normalize the costs such that $E_R + E_S = 1$.

6.1 DP-based Algorithm

We analyze DP-based encoding method provided in Algorithm 1 for different word lengths. We compare the average energy costs obtained from this method to that of the no-coding (nc) method and the optimal bound (opt) from Section 4.3. The no-coding method is equivalent to the problem $\mathcal{P}(N,0)$. We denote the average overwrite energy costs for the words for the above three methods as follows; $\mathcal{C}_{nc}(N,0)$, $\mathcal{C}_{dp}(N,K)$ and $\mathcal{C}_{opt}(N,K)$.

Table 6.1, shows the results for the case where $\frac{E_R}{E_S} = 2$ for different word lengths N and number of extra-bits K. Columns six shows the average improvement in the cost obtained from the DP compared to no-coding method. The result shows notable savings. For example, for $N = 8$ and $K = 2$, energy cost is reduced by 72%. Thus, for each word overwrite, we save on average 28% of the energy at the expense of adding 2 bits. The last column shows the DP performance compared to the optimal achievable bounds. Our results show that DP algorithm achieves values very close (in some cases equal) to the optimal bound.

Table 1: DP cost ($\mathcal{C}_{dp}$) comparison against no-coding cost ($\mathcal{C}_{nc}$) and optimal cost ($\mathcal{C}_{opt}$).

N	$\mathcal{C}_{nc}(N,0)$	K	$\mathcal{C}_{opt}$	$\mathcal{C}_{dp}$	$\frac{\mathcal{C}_{dp}(N,K)}{\mathcal{C}_{nc}(N,0)}$	$\frac{\mathcal{C}_{opt}}{\mathcal{C}_{dp}}$
2	.5	1	.37	.37	.75	1
3	.75	1	.55	.55	.73	1
4	1	1	.75	.75	.75	1
4	1	2	.68	.72	.72	.94
8	2	1	1.48	1.48	.74	1
8	2	2	1.42	1.44	.72	.98
8	2	3	1.34	1.39	.69	.96
8	2	4	1.30	1.36	.68	.95

Figure 3 shows the average energy cost $\mathcal{C}(N,K)$ for different $\frac{E_R}{E_S}$ values; N is set to 10 and K is in the range 1,2,...,5. We see that as the ratio $\frac{E_R}{E_S}$ increases, better energy savings are achieved. This is because our coding scheme aims to optimize the energy consumption by minimizing the number of overwrites. Since resets have a higher energy cost, the minimization impact will be higher for them.

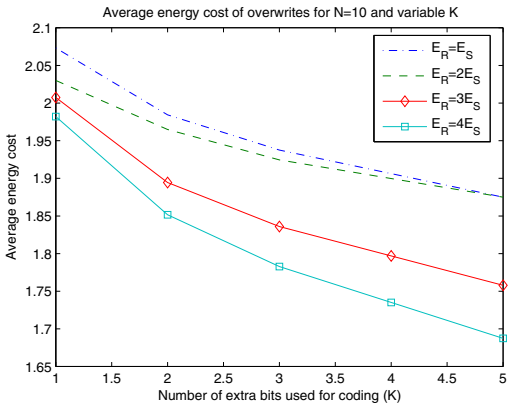

Figure 3: Cost reduction by data-aware coding.

6.2 Performance on Audio and Image Data

We use the encoding method to store audio and image data on PCM. Our benchmark data are from Columbia University audio and Caltech Vision image databases, [1, 6]. The audio data are $msmn1.wav$, $msmv1.wav$, $mssp1.wav$, and $msms1.wav$. The image files are $dcp-2897.jpg$, $dcp-2898.jpg$, $dcp-2899.jpg$ and $dcp-2830.jpg$. Figure 4 shows the average energy reductions for all file overwrites. More details are outlined in Appendix D.

For audio data, $\mathcal{P}(4,1)$ and $\mathcal{P}(4,2)$ encodings are applied and the results show an energy reduction of 11% and 21% respectively. For image data, $\mathcal{P}(8,1)$ and $\mathcal{P}(8,2)$ encodings are applied and the results demonstrate an average energy reduction by 18% and 28% respectively. The savings are significant and confirm the notable energy improvements of encoding at the expense of adding a few extra bits.

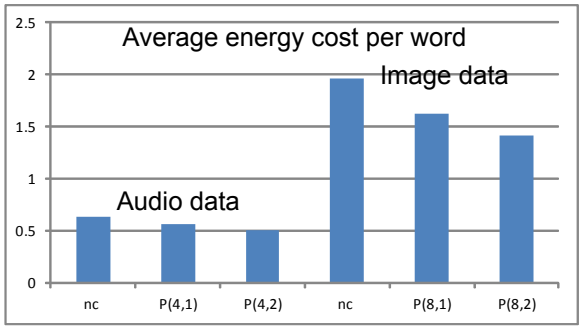

Figure 4: Average energy cost per word, $\frac{E_R}{E_S} = 2$.

6.3 Distribution-Aware Data Coding

We first evaluate English alphabet coding as described in Section 5.2. Then, we provide coding and evaluations for

the ASCII characters. We used two text benchmarks, the 31 MB *text8.txt* file from [2], for alphabet evaluations; and the 4.8 MB *KJV.txt* file from [4] for ASCII evaluations.

6.3.1 Alphabet Letters

We encoded the alphabet letters with the distribution-aware encoding. Since there are 26 alphabet letters, $N = 5$; we set $K = 1$, and *Prefix*=3. The codes are of length *Prefix-length*$+N+K = 9$. We evaluated the method on $Text8.txt$ data for different test trials. For each trial, we created 100 pairs of vectors by randomly reading the data from the text file. Each vector has 1000 letters. We overwrote the vectors of each pair and computed the average overwrite cost for $\frac{E_R}{E_S} = 2$. The results demonstrate an average 44.1% reduction when compared to the no-coding scheme and 9.3% reduction compared to the uniform coding $\mathcal{P}(5, 2)$.

6.3.2 ASCII Characters

According to the frequencies of ASCII characters from [3], 59% of all the possible rewrites are to/by one of the first 15 most frequent characters. Thus, we optimize our coding for these characters by assigning separate prefixes to them.

The first 15 most frequent characters are: space, e, t, a, o, i, n, s, h, r, d, l, u, m, c. We assigned the following 4-it prefixes to them respectively: (0000), (0001), (0010), (0100), (1000), (1001), (1010), (0110), (0111), (1011), (1101). The prefix for all the other characters is (1111). Since there are 2^7 ASCII characters, $N = 7$ and we set $K = 1$. Thus, the codes will be of length $4 + N + K = 12$. The encoding method is the same as described for alphabet letters.

We evaluated the ASCII coding scheme on the *KJV.txt* file. We created 100 pairs of vectors, each of length 1000 from the file. The first vector in each pair was overwritten by the second vector. We considered $\frac{E_R}{E_S} = 2$. To compare this method with the uniform coding, we encoded the ASCII characters with the codes from $\mathcal{P}(7, 1)$, $\mathcal{P}(7, 2)$ and $\mathcal{P}(7, 3)$ and report the corresponding average costs in the following:

Encoding	Data-aware	$\mathcal{P}(7,1)$	$\mathcal{P}(7,2)$	$\mathcal{P}(7,3)$
Avg cost	1.24	1.42	1.37	1.34

We see that the ASCII data-aware coding, on average, reduces the energy cost to 92% of the best cost achieved from $\mathcal{P}(7, 3)$. Thus, for overwriting each ASCII character, there will be an 8% reduction in the energy cost compared to the results of the uniform encoding. This improvement is at the expense of two extra bits per character.

7. CONCLUSION

We proposed a novel data coding methodology for minimizing PCM write energy. Our approach creates several alternative symbols for each word being written in the memory, trading off energy efficiency with encoding overhead. The new word that is going to be written on the memory is encoded by the symbol with minimum distance to the existing word on that memory location. To address the problem, we developed (i) an ILP-based solution that mostly incurs a high combinational complexity; and (ii) a Dynamic Programming-based approach that combined the smaller optimal codewords. For cases where the distributions of the letters in the alphabet were a priori known, we created a new data-aware algorithm that incorporated those information for further energy reductions. Evaluations on a diverse set of text, image, and audio benchmark data demonstrated the applicability and effectiveness of our new methods.

7.1 Acknowledgments

This research is in part supported by ONR YIP award under grant No. R16480, ARO YIP award under grant No. R17450 and NSF CCF-0926127 award.

8. REFERENCES

[1] http://labrosa.ee.columbia.edu/sounds/.
[2] http://mattmahoney.net/dc/textdata/.
[3] http://millikeys.sourceforge.net/freqanalysis.html.
[4] http://patriot.net/ bmcgin/kjvpage.html.
[5] http://www.gurobi.com/.
[6] http://www.vision.caltech.edu/html-files/archive.
[7] R. Ahlswede and Z. Zhang. Coding for write-efficient memory. *Info and Comp.*, 83(1):80–97, 1989.
[8] J. Anxiao, M. Langberg, M. Schwartz, and J. Bruck. Universal rewriting in constrained memories. In *ISIT*, pages 1219–1223, 2009.
[9] S. Cho and H. Lee. Flip-N-Write: a simple deterministic technique to improve PRAM write performance, energy and endurance. In *MICRO*, pages 347–357, 2009.
[10] G. Dhiman, R. Ayoub, and T. Rosing. PDRAM: A hybrid *PRAM* and *DRAM* main memory system. In *DAC*, pages 664 –669, 2009.
[11] T. F. and Gonzalez. Clustering to minimize the maximum intercluster distance. *Theoretical Computer Science*, 38(0):293–306, 1985.
[12] F. Fu and R. Yeung. On the capacity and error-correcting codes of write-efficient memories. *IEEE Tran. on IT*, 46(7):2299 –2314, 2000.
[13] S. Lai. Current status of the phase change memory and its future. In *IEDM*, pages 10.1.1 – 10.1.4, 2003.
[14] B. Lee, E. Ipek, O. Mutlu, and D. Burger. Architecting phase change memory as a scalable dram alternative. In *ISCA*, pages 2–13, 2009.
[15] T. Mittelholzer, L. Lastras-Montañ Ando, M. Sharma, and M. Franceschini. Rewritable storage channels with limited number of rewrite iterations. In *ISIT*, pages 973 –977, 2010.
[16] M. Qureshi, J. Karidis, M. Franceschini, V. Srinivasan, L. Lastras, and B. Abali. Enhancing lifetime and security of PCM-based main memory with start-gap wear leveling. In *MICRO*, pages 14–23, 2009.
[17] R. Rivest and A. Shamir. How to reuse a write-once memory. In *STOC*, pages 105–113, 1982.
[18] S. Schechter, G. Loh, K. Straus, and D. Burger. Use ECP, not ECC, for hard failures in resistive memories. In *ISCA*, pages 141–152, 2010.
[19] H. Wong, S. Raoux, S. Kim, J. Liang, J. Reifenberg, B. Rajendran, M. Asheghi, and K. Goodson. Phase change memory. *Proceedings of the IEEE*, 98(12):2201–2227, 2010.
[20] Y. Wu and A. Jiang. Position modulation code for rewriting write-once memories. *IEEE Tran. IT*, 57(6):3692–3697, 2011.
[21] P. Zhou, B. Zhao, J. Yang, and Y. Zhang. A durable and energy efficient main memory using phase change memory technology. In *ISCA*, pages 14–23, 2009.

APPENDIX

A. PCM OPERATION AND ENERGY MODEL

A key challenge for non-volatile memory technology, in particular flash, is the high energy cost of writes [19]. The speed of writing and reading from the caches and from the DRAM is often high, and therefore, the number of transitions is higher than the external memories. Therefore, since resistive memory is suggested for replacing and complementing various storage units in the memory hierarchy, saving the energy cost of set and reset transitions is of a high value [19, 16].

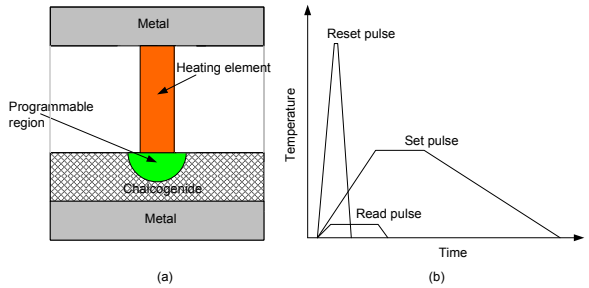

Figure 5: (a) A PCM memory cell; (b) Current pulses for set, reset, and read operations.

As shown in Figure 5(a), the phase change media is placed between an electrode layer and a layer composed of a heater. The current flows through the phase change material from the electrode to the heater. This current is provided as a pulse, and its duration and amplitude controls the temperature needed for the set and reset operations. Heating the phase change material above a crystallization temperature by applying an average current but wide duration pulse results in the set operation. A very high current (melt quenching) pulse with a short duration resets the device to its amorphous state. The read is done by applying a very low amplitude and low power pulse that senses the device resistance. The shape of the three pulses used for set, reset, and read commands is plotted in Figure 5(b), [19]. The energy discrepancy between the PCM set and reset operations has been experimentally shown.

B. A WORD ENCODING/DECODING EXAMPLE

In this example, we describe how one may benefit from coding the PCM data. Here we are solving the problem of finding the optimal coding for 2-bit words with 3-bit codes. We denote the words by W_1=(00), W_2=(01), W_3=(10), W_4=(11) and denote the codes corresponding to the word W_i by Z_{i1} and Z_{i2}, for $1 \leq i \leq 4$; since K=1 each word has $2^{K=1}$=2 code representation. The key point is to exploit multiple representations of each word for minimizing the write energy. For instance, if the existing data is Z_{11} and W_2 is to be written on it, among its representations Z_{21} and Z_{22}, the one that incurs the minimum energy cost to overwrite Z_{11} (which in this case is Z_{21}) is selected.

Figure 6 shows a graph representation of the encodings for the 2-bit words shown in separate clusters. The vertices of the graph are the codes and each cluster represents a word.

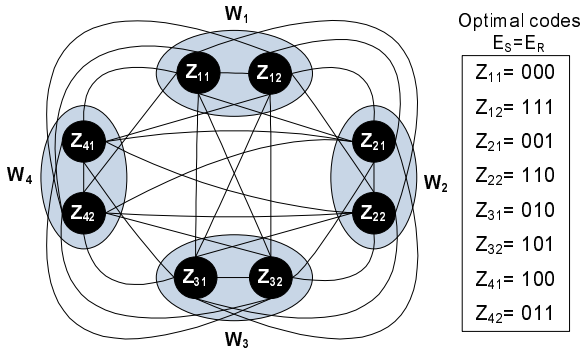

Figure 6: A 3-bit encoding for the 4 words W_1, W_2, W_3, and W_4.

The graph is a directed graph and the weight of each edge shows the cost of overwriting one node with the other. The optimal encoding is derived and provided on the figure. If the code Z_{22} is to be overwritten by a code of W_3, Z_{31} is selected because its energy cost is only equal to E_S. If no coding was used, overwriting W_2 with W_3 would cost the higher value of E_R+E_S. Another example is a cycle of word overwrites (W_1,W_2,W_3,W_4,W_1). Assume that W_1 is coded as Z_{11}. Then, the minimum cost codes would be selected as follows (Z_{11},Z_{21},Z_{32},Z_{41},Z_{11}). The cost associated with the code overwrites is $E_S + E_S + E_S + E_R + E_R = 2.E_S + 2.E_R$. Whereas the cost for overwriting the codes without coding is $E_S + (E_S + E_R) + E_S + (2E_R) = 3.E_S + 3.E_R$.

An example of a binary tree for decoding the data with 8 code words ($N + K$=3) for the codes developed in Figure 6 is shown in Figure 7.

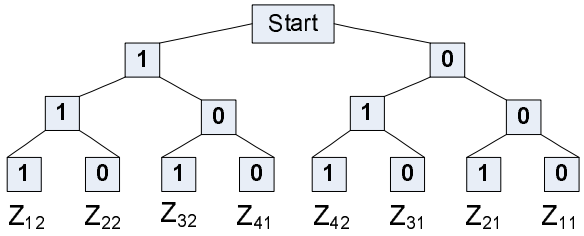

Figure 7: Binary tree for encoding.

C. INTEGER LINEAR PROGRAMMING FORMULATION

To formulate OF in a linear form, we define an index variable that for each symbol, keeps track of the index of the element (in each of the other clusters) with the minimum distance to the symbol. The following set of variables were used in our ILP formulation:

l, l'	Words indices W_l or W'_l for $1 \leq l, l' \leq 2^N$.
i, i'	Code indices within each cluster, $1 \leq i, i' \leq 2^K$.
Z_{li}	The i-*th* code $\in W_l$ for all i.
$\Phi_{ll'i}$	$\phi(Z_{l'i}, W_l)$ for all l, l', i and i'.
$w_{ll'ii'}$	$w(Z_{li}, Z_{l'i'})$ for all l, l', i and i'.
$\Delta_{ll'ii'}$	$w_{ll'ii'} - \Phi_{ll'i}$ for all l, l', i and i'.
X_{lij}	j-*th* significant bit of Z_{li} for $1 \leq j \leq (N+K)$.
$F_{ll'ii'j}$	$w(X_{lij}, X_{l'i'j})$ for all l, l', i, i' and j.
$Id_{ll'ii'}$	An indicator binary; =0 iff $\Delta_{ll'ii'} = 0$ for all l, l', i and i'.

The codes representing a word W_l are shown by Z_{li}; $\Phi_{ll'i}$ denotes the cost of overwriting Z_{li} by a code in $W_{l'}$ that requires the minimum overwrite energy; $w_{ll'ii'}$ is the cost of overwriting two codes U_{li} and $U_{l'i'}$. Thus, $\Phi_{ll'i} = \min_{i'} w_{ll'ii'}$. Each code Z_{li} consists of $N+K$ bits and can be written as $(X_{liN+K}, \ldots, X_{li2}, X_{li1})$. The parameter $F_{ll'ii'j}$ is defined to be the cost of overwriting X_{lij} with $X_{l'i'j}$ and its range of values is shown in the table below. Variable $Id_{ll'ii'}$ is an indicator binary variable that indicates if the closest code to Z_{il} in cluster l' is $Z_{i'l'}$ or not.

X_{lij}	$X_{l'i'j}$	$F_{ll'ii'j}$
0	0	0
0	1	E_S
1	0	E_R
1	1	0

Using the above variables, we define our OF and provide constraints to our problem in a way that conforms to the ILP format. Our OF, as written in Equation 3, minimizes the average cost of overwriting the codes for all possible overwrites:

$$OF: \min \frac{1}{2^N.2^N.2^K} \sum \Phi_{l'li} \qquad \text{for all } l', l \text{ and } i \text{ variables} \tag{6}$$

The following constraints define $\Phi_{ll'i}$:
C1. $\Delta_{ll'i} \geq 0$ for all l, l' and i variables,
C2. $\Sigma_{i' \in 1,\ldots,2^k} Id_{ll'ii'} \leq 2^K - 1$,
C3. $Id_{ll'ii'} \leq \Delta_{ll'ii'}$,
C4. $E_R.(N+K).Id_{ll'ii'} \geq \Delta_{ll'ii'}$.

Constraints C1 and C2 set $\Phi_{ll'i}$ not greater than each distance $\Delta_{ll'i}$ and equal to at least one of them respectively; Constraints C3 and C4 define the indicator variable based on the fact that $E_R.(N+K)$ is always grater than $\Delta_{ll'ii'}$.

The following linear constraints set $F_{ll'ii'j}$ as defined in the table above:
C5. $\frac{1}{E_R+E_S} F_{ll'ii'j} + X_{lij} + X_{l'i'j} \leq 2$,
C7. $F_{ll'ii'j} - E_R.X_{lij} - E_S.X_{l'i'j} \leq 0$,
C8. $F_{ll'ii'j} - E_R.X_{lij} - E_R.X_{l'i'j} \geq 0$,
C9. $F_{ll'ii'j} - E_S.X_{lij} - E_S.X_{l'i'j} \geq 0$.

The following constraint defines distance $w_{ll'ii'}$:
C10. $w_{ll'ii'} = \Sigma_{1 \leq j \leq N+K} F_{ll'ii'j}$.

The following constraint is set to ensure that no code is assigned to more than one word; E_S is the minimum cost of overwriting two different codes:
C11. $w_{ll'ii'} \geq E_S$.

The output of the above ILP is the values of X_{lij} that constructs the codes U_{il}. The above constraints are all in linear format and can be readily implemented by any ILP solver. The complexity and runtime for solving the instances of the ILP for our NP-complete problem exponentially increases with the instance size. In our experiments, we have been able to find the optimal solution by using a limited version of an ILP solver licensed to one user for N and K ($N = 2, 3, 4$, $K = 1, 2$). If one has access to the commercial ILP solvers that run on supercomputers, it is likely possible to find the optimal codes for the practical codes of longer sizes. The longer runtimes can be tolerated since the ILP needs to be used only once and offline (See Section 5.3).

C.1 ILP Results

We used the latest version of Gurobi ILP solver, Gurobi 4.5.2, to solve the ILP defined in Section C, [5]. Gurobi provides free access for academic purposes. The runtime of the solver for solving $\mathcal{P}(4,2)$ is about 30 hours on an Intel Core 2 Duo Processor T9600 computer. Thus, due to the time constraint we were not able to solve the OF for larger problems. However, the authors make the python ILP code available to the interested readers.

D. RESULTS

D.1 Audio and Image Data

We provide more details of the encoding evaluations on audio and image data as described in Section 6.2. The audio data were msmn1.wav, msmv1.wav, mssp1.wav, and msms1.wav and are denoted by $a1$, $a2$, $a3$ and $a4$ in Table D.1. The image files are $dcp-2897.jpg$, $dcp-2898.jpg$, and $dcp-2899.jpg$ and $dcp-2830.jpg$ and are presented by $i1$, $i2$, $i3$ and $i4$ in Table D.1.

Table 2: Average energy costs of DP and no-coding methods for audio data.

Energy costs	nc $\mathcal{P}(4,0)$	dp $\mathcal{P}(4,1)$	dp $\mathcal{P}(4,2)$
$a1 \rightarrow a2$	0.59	0.50	0.43
$a1 \rightarrow a3$	0.66	0.54	0.50
$a1 \rightarrow a4$	0.66	0.62	0.60
$a2 \rightarrow a3$	0.61	0.48	0.39
$a2 \rightarrow a4$	0.59	0.67	0.58
$a3 \rightarrow a4$	0.71	0.60	0.54

Table 3: Average energy costs of DP and no-coding methods for image data.

Energy costs	nc $\mathcal{P}(8,0)$	dp $\mathcal{P}(8,1)$	dp $\mathcal{P}(8,2)$
$i1 \rightarrow i2$	2.20	1.78	1.61
$i1 \rightarrow i3$	2.44	1.66	1.44
$i1 \rightarrow i4$	1.90	1.70	1.58
$i2 \rightarrow i3$	1.66	1.73	1.46
$i2 \rightarrow i4$	1.79	1.38	1.22
$i3 \rightarrow i4$	1.75	1.46	1.19

In both tables, the first column shows the files that are overwritten. For example $a1 \rightarrow a2$ means that $a2$ is overwritten by $a1$. The second and third column show the average overwrite costs for the DP-based algorithm (Data is

encoded) and the no-coding method for $\mathcal{P}(4,1)$ (audio data) and $\mathcal{P}(8,1)$ (image) data encodings. For example, the average cost of a word overwrite in $a1 \rightarrow a2$ is 0.50 in DP, while this value is 0.59 in the no-coding method while encodings from $\mathcal{P}(4,1)$ is applied. The forth and fifth columns show the same results for $\mathcal{P}(4,2)$ (audio data) and $\mathcal{P}(8,2)$ (image) data encodings. All results correspond to $\frac{E_R}{E_S} = 2$. Meaningful improvements are achieved by the energy-minimization coding method. The energy cost on average is reduced by our 15.6% and 22.5% for audio and image data respectively.