Reducing Data Movement Energy via Commutative Data Reordering

Ben Feinberg*, Benjamin Heyman*, Darya Mikhailenko*, Ryan Wong*, and Engin Ipek*†
*Department of Electrical and Computer Engineering †Department of Computer Science
University of Rochester
Rochester, NY 14627
Email Contact: ipek@cs.rochester.edu

***Abstract*—Data movement is a significant consumer of energy in modern computer systems. To reduce these costs, recent work has proposed architecting interconnects with asymmetric data transmission costs, and developing encoding techniques to exploit this asymmetry. Although promising, these encoding techniques do not take full advantage of application level characteristics. As an example of a missed optimization opportunity, consider the case of computing a dot product as part of a neural network inference task. The order in which the weights are fetched from memory does not affect correctness, and can be optimized to minimize data movement energy—an optimization that is not possible on today's systems.**

This paper examines commutative data reordering (CDR), a new technique that leverages the commutative property in linear algebra to strategically select the lowest energy order in which data can be transmitted. When applied to sparse matrix vector multiplication, CDR reduces data movement energy by 21% over existing encoding techniques, for a total reduction of 1.89x over a baseline interconnect. These energy savings are achieved with zero metadata or bandwidth overhead to support the reordering.

***Keywords*—computer architecture; data encoding**

I. Introduction

Data movement energy has become an increasingly important issue in modern system design. One recent analysis showed that the cost of moving data 10mm on chip consumed more than $10\times$ the energy of a double precision floating point operation [1]. Notably, the high cost of data movement applies across a wide range of devices and platforms, from ultra-low power internet of things processors to specialized machine learning accelerators and GPUs with power consumptions in the hundreds of watts. Thus, improvements in data movement energy have the potential to provide benefits to a diverse set of applications.

Due to this importance, a number of techniques to reduce data movement energy have been proposed, both on and off chip, and in a variety of architectures. These approaches architect the interconnects such that transmitting a **0** is significantly less expensive than transmitting a **1**. This asymmetry enables encoding techniques that save energy by reducing the number of **1**s in a transmission. Interconnects with asymmetric data transmission costs can be architected in two differet ways. Terminated interconnects, found in the GDDR5/6 standard, have a termination resistor that forms a direct path between V_{DD} and ground when transmitting a **1**. Unterminated interconnect, by contrast, consume energy primarily when the state of the interconnect changes from a **0** to a **1** or vice versa. In unterminated interconnects, transition signaling [2] can be used to create the same asymmetric data transmission cost, where **1**s and **0**s respectively are represented by the presence or absence of a voltage transition on the wire.

Given an interconnect with asymmetric data transmission costs, numerous encoding techniques proposed to reduce the number of **1**s in a transmission. Several of these techniques exploit well known properties of data similarity and value locality to encode data as the bitwise XOR between current and previous transmissions [3], [4]. Although these techniques show promise, they all share a similar limitation: the encodings do not consider application level characteristics beyond inherent data-similarity, blindly treating the data passing over the interconnect as a stream of bits. To take advantage of application level characteristics, we propose **Commutative data reordering (CDR)**.

II. Commutative Data Reordering

Data similarity based encoding methods can be improved by reordering the data stored in memory such that elements with high similarity are fetched close together. Reordering does not impact application correctness so long as the reordered data has the commutative property ($a + b = b + a$). Importantly, many parallel applications already rely on the commutative property, since parallel execution on many systems does not have strong ordering guarantees.

To understand the benefits of CDR, consider the summation of a vector containing the binary values $(0001, 1111, 1110, 1001)$ where each value is transmitted over a bus where **1**s are more expensive than **0**s. Without any encoding, the total cost of transmitting the values is proportional to the sum of the hamming weights of each value, 10 in this example. Alternatively, if the values are transmitted as bitwise differences between successive vector elements, the vector is transmitted as $(0001, 1111 \oplus 0001, 1111 \oplus 1110, 1110 \oplus 1001)$ where $\oplus$ is the bitwise xor between two values. With this bitwise difference encoding, the total hamming weight of the transmission is 8, a 20% reduction. Since addition is commutative, we can apply CDR to this vector, changing the

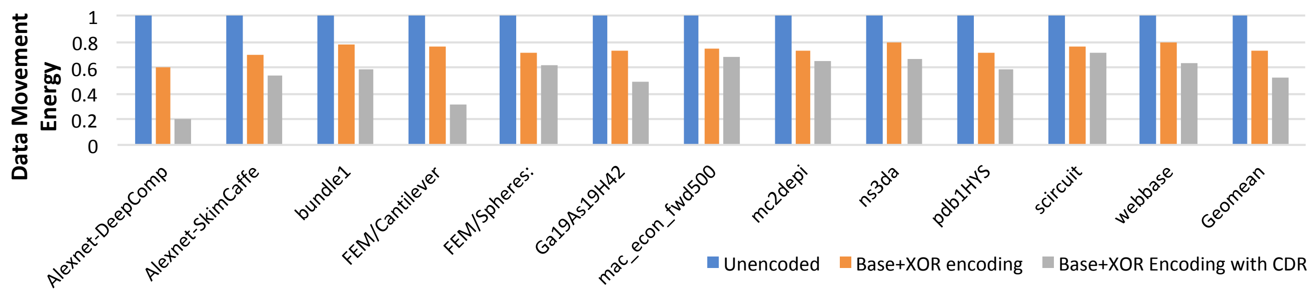

Fig. 1. Reduction in transmitted **1**s normalized to interconnect without encoding.

order to $(0001, 1001, 1111, 1110)$ and transmit that vector with bitwise difference encoding, for a total hamming weight of 5, an additional 30% reduction over the unreordered data.

Although the example above uses summation, CDR can be applied to other operations such as min, max, and dot products. For specificity, we consider matrix-vector multiplication (MVM), an essential computational kernel in a variety of applications. MVM is also a very data movement intensive kernel as the matrix, the part of the computation with the largest memory footprint, has no reuse, requiring every element to be fetched from off-chip memory at a high cost. The challenge with applying CDR to MVM is that unlike the summation example above, the commutative operation is the summation over individual matrix-element vector-element products. For example, consider a matrix row $\mathbf{A}_i$ being multiplied by a vector $\mathbf{x}$, where the dot product is $\mathbf{y}_i = \sum \mathbf{A}_{i,j}\mathbf{x}_j$. In this example, if the elements of $\mathbf{A}_i$ are reordered, either the vector $\mathbf{x}$ must be reordered in the same way for the entire matrix—even if not all rows are best reordered in the same way—or some metadata must be added such that elements of $\mathbf{A}_i$ can be associated with the corresponding element of $\mathbf{x}$.

Fortunately, sparse matrix operations already have this meta-data embedded in the format. In sparse matrices, non-zero values are stored as <data, coordinate> tuples so that zeros can be ignored. Therefore, so long as the tuples are reordered together, a sparse matrix can be rearranged without extra metadata. This creates a significant opportunity since sparse matrices are at the heart of many important problems, including scientific computing, graph analytics, pruned neural networks, and emerging simultaneous localization and mapping applications that are essential to augmented reality.

There are many sparse matrix formats available that attempt to optimize for different architecture behavior and metadata overheads. For specificity, consider the widely used compressed sparse row (CSR) format, where each element is a two-element tuple for the value and column with a separate row pointer array that specifies the index where each row starts. When applying CDR to a matrix stored in CSR format the values within a single row can be freely reordered. The reordering of elements can be modeled as an instance of the widely studied traveling salesman problem (TSP) [5], where each element is a node, and the edge weights are the hamming distances between elements.

III. Evaluation

We evaluate 12 sparse MVM workloads from a variety of problem domains, making comparisons to the previously published Base+XOR encoding technique [4]. The workloads are run on GPGPU-Sim, modeling a GTX480 with GDDR5 memory. The first two workloads, AlexNet-DeepComp and AlexNet-SkimCaffe, are inference operations using pre-trained neural networks [6], [7]. The next ten sparse matrices are from the SuiteSparse matrix collection [8]. The results, shown in Figure 1, indicate and show that Base+XOR encoding alone reduces the number of **1**s transmitted over the memory interface by 27%. CDR achieves an additional 21% reduction, for a total improvement of $1.89\times$ over the baseline interconnect.

IV. Conclusion

Previous work on encoding techniques to reduce data movement energy have shown promising results; however, these techniques do not leverage application level information. CDR shows that by exploiting application characteristics, such as the commutative property, the efficacy of data similarity based encodings can be significantly improved. When applied to sparse MVM, an important computational kernel in problem domains from machine learning to computational fluid dynamics, CDR incurs no metadata overhead and provides an additional 21% reduction in data movement energy. These results show significant potential for additional application level optimizations to reduce data movement energy.

References

[1] “Challenges for future computing systems,” Jan 2015.
[2] M. Anders, N. Rai, R. K. Krishnamurthy, and S. Borkar, “A transition-encoded dynamic bus technique for high-performance interconnects,” *IEEE J. Solid-State Circuits”*, vol. 38, no. 5, pp. 709–714, May 2003.
[3] S. Wang and E. Ipek, “Reducing data movement energy via online data clustering and encoding,” in *IEEE/ACM Intl. Symp. on Microarchitecture (MICRO)*, Oct 2016.
[4] D. Lee, M. O’Connor, and N. Chatterjee, “Reducing data transfer energy by exploiting similarity within a data transaction,” in *IEEE Intl. Symp. on High Performance Computer Architecture (HPCA)*, Feb 2018.
[5] S. Lin and B. W. Kernighan, “An effective heuristic algorithm for the traveling-salesman problem,” *Operations Research*, vol. 21, no. 2, pp. 498–516, 1973.
[6] S. Han, H. Mao, and W. Dally, “Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding,” in *Intl. Conf. on Learning Representations (ICLR)*, Feb 2016.
[7] J. Park, S. Li, W. Wen, P. Tang, H. Li, Y. Chen, and P. Dubey, “Faster cnns with direct sparse convolutions and guided pruning,” in *Intl. Conf. on Learning Representations (ICLR)*, April 2017.
[8] T. A. Davis and Y. Hu, “The university of florida sparse matrix collection,” *ACM Transactions on Math Softwware (TOMS)*, vol. 38, no. 1, pp. 1:1–1:25, Dec. 2011.