Reducing Memory Access Latencies using Data Compression in Sparse, Iterative Linear Solvers

All-College Thesis Defense

Neil Lindquist

April 16th, 2019

Motivation

- Sparse systems of linear equations used in many computations
- Iterative solvers are used
- Spend most of the time fetching data

Mathematics of Conjugate Gradient

- Solving $\mathbf{A}\vec{x} = \vec{b}$

Mathematics of Conjugate Gradient

- Solving $\mathbf{A}\vec{x} = \vec{b}$
- Minimizing $f(\vec{x}) = \frac{1}{2}\vec{x}^T\mathbf{A}\vec{x} - \vec{b} \cdot \vec{x}$

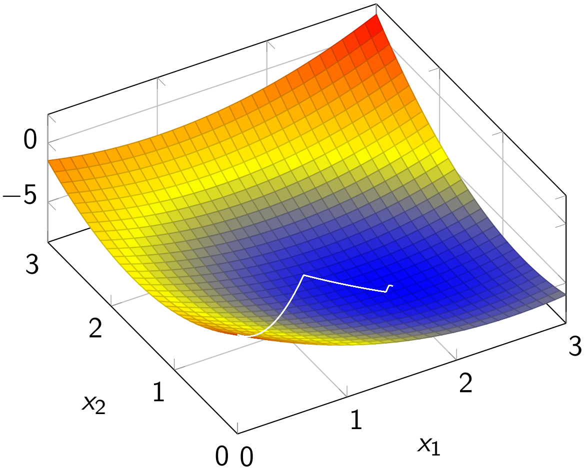
Mathematics of Conjugate Gradient

- Solving $\mathbf{A}\vec{x} = \vec{b}$
- Minimizing $f(\vec{x}) = \frac{1}{2}\vec{x}^T\mathbf{A}\vec{x} - \vec{b} \cdot \vec{x}$
- If $\mathbf{A}$ is symmetric, then $\nabla f(\vec{x}) = \mathbf{A}\vec{x} - \vec{b}$

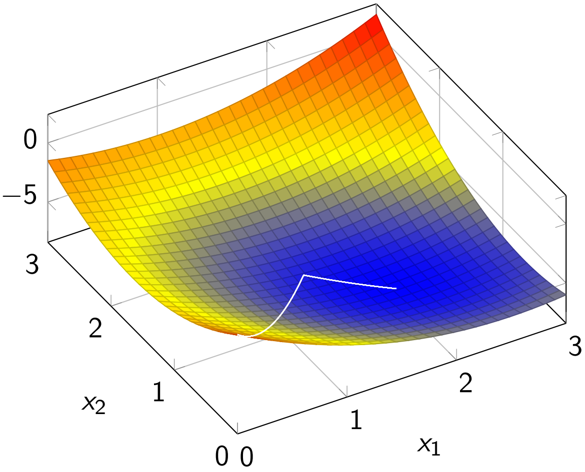
Mathematics of Conjugate Gradient

- Solving $\mathbf{A}\vec{x} = \vec{b}$
- Minimizing $f(\vec{x}) = \frac{1}{2}\vec{x}^T\mathbf{A}\vec{x} - \vec{b} \cdot \vec{x}$
- If $\mathbf{A}$ is symmetric, then $\nabla f(\vec{x}) = \mathbf{A}\vec{x} - \vec{b}$
- $-\nabla f(\vec{x})$ is in direction of maximal decrease

Mathematics of Conjugate Gradient



Mathematics of Conjugate Gradient

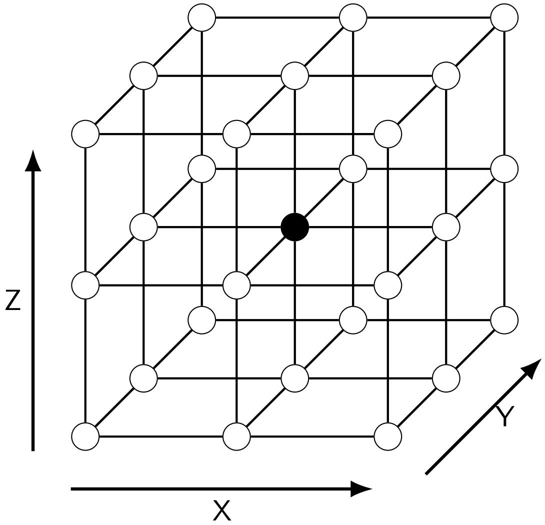


Test Problem

- Approximating the steady state heat equation in 3 dimensions
  - Discretized with a 27-point stencil

Test Problem

- Approximating the steady state heat equation in 3 dimensions
  - Discretized with a 27-point stencil



Solver Description

- Preconditioned Conjugate Gradient was used
- 3-level multigrid preconditioner with Symmetric Gauss-Seidel smoother

Compressed Sparse Row Format

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 5 & 8 & 0 & 0 \\ 0 & 0 & 3 & 0 \\ 0 & 6 & 0 & 4 \end{bmatrix}$$

Row	0	1	2	3
Values	[1]	[5, 8]	[3]	[6, 4]
Indices	[0]	[0, 1]	[2]	[1, 3]

Main Data Structures

1. Vector Values
2. Matrix Values
3. Matrix Indices

Compression Methods

- Mixed Floating Point Precision
- SZ Compression
- Elias Gamma and Delta Codings
- ZFP Compression
- Huffman Coding
- Op Code Compression

Compression Methods

- Mixed Floating Point Precision
- SZ Compression
- Elias Gamma and Delta Codings
- ZFP Compression
- Huffman Coding
- Op Code Compression

Mixed Floating Point Precision

- Two versions of floating point numbers
- Trade off between storage and precision
- Certain vectors can be lower precision without slowing convergence

Squeeze “SZ” Compression

- Predicts each value from the previous few
- Enforces minimum accuracy

Squeeze “SZ” Compression

- Predicts each value from the previous few
- Enforces minimum accuracy
- Available prediction functions are based on the data

Squeeze "SZ" Compression

- Predicts each value from the previous few
- Enforces minimum accuracy
- Available prediction functions are based on the data
- Compression rate is highly dependent on local patterns in the data

Elias Gamma Coding

- Positive integers
- For each value, stores the size then the data
  - Very effective for small integers
  - Storing the difference from the previous index reduces the size of values

Elias Gamma Coding

- Positive integers
- For each value, stores the size then the data
    - Very effective for small integers
    - Storing the difference from the previous index reduces the size of values
- Elias Delta Coding is similar, but uses Gamma coding for the length

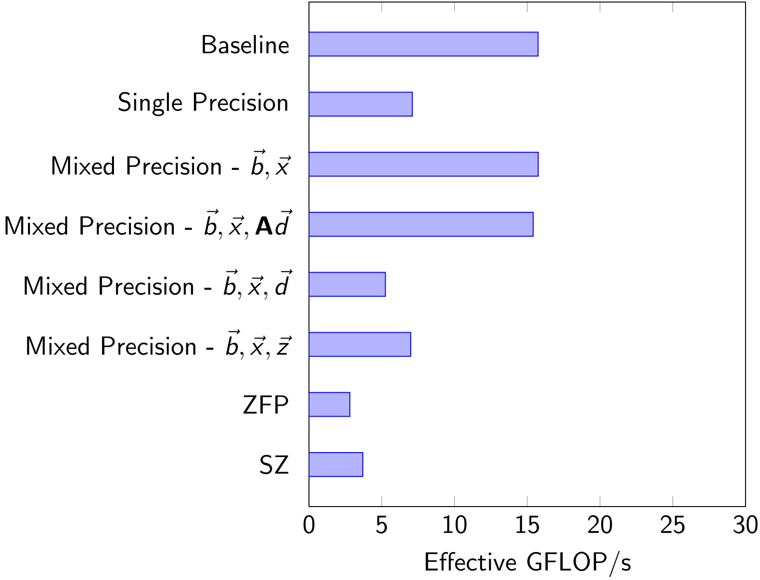
Elias Gamma Coding

- Positive integers
- For each value, stores the size then the data
  - Very effective for small integers
  - Storing the difference from the previous index reduces the size of values
- Elias Delta Coding is similar, but uses Gamma coding for the length
- Compression rate is only dependent on the magnitude of the values

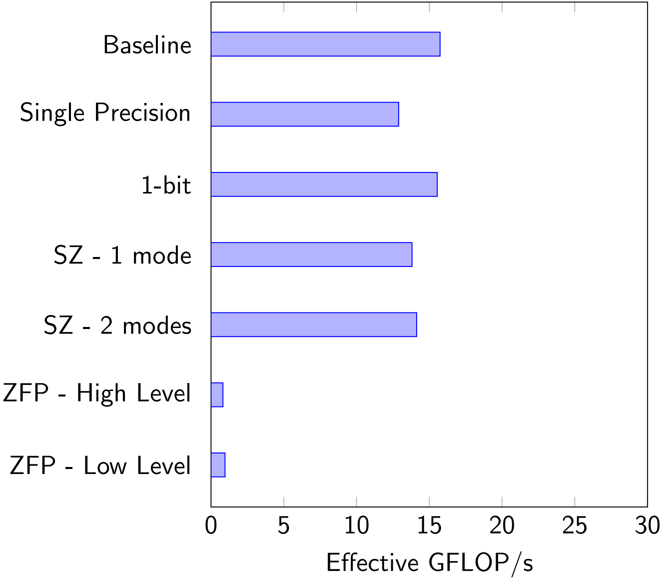
Timing Results

- 60 processes with 96^3 rows each
  - 53,084,160 total rows
- A 20-core, 2.2GHz, Intel Broadwell head node
- Plus five 8-core, 1.7GHz Intel Broadwell nodes

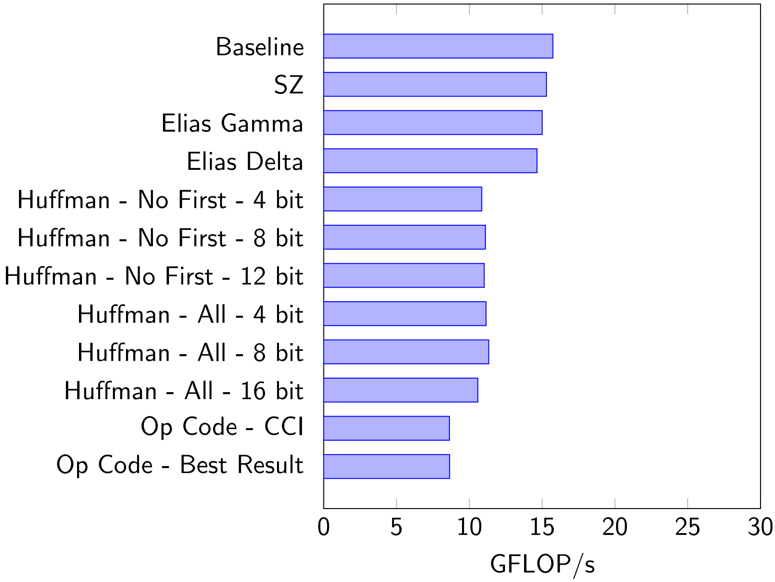
Vector Compression



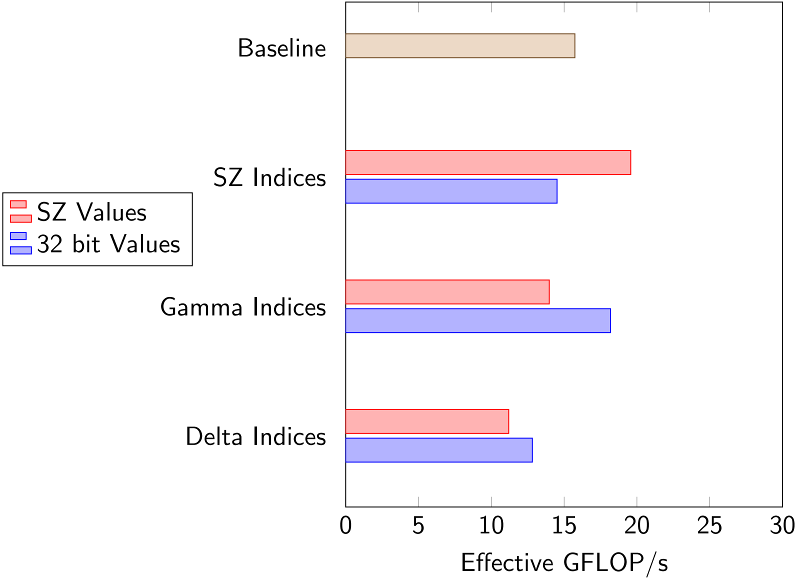
Matrix Value Compression



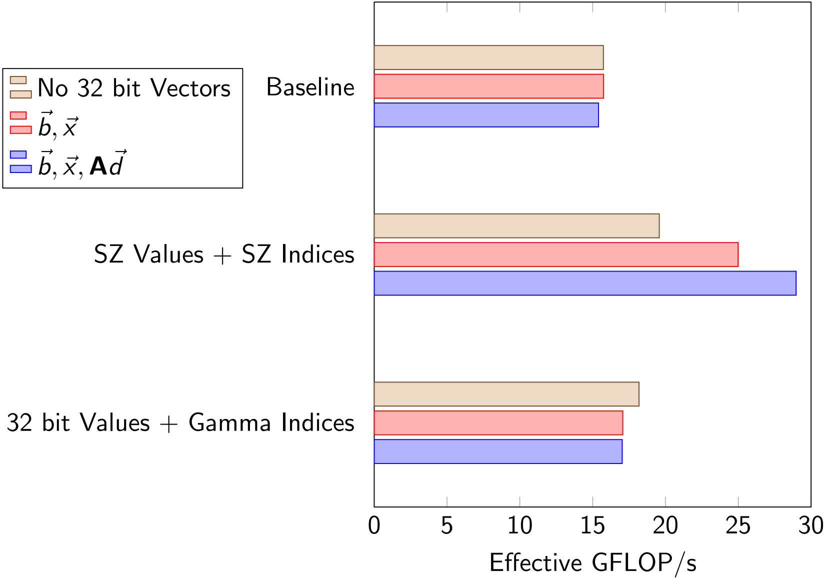
Matrix Index Compression



Matrix Value and Index Compression



Vector and Matrix Compression



Conclusion

- Iterative, sparse linear solvers are memory access bound
- Compressing key data structures provided an 84% increase in performance

Sources

`Github.com/Collegeville/HPCG-ZFP`