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

Neil Lindquist

All-College Thesis Defense

April 16th, 2019

Motivation

- Sparse linear systems used by many scientific computations
- Problems can be large, with millions of variables
- Arithmetic is faster than fetching data from memory

Solver Description

- Preconditioned Conjugate Gradient was used
  - Preconditioned with a 3-level multigrid preconditioner using Symmetric Gauss-Seidel step smoother
- Matrix store in CSR format
  - Stores the column index and value for each nonzero entry
- 3 compressible data structures
  - Vector Values
  - Matrix Indices
  - Matrix Values

Main Data Access Pattern

for row in rows **do**
  for nonzero entry in row **do**
    LOAD entry's value
    LOAD entry's column index
    LOAD vector value for column index
  end for
  WRITE vector value for row
end for

- need random vector reads
- need vector writes
- need both forward and backward iteration of matrix rows

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

- Single Precision takes half the storage space
- But drops from 15-17 significant digits to 6-9 digits
- Certain vectors can be lower precision without slowing convergence
    - $\vec{b}, \vec{x}, \mathbf{A}\vec{d}$

Squeeze "SZ" Compression

- Tries to predict each value from the previous few
- Enforces a minimum accuracy
- Available prediction functions are chosen based on the type of data
- Compression rate is highly dependent on local patterns in the data

Elias Gamma Coding

- Positive integers
- For each value, stores the number of bits needed 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
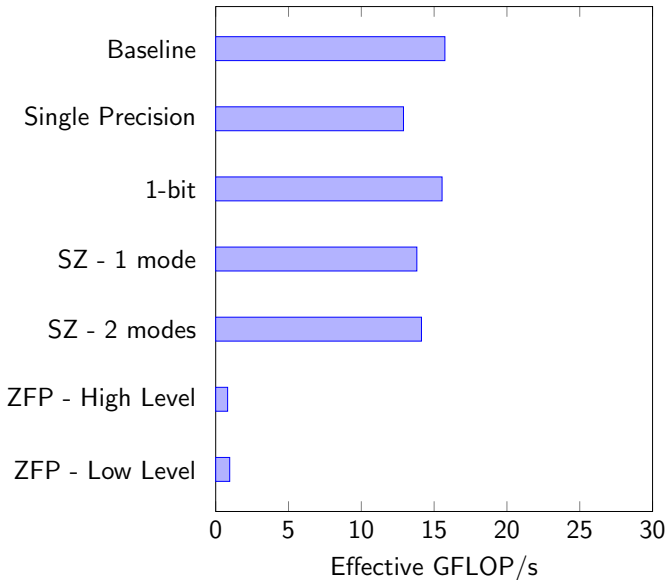- Compression rate is only dependent on the magnitude of the values

Timing Results

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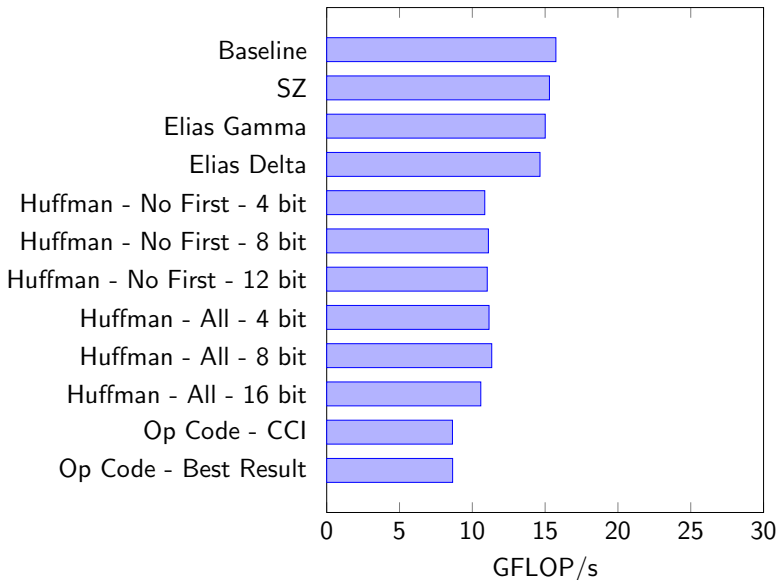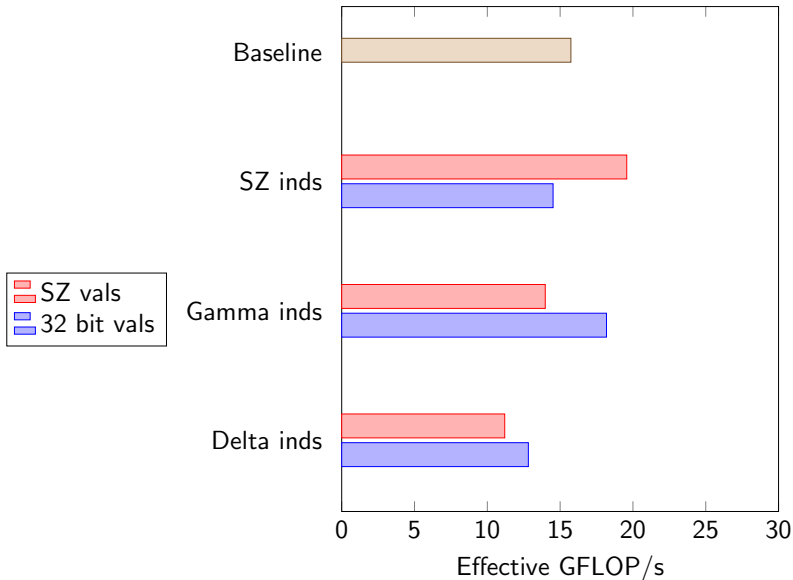- Plus five 8-core, 1.7GHz Intel Broadwell nodes
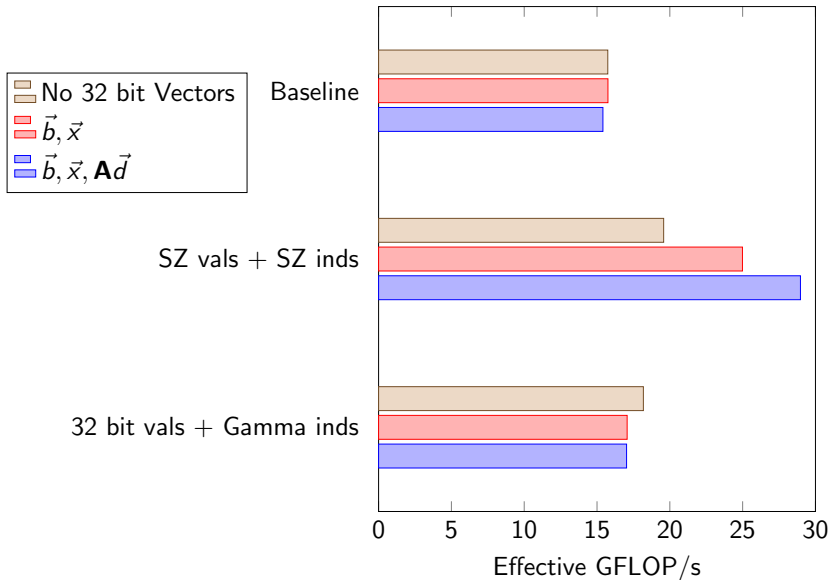
Vector Compression
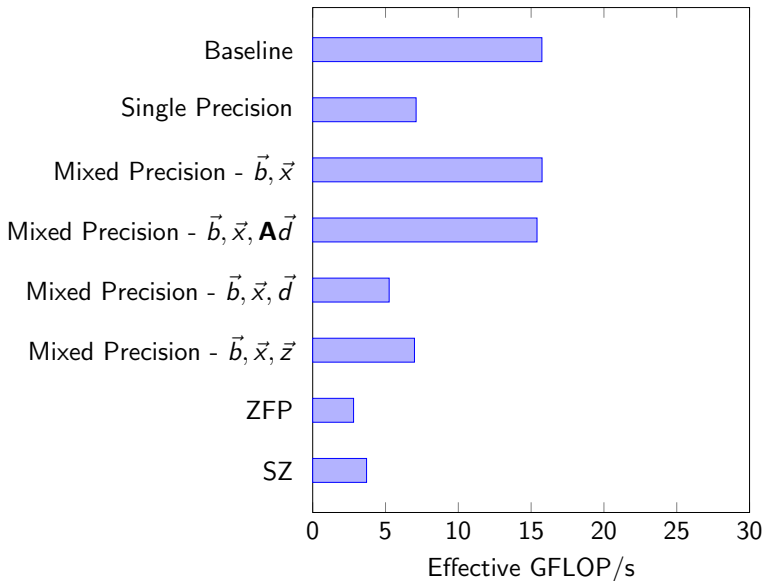
Matrix Value Compression

Matrix Index Compression

Matrix Value and Index Compression

Vector and Matrix Compression

Conclusion

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