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

Neil Lindquist

April 16th, 2019

Goal

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

Solver Description

- Preconditioned Conjugate Gradient was used for tests
  - Preconditioned with a 3 level multigrid
  - Symmetric Gauss-Seidel step smoother
- Matrix store in CSR format
  - Stores the column index and value for each nonzero entry

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

Analytical Performance Model

- System of equations that assumes no processor-level parallelism
- Solving for what conditions outperform the baseline gives:

$$\begin{aligned}
\text{totalDecode} <& 32.5375 - 0.037037 \cdot \text{vectEncode} \\
\text{matBytes} <& 12.9664 - 0.398506 \cdot \text{totalDecode} \\
& - 0.0147595 \cdot \text{vectEncode} \\
\text{vectBytes} <& 7.9998 + 8.27835 \cdot (12 - \text{matBytes}) \\
& - 3.29897 \cdot \text{totalDecode} - 0.122184 \cdot \text{vectEncode}
\end{aligned}$$

Simulation Performance Model

- Some processor level parallelism
- Conditions for outperforming the baseline:

Bytes	Matrix Index Decode	Matrix Value Decode	Vector Encode and Decode
1	66	41	$4.75 \geq 1.75 \cdot \text{decode} + \text{encode}$
2	51	14	$4.75 \geq 1.75 \cdot \text{decode} + \text{encode}$
3	66	40	$2 \geq 2 \cdot \text{decode} + \text{encode}$
4	0	0	$0 = \text{decode} = \text{encode}$
5	-	37	$4.75 \geq 1.75 \cdot \text{decode} + \text{encode}$
6	-	9	Not Possible
7	-	35	$0 = \text{decode} = \text{encode}$
8	-	0	$0 = \text{decode} = \text{encode}$