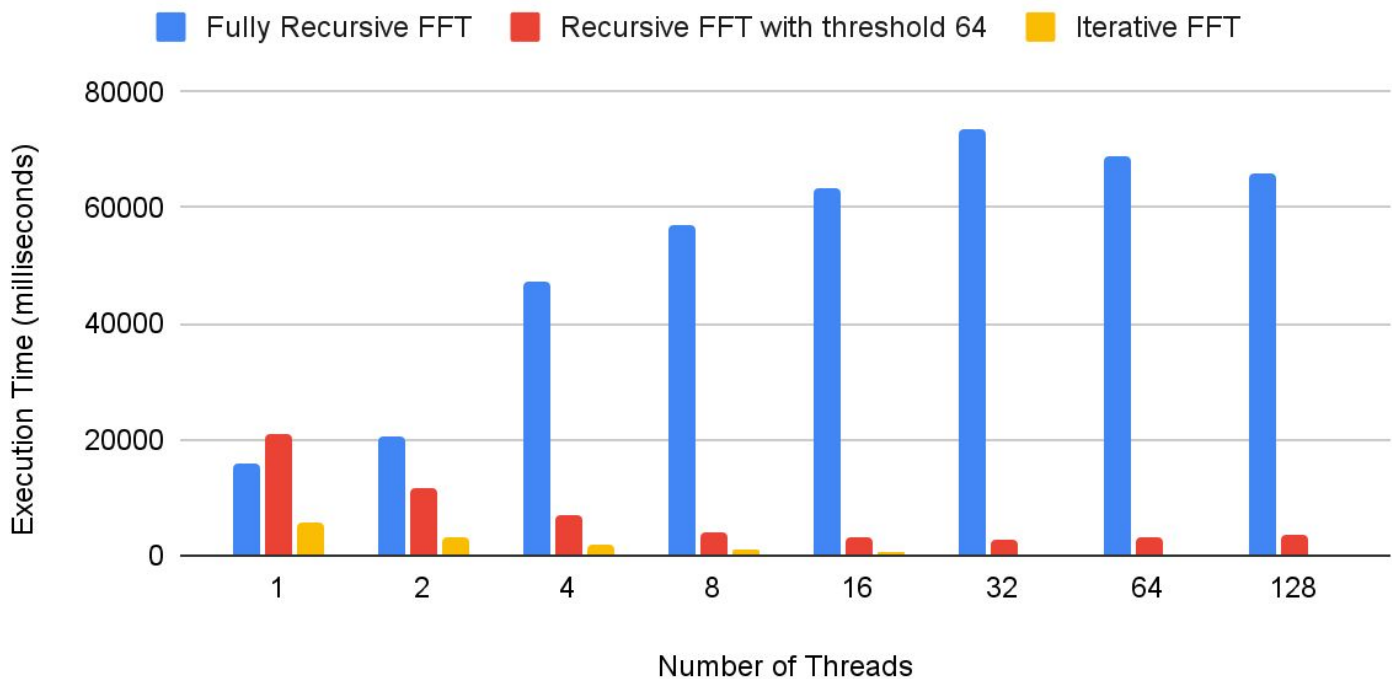


Parallelizing FFT with a Focus on Bandwidth Efficiency

Anuvind Bhat and Saatvik Suryajit Korisepati

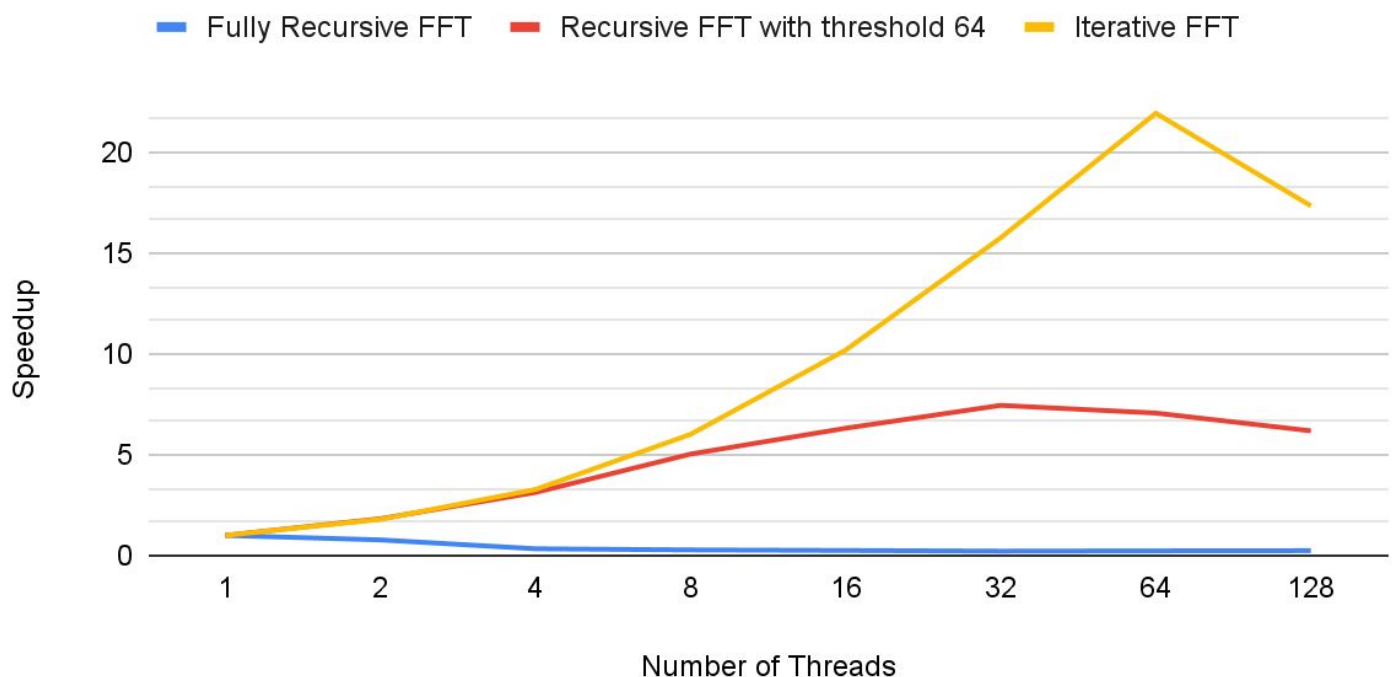
Scaling of Parallel FFT Implementations

Measured on the Bridges 2 super computer with data set of size 2^{25} elements



Speedup of Parallel FFT Implementations

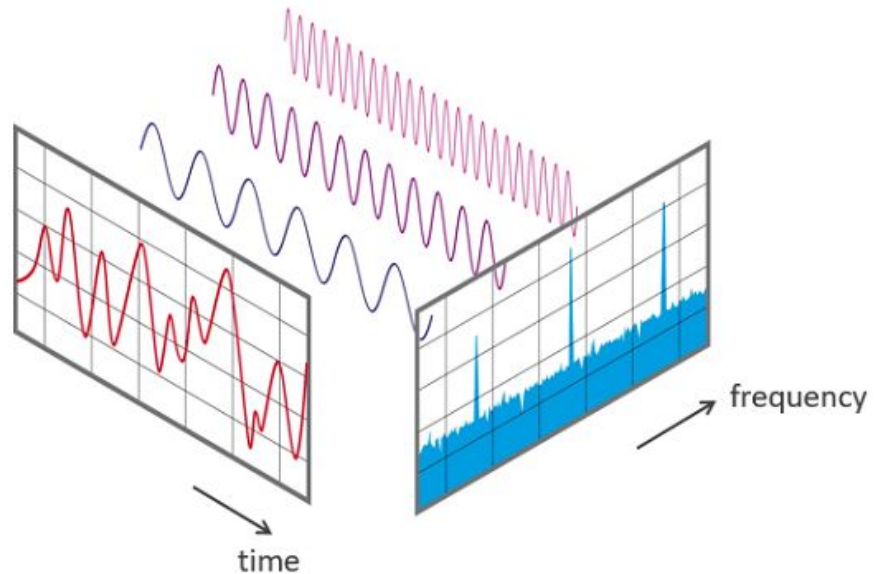
Measured on the Bridges 2 super computer with data set of size 2^{25} elements



FFT and our Approach

- What is Fast Fourier Transform (FFT)?
- Discrete Fourier Transform
- FFT (Fully Recursive, Recursive with Threshold, and Iterative)
- 2D FFT
- Image Compression

$$X_k = \sum_{n=0}^{N-1} x_n e^{-\frac{2\pi i}{N}nk}$$

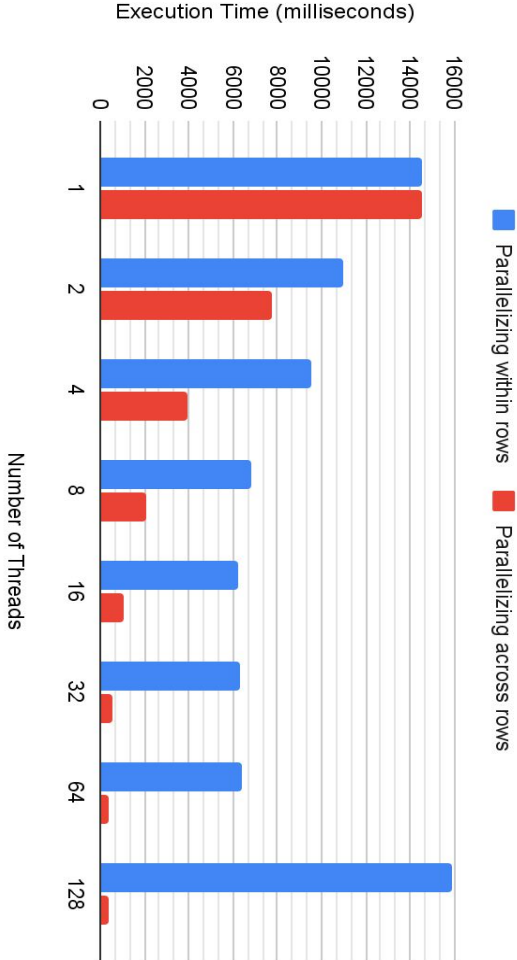


Main Optimizations

- Parallel pre-computation of Twiddle Factors
 - Loop collapse
- Switching to DFT in Recursive FFT at Threshold
- Iterative Algorithmic Modifications
- Bit Reversal allowing Butterfly Networks
- Chunking of Data that fits in L1/L2 Cache
- Matrix Transpose enabling better Spatial Locality
- Exploiting more available parallelism

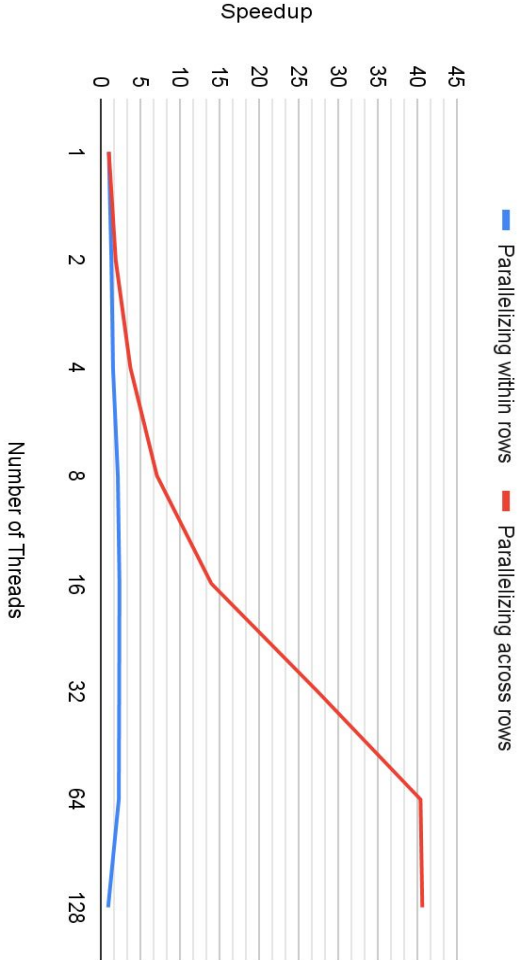
Scaling of 2D Parallel FFT

Measured on the Bridges 2 super computer with data set of size $2^{14} \times 2^{14}$ elements



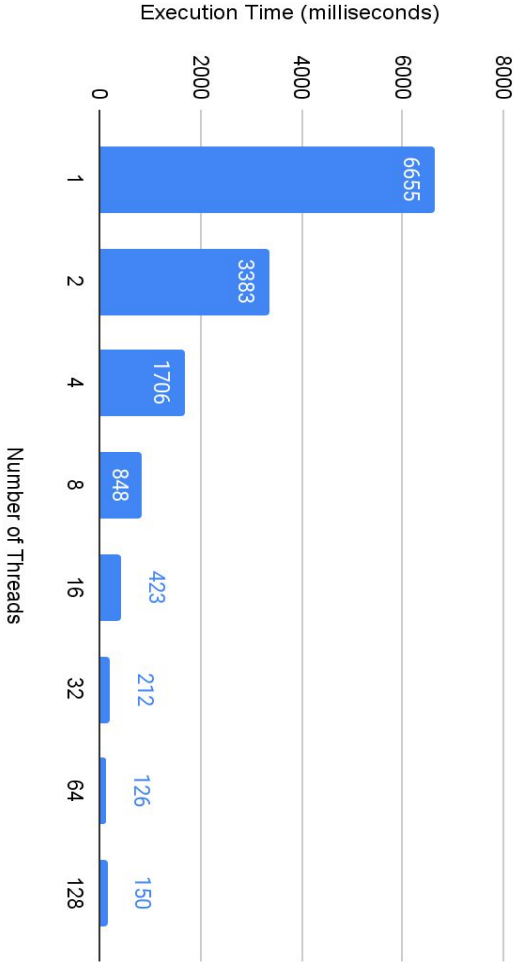
Speedup of 2D Parallel FFT

Measured on the Bridges 2 super computer with data set of size $2^{14} \times 2^{14}$ elements



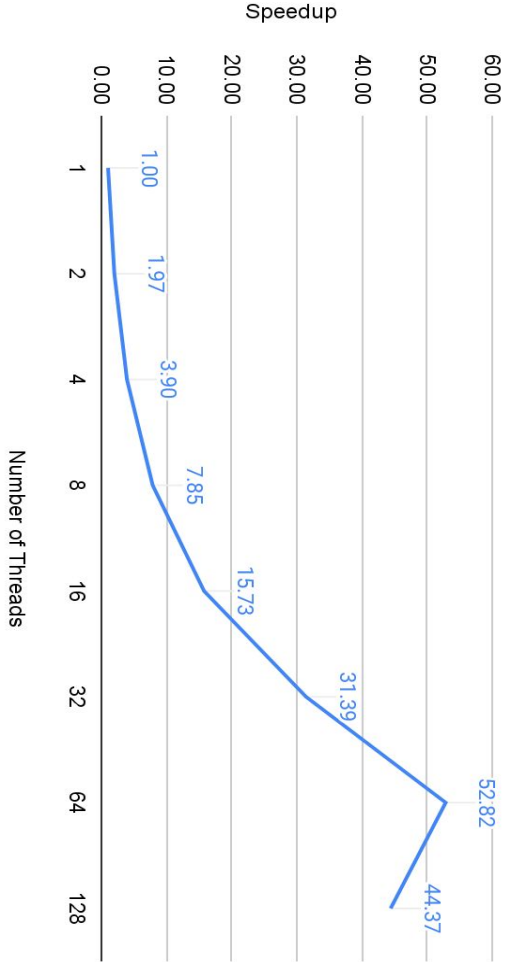
Scaling of DFT

Measured on the Bridges 2 super computer with data set of size 2^{15} elements



Speedup of DFT

Measured on the Bridges 2 super computer with data set of size 2^{15} elements



$$X_k = \underbrace{\sum_{m=0}^{N/2-1} x_{2m} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of even-indexed part of } x_n} + e^{-\frac{2\pi i}{N} k} \underbrace{\sum_{m=0}^{N/2-1} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}}_{\text{DFT of odd-indexed part of } x_n}$$

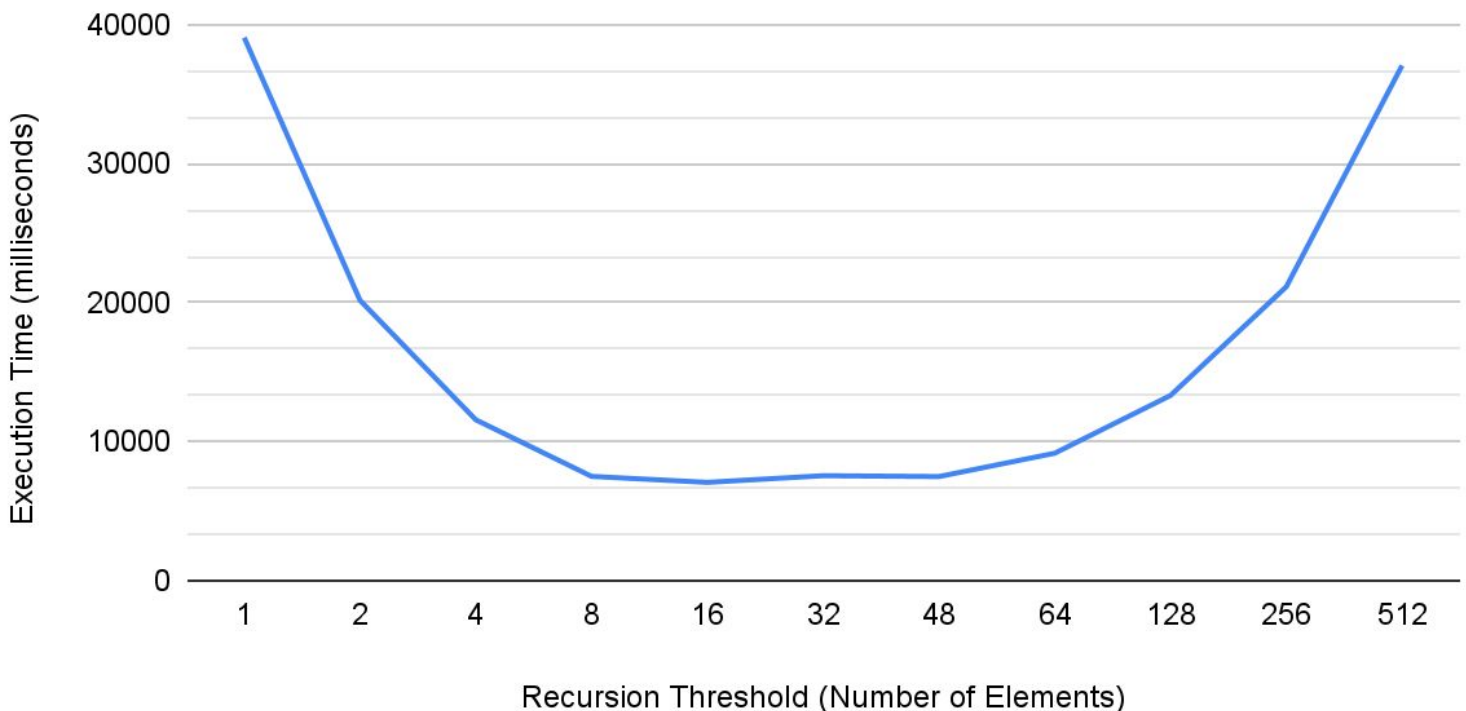
$$X_k = E_k + e^{-\frac{2\pi i}{N} k} O_k$$

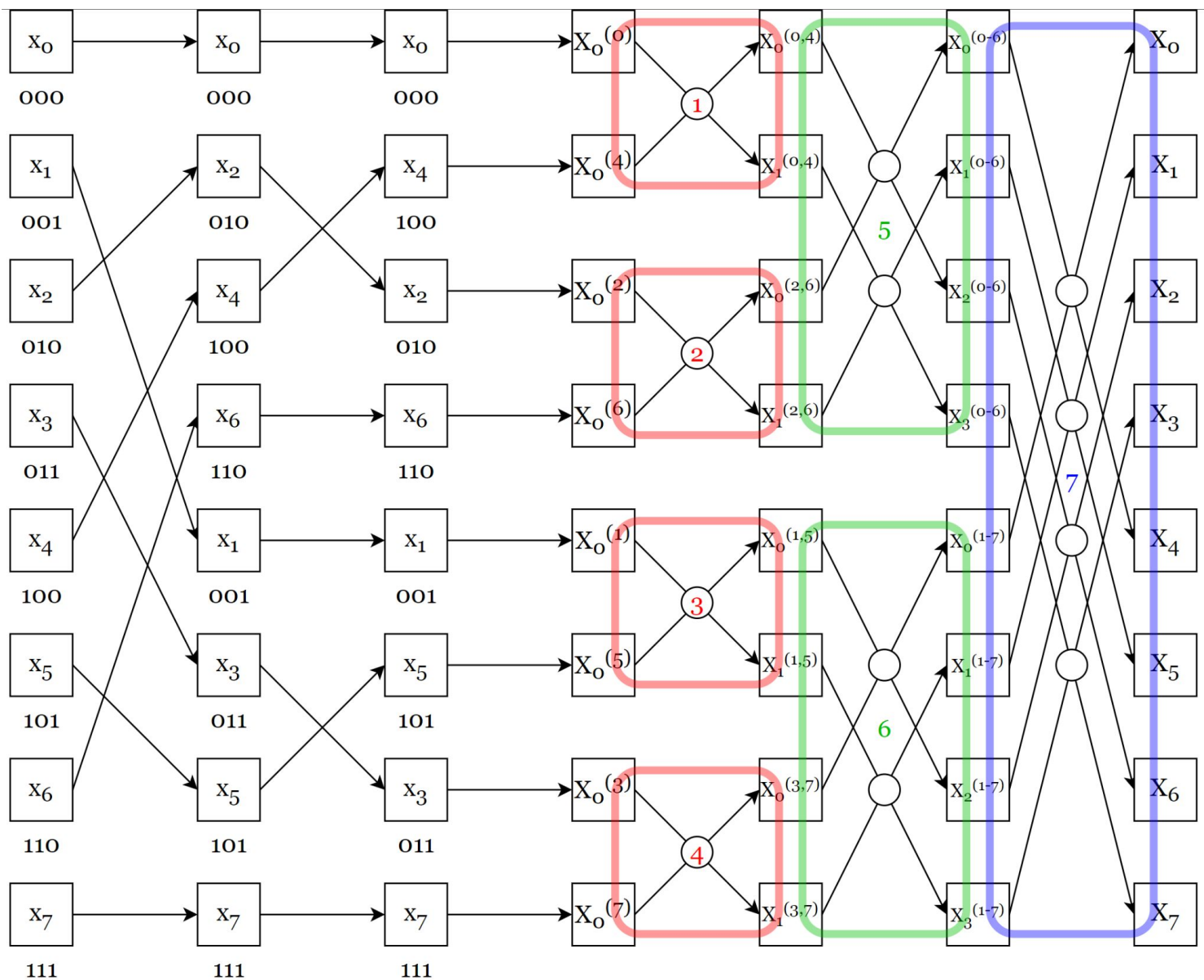
$$X_{k+\frac{N}{2}} = E_k - e^{-\frac{2\pi i}{N} k} O_k$$

- Sequential access within a recursive call
- Needless copying causes poor locality
- High overhead due to recursion at small sizes
- Better scaling and constant factors for DFT

Recursion Threshold for Recursive FFT vs Execution Time

Measured on GHC with data set of size 2^{25} elements and 8 threads

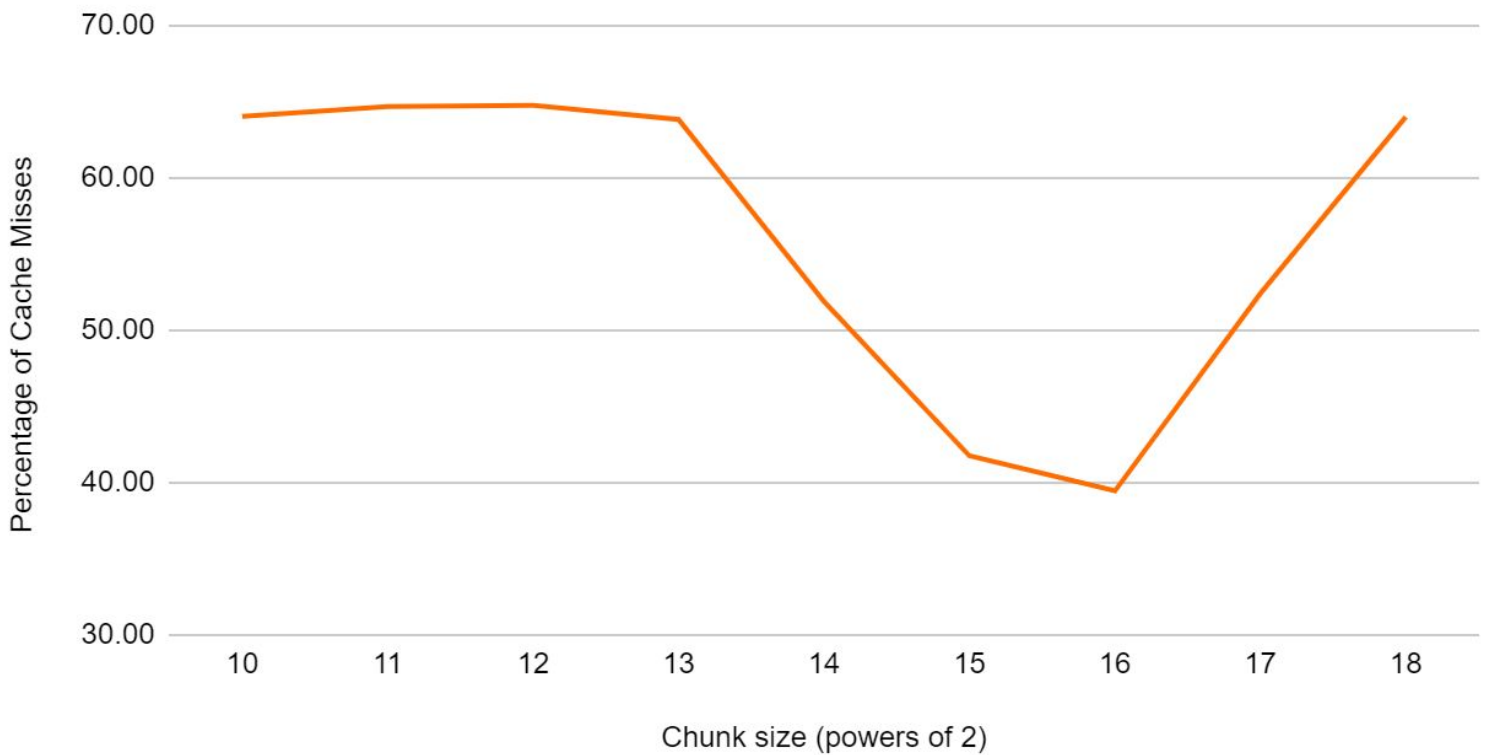




- Recursive Shuffle
 - Sort by reverse of the bits of the index
 - Bit reversal is a bijection and swaps are independent of each other
- Chunking (4 elements in cache)
 - 1 -> 2 -> 3 -> 4 -> 5 -> 6 -> 7
 - 5 reuses data brought in by 1 and 2
 - 1 -> 2 -> 5 -> 3 -> 4 -> 6 -> 7
 - $N \log(N/C)$ instead of $N \log(N)$

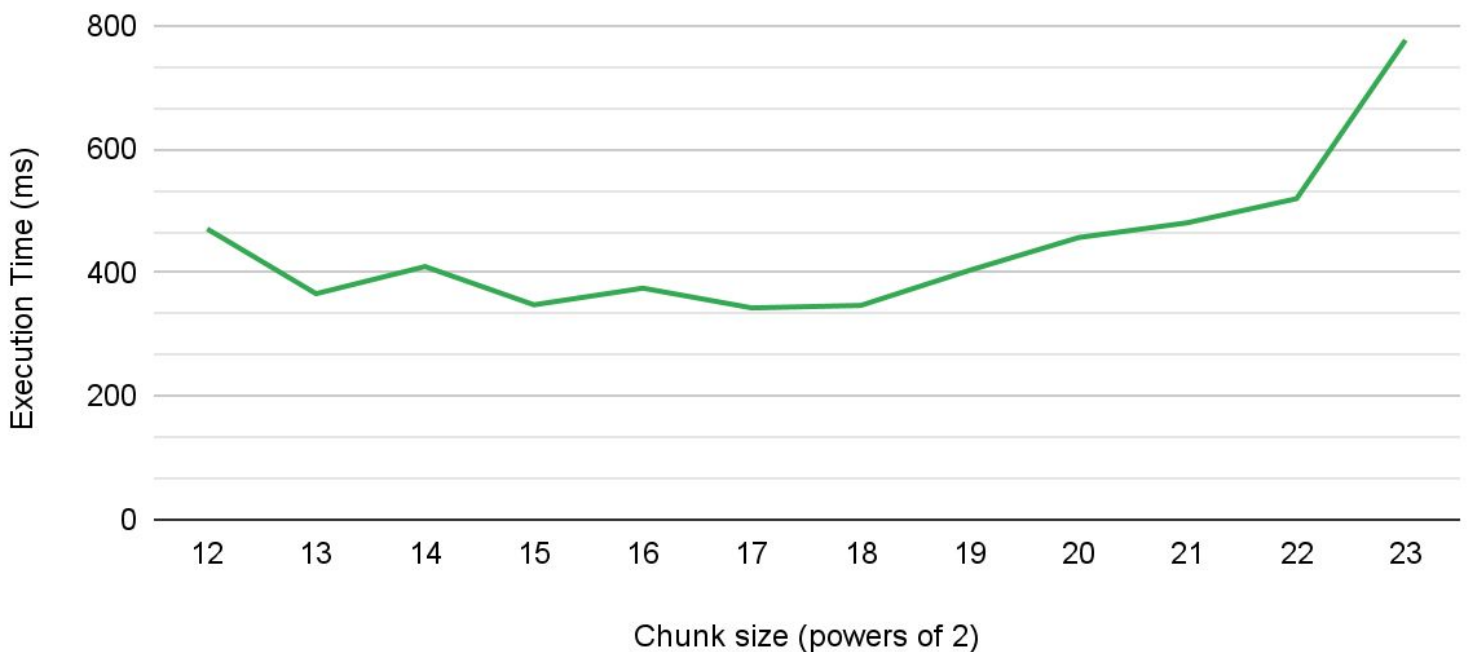
Number of Iterations in Cache vs % Cache Misses for Iterative FFT

Measured on GHC with dataset of size 2^{25} elements and 8 threads



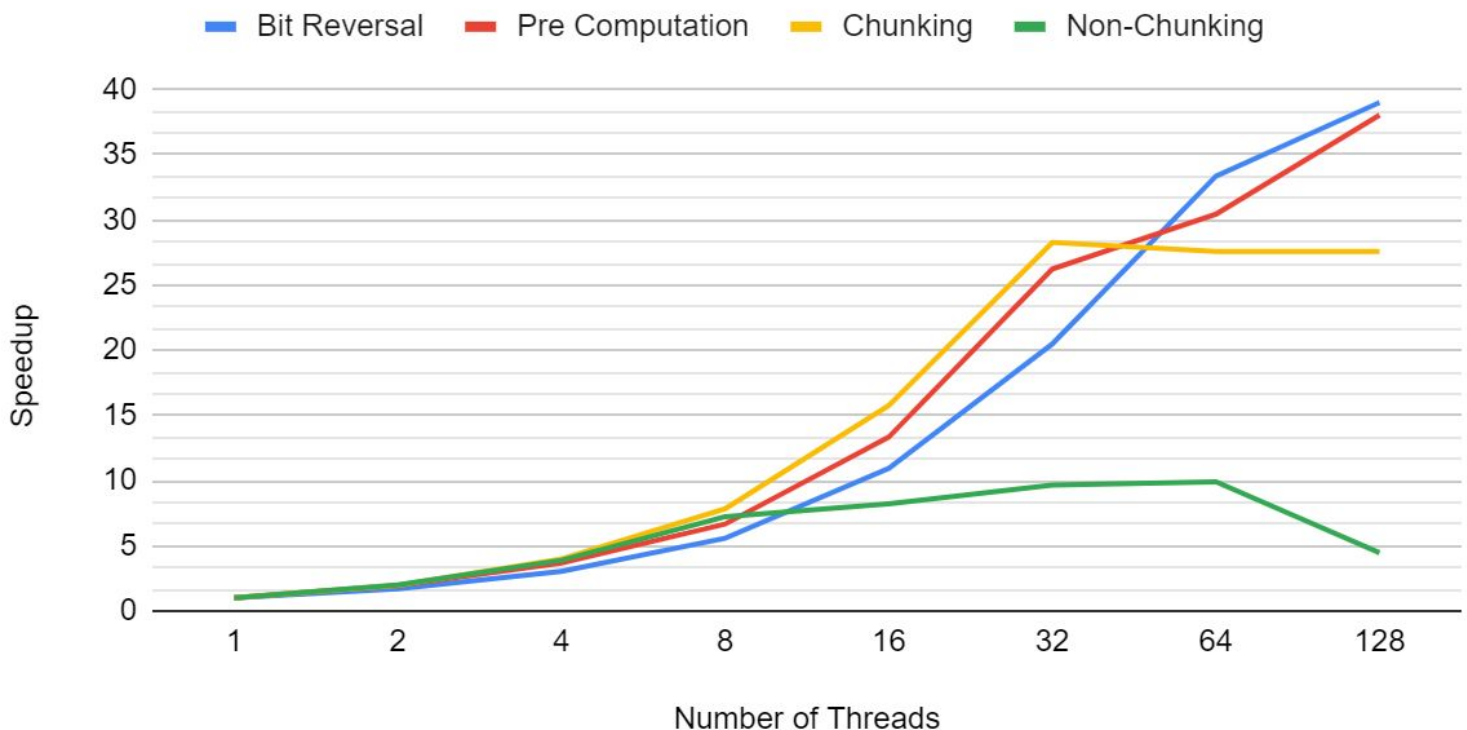
Number of Iterations in Cache vs Execution Time For Iterative FFT

Measured on GHC with data set of size 2^{25} elements and 8 threads



Speedup for Distinct Components of Iterative FFT

Measured on the Bridges 2 super computer with data set of size 2^{25} elements



- Chunking is effective
- Non-chunking still bandwidth bound
- Bit reversal (bit shifting) and pre computation (std::polar) are more compute bound

2-Dimensional FFT

- Steps
 - Perform in-place 1D FFT on the rows
 - Transpose the matrix
 - Perform in-place 1D FFT on the rows (which were the columns pre-transpose)
 - Transpose the matrix
- Parallel across rows or within row

Image Compression

- 2D FFT on each color obtaining the Fourier Coefficients
- Take top x% of coefficients by magnitude and remove the rest
- Store remaining coefficients and their indices
- Convert back to matrix by adding zeros for missing data
- Perform Inverse FFT on the matrix to obtain image

