FFTs of Arbitrary Dimensions on GPUs

Xiaobai Sun and Nikos Pitsianis
Duke University

September 19, 2007

At High Performance Embedded Computing 2007

MIT-LL

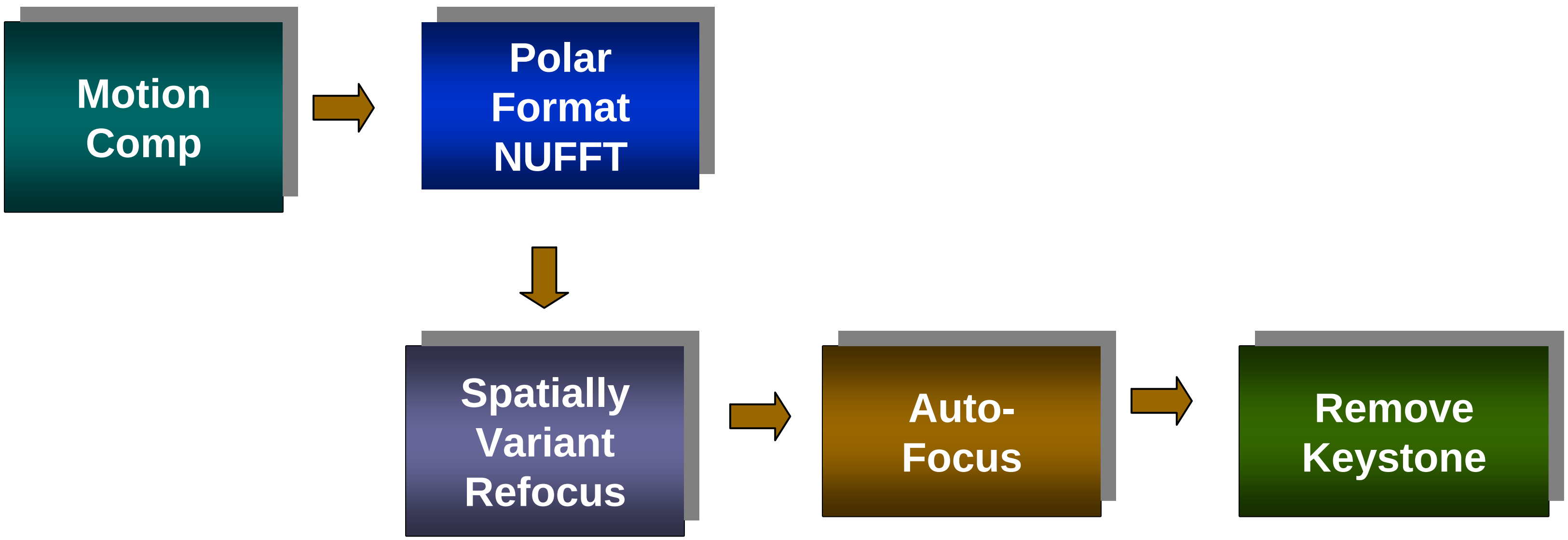
DARPA DSO

DEFENSE SCIENCES OFFICE

Overview

- Motivation
  - FFTs of arbitrary dimensions and their applications
  - Graphics processing units (GPUs)
- Basic facts on dimensionality
- FFTs on GPUs
  1. 2D FFT is chosen as the primitive one at API level
  2. 2D FFT performance is conveyed to FFTs of other dimensions
- Experimental results
- Discussion of related issues and works

Motivation : FFT Applications



From S. Bellofiore and H. Schmitt at Raytheon

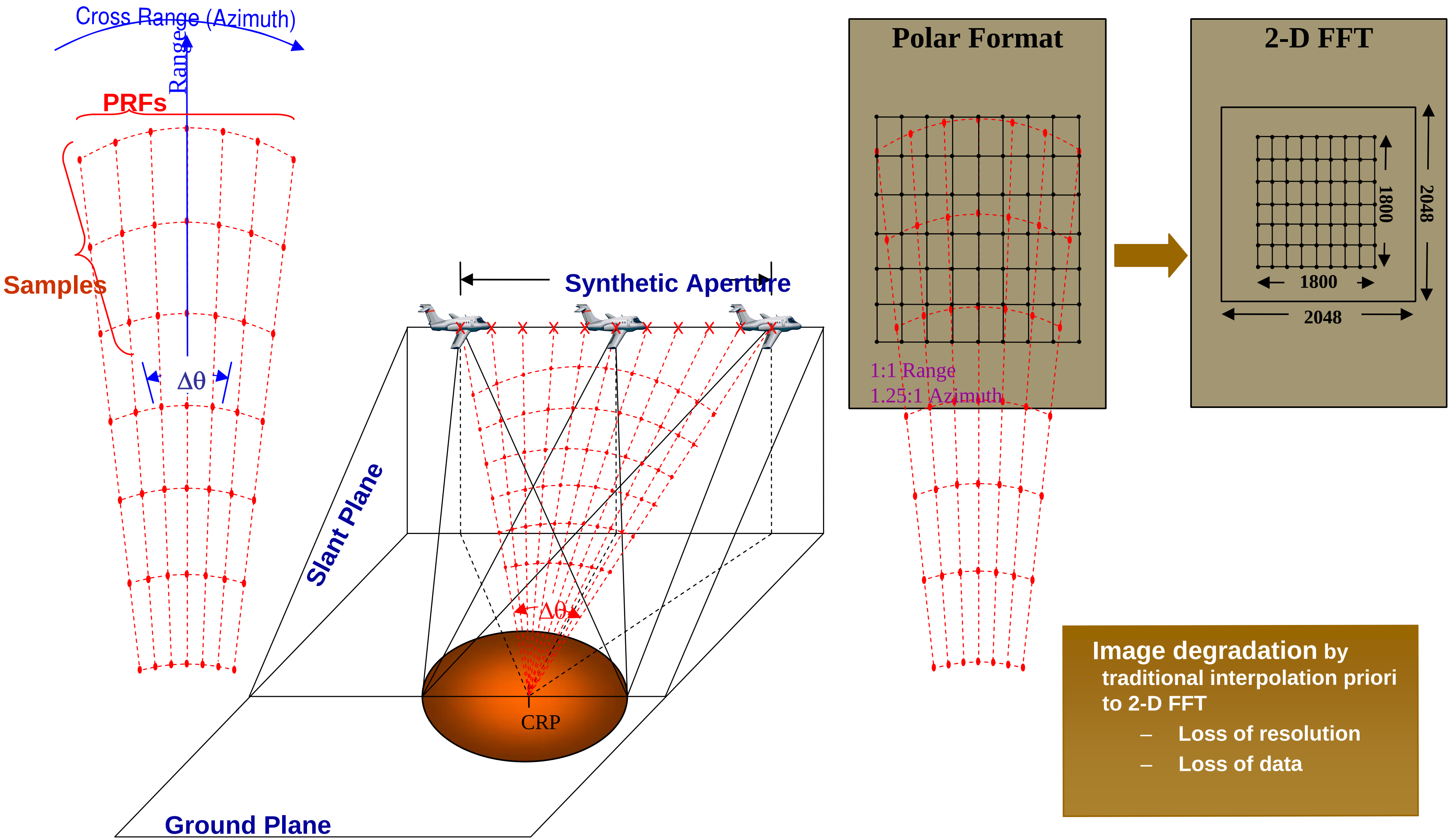


Image degradation by traditional interpolation priori to 2-D FFT

- Loss of resolution
- Loss of data

From S. Bellofiore and H. Schmitt at Raytheon

Motivation : GPU Architecture

GPU : Graphics Processing Unit

- Highly parallel multi-processors
- Affordable commodity product
- Initially dedicated to graphics processing and rendering
- Presently capable of co-processing on Desktop, Laptop
- Increasing programmability and API support
- Image processing & rendering
- GP-GPU

Basic Facts on Dimensionality

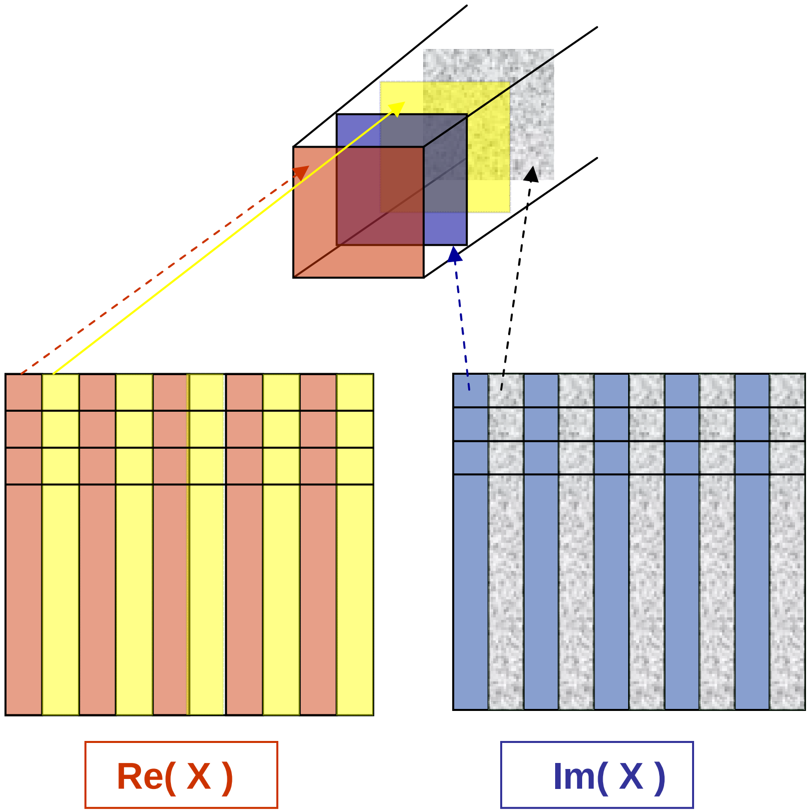
1. In mathematics, *FFTs are considered dimensionless* in the sense that the factorizations can be described in a unified, recursive representation with provided scaling factors some of which are dimension dependent. In computation, trivial scaling may be skipped.

2. In application, it is often required that phase-frequency information, or spatial and geometric relation, be provided explicitly at input and output. *FFT data are not shapeless*.

3. In architecture, extra dimensions are induced by the data access patterns most efficiently supported .
   FFT data are transfigured at different memory level.
   GPUs support fine-granularity, 2D access to memory frames at the API level

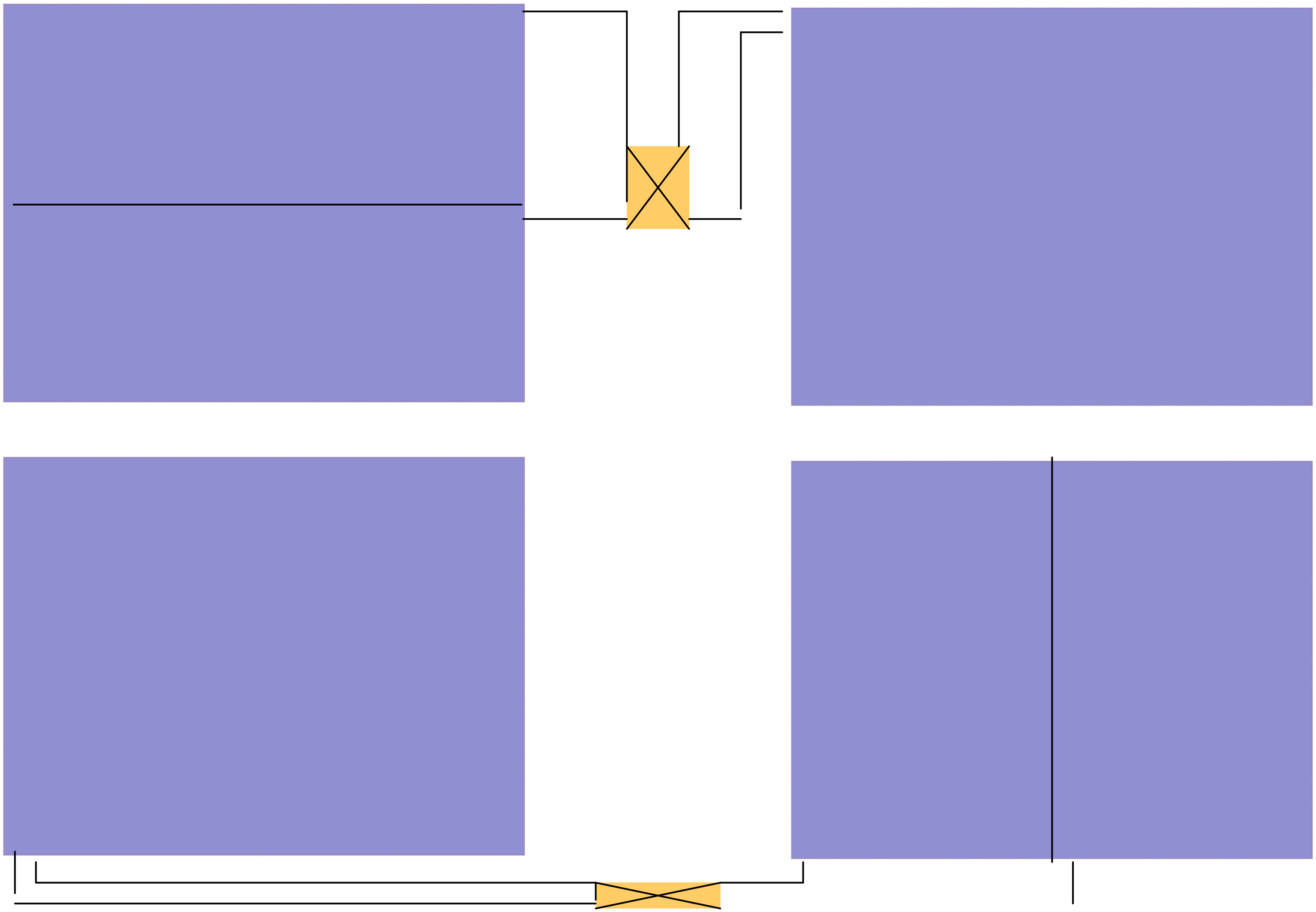
2D FFT as the Primitive

- 2D FFT $$Y := F_m \, X \, F_n$$
- 2D data placement
  - Two complex numbers per pixel vector (4 floating point numbers) : one at the front, one at the back
  - Even columns at the front layers, odd columns at the back layers
- 2D array operations through
  - utilizing best the architectural support of 2D data access at API level
  - Radix-2, radix 3 and mixed radices
- Direct 2D bit-reversal
  - Up to certain sub-array size
  - 2D data partitioning in large data array

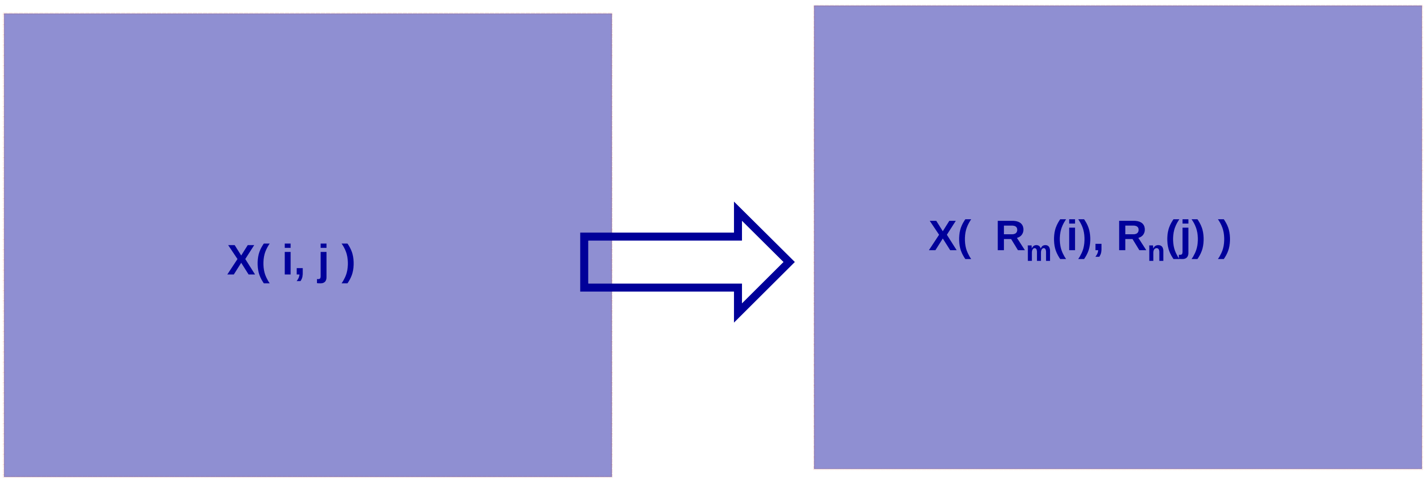


$$X(R(i), R(j)) := X(i, j), \qquad \forall (i, j)$$

Radix 2, Radix 3 and Mixed Radices

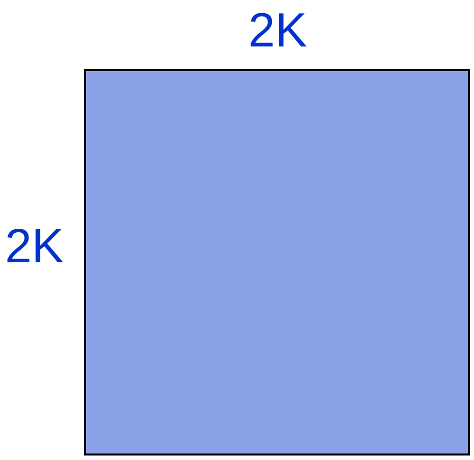


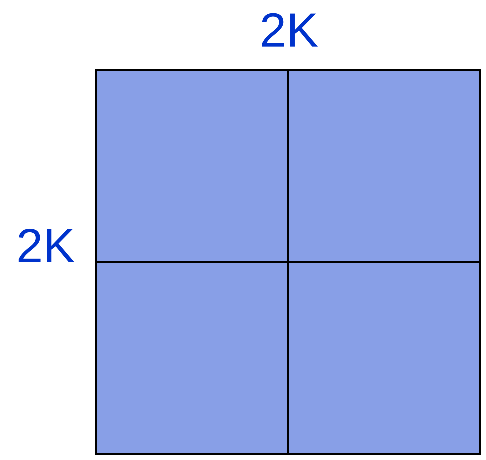
Direct Two Dimensional Bit Reversal



- Not one dimension after another
- Not recursion up to certain frame block size (low bits)
- For large data size, block swaps (bit reversal in high bits)

2D Bit Reversal

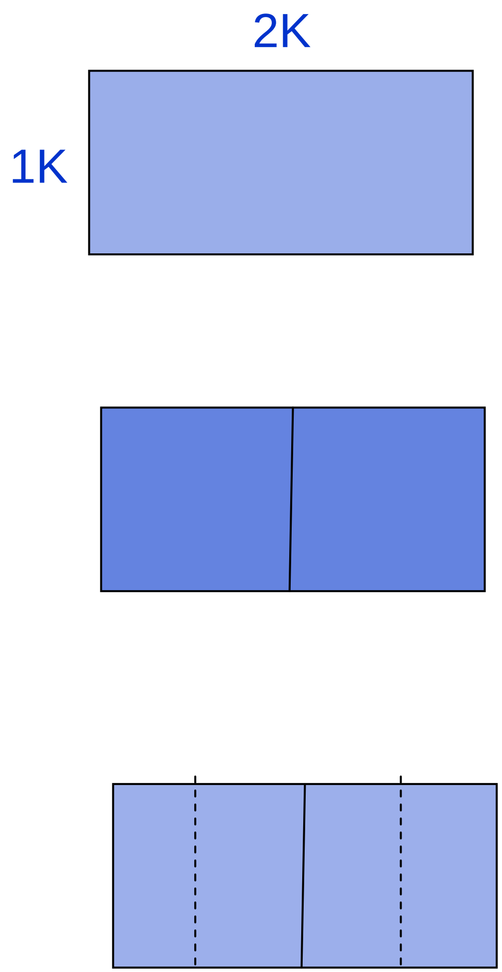




30.4 ms

19.6 ms

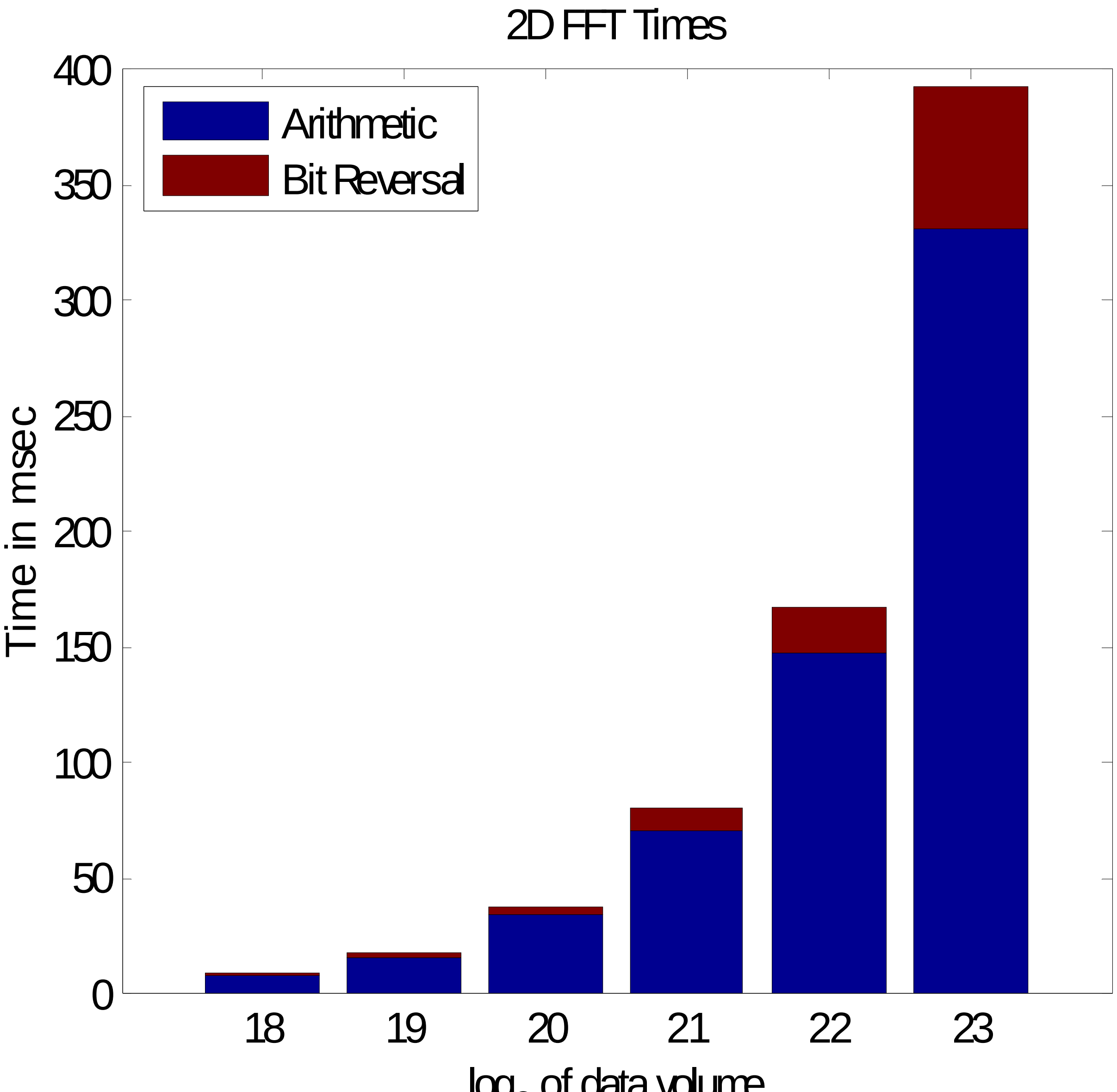
2D Bit Reversal

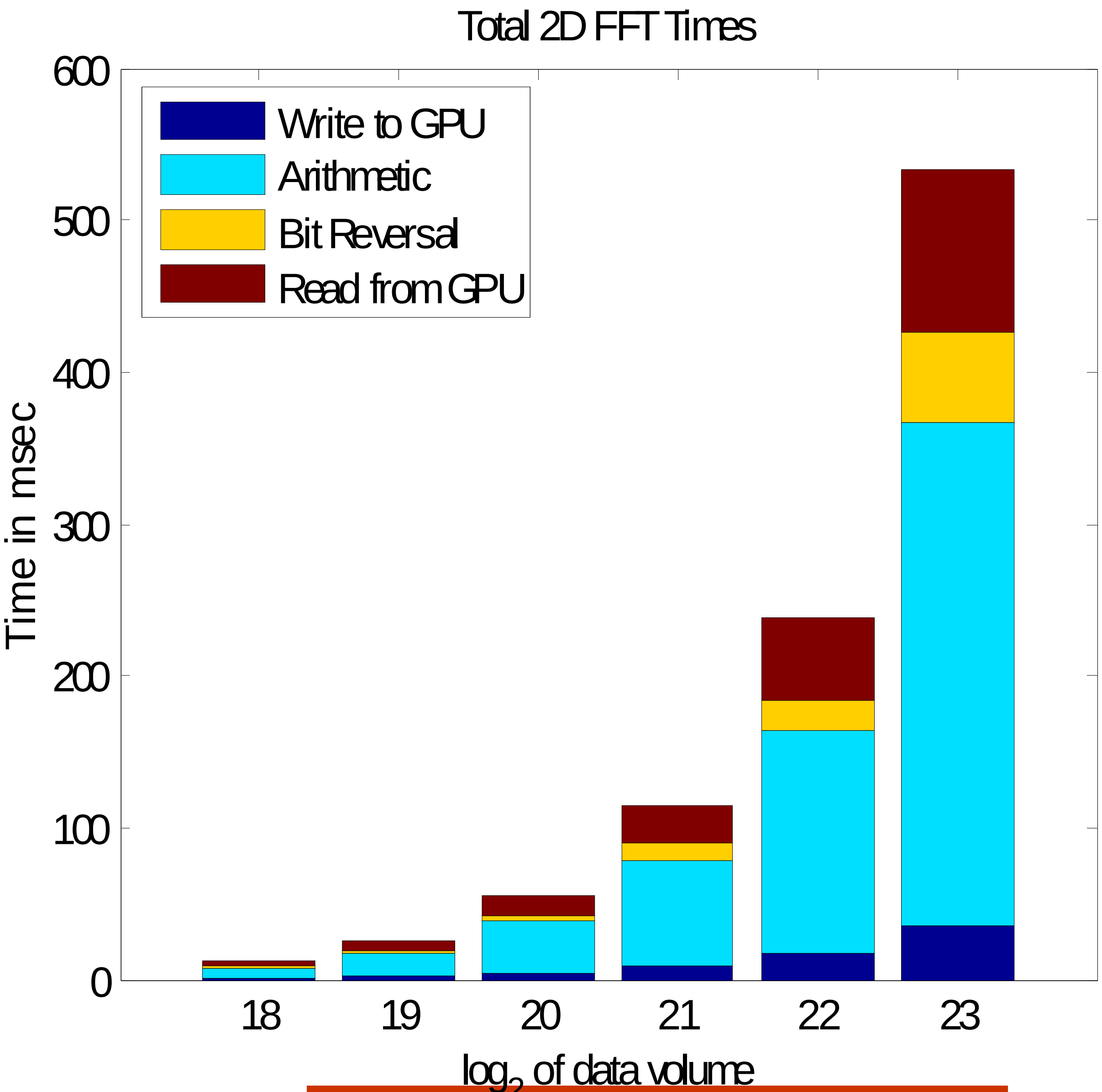


17.5 ms

9.8 ms

14.8 ms


2D FFT Times
Arithmetic
Bit Reversal
Time in msec
400
350
300
250
200
150
100
50
0
18
19
20
21
22
23
log2 of data volume


Total 2D FFT Times
Write to GPU
Arithmetic
Bit Reversal
Read from GPU
Time in msec
600
500
400
300
200
100
0
18
19
20
21
22
23
log2 of data volume

FFTs of Other Dimensions

- 1D FFT of size n

$$Y := (F_p X \odot W_{pq}) F_q, \qquad n = p \cdot q$$

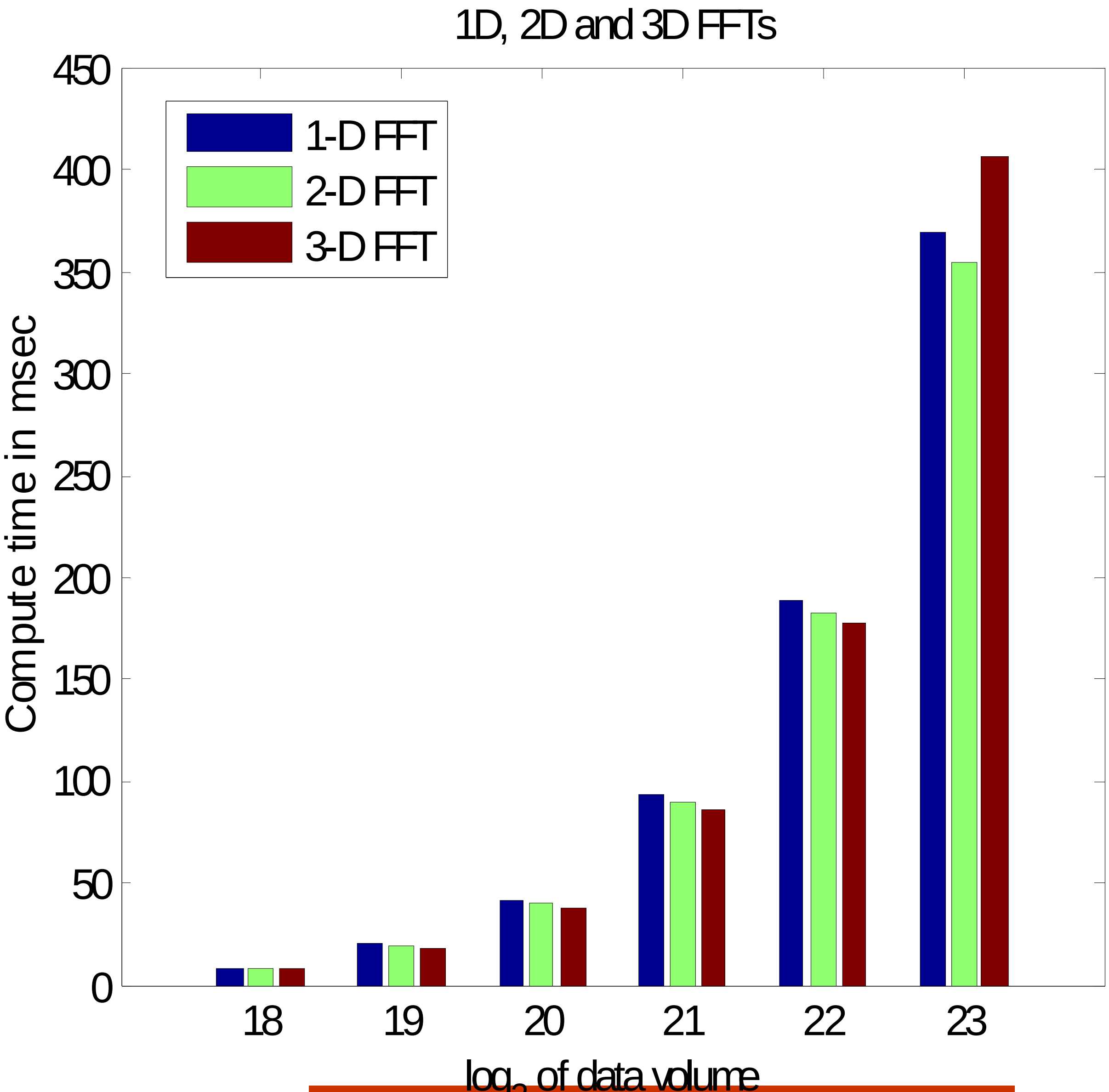
Add a scaling stage in 2D FFT

- 3D FFT of dimensions $\ell \times m \times n$

$$Y_{\ell m,n} := (F_\ell \otimes F_n) X_{\ell m,n} F_n$$

(In a simple case)

Skip a scaling stage in 2D FFT


1D, 2D and 3D FFTs
1-D FFT
2-D FFT
3-D FFT
Compute time in msec
450
400
350
300
250
200
150
100
50
0
18
19
20
21
22
23
log2 of data volume

Other Issues and Works

- Twiddle factors :
  - Pre-calculated, partially calculated, calculate on the fly
  - Numerical behavior
- Data loading and unloading
  - Data placement in main memory
  - A sequence of successive FFTs
- Automated tuning
- Other commodity products
  - IBM Cell
- FPGAs