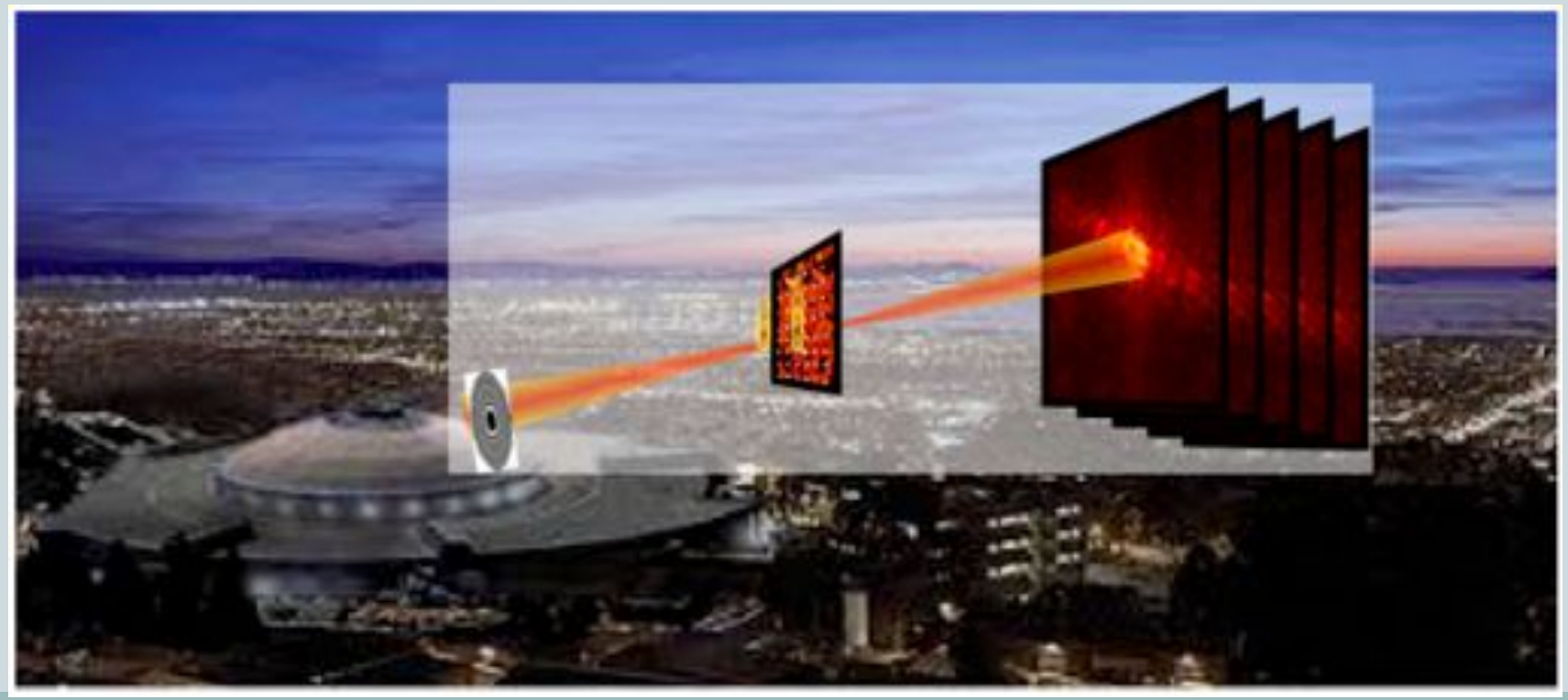
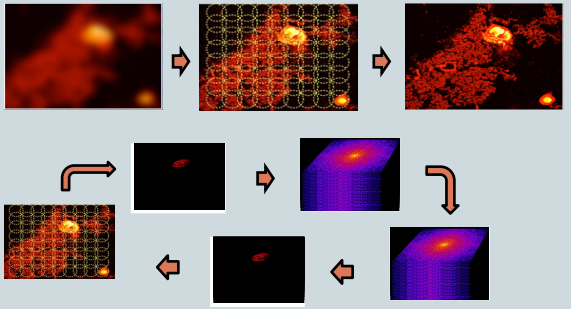
SHARP: Architecture

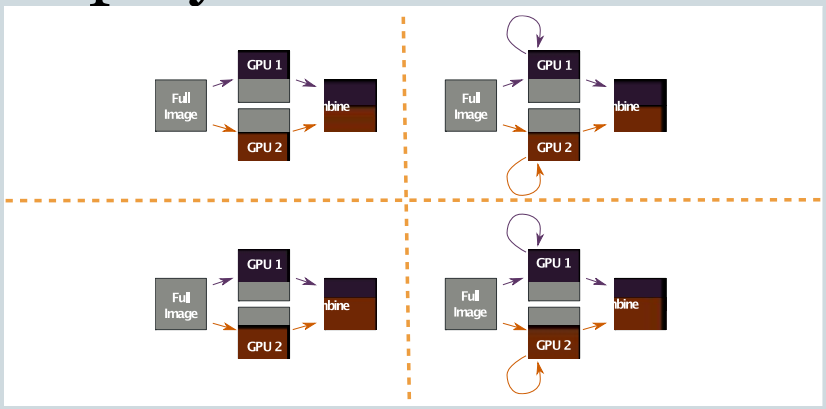


Stefano Marchesini, David Shapiro, Filipe Maia, Hari Krishnan, CAMERA

SHARP

- Core Algorithm
- Parallelism
- Performance Considerations
- Deployment



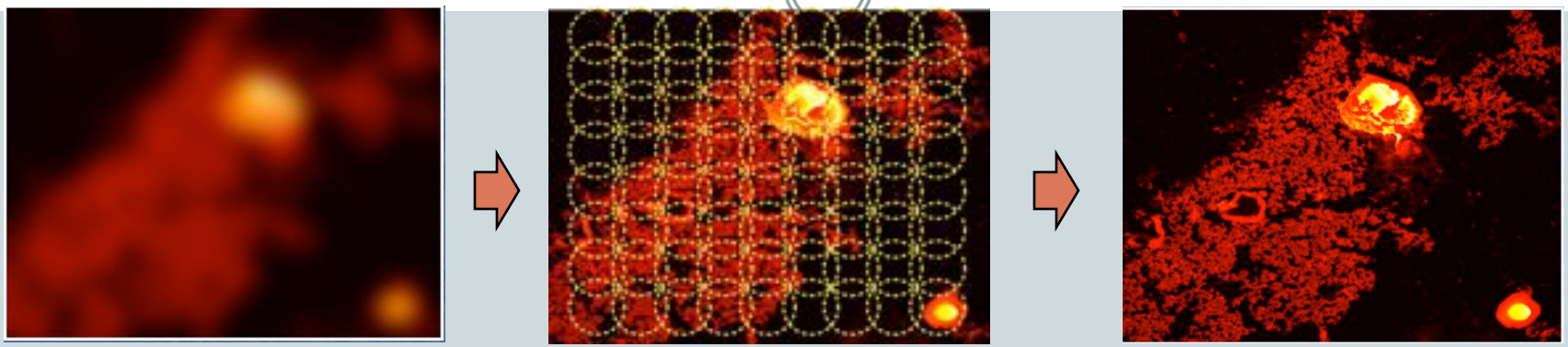


*Covers Architecture Design, Filipe will cover the Kernels & CXI Data Model

Core Algorithm

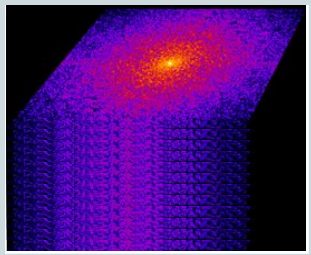
- Input/Output: Handles parallel read and write operations for SHARP. Efficiently reads and writes metadata and manages access to raw CXI data.
- Strategy: Contains the core logic on how to separate the problem into smaller pieces.
- Engine (Thrust/Cuda/OpenMP) : Performs the core image reconstruction algorithm.

Recap

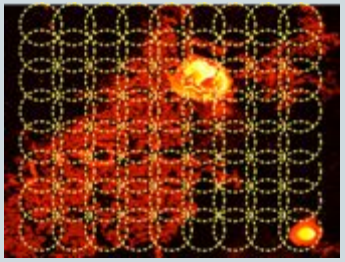


Split

FFT

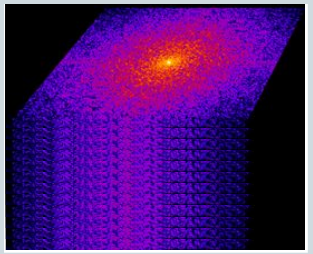


Solver
1. setup interface
2. #recons
3. write out CXI

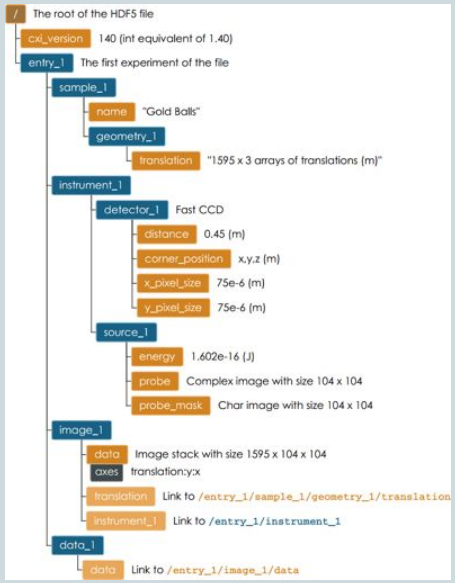


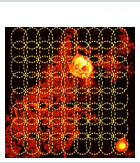
Overlap

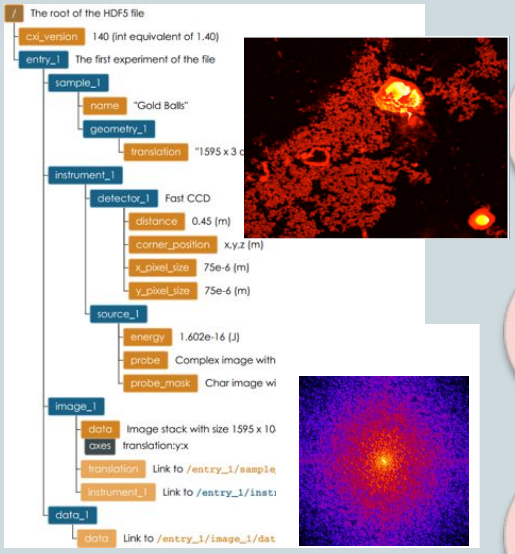
Inverse FFT



Handling Input & Output







Input CXI File:
data, probe,
probe_mask

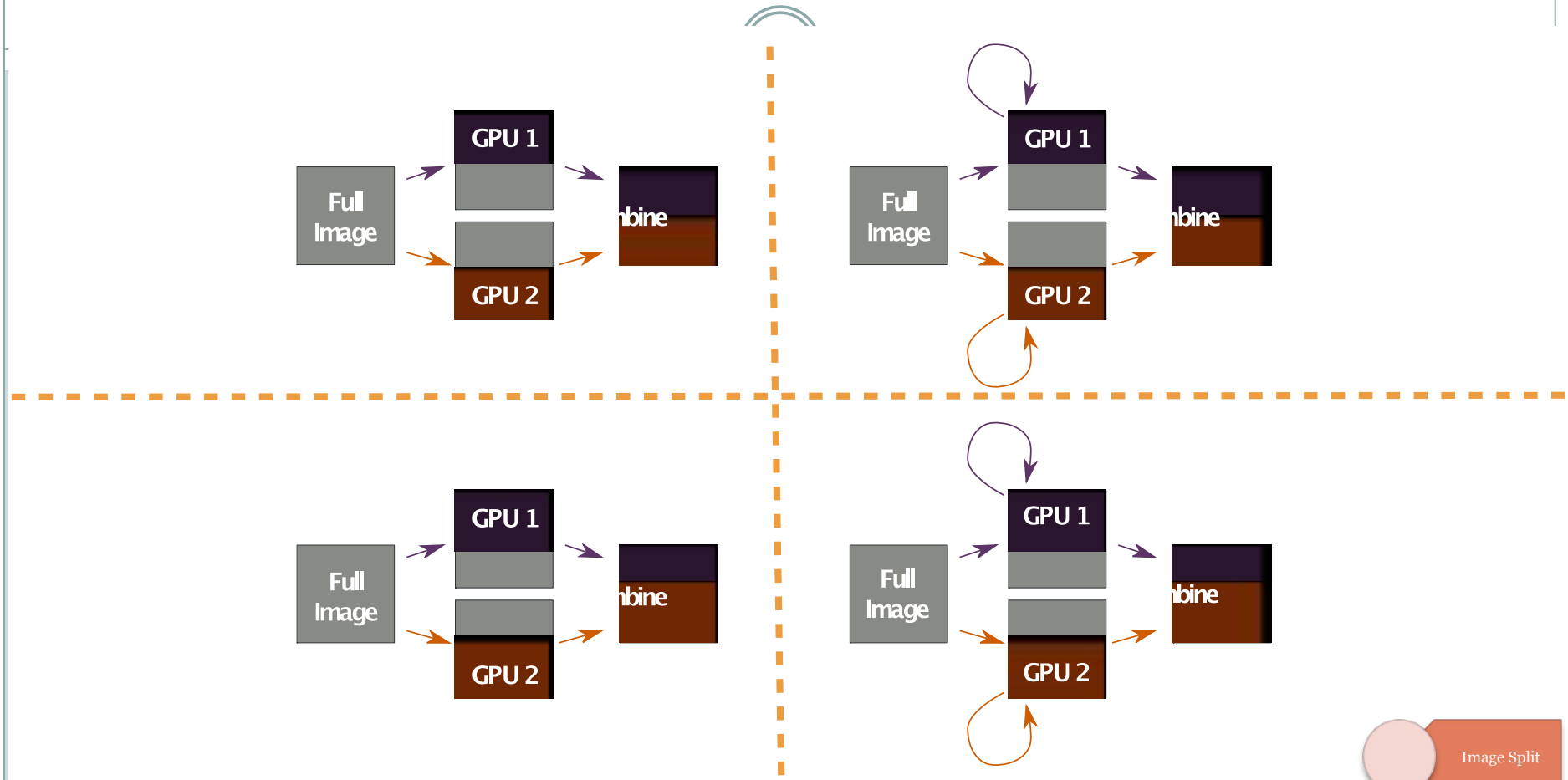
Output CXI File:
/image_1/process
/image_1/data

- Reads Input
- Loads Meta Data (in parallel)
- Calculates Frame Corners, Illumination Shifts, Crops Image (if needed)
- Generates Illumination (if not provided)
- Writes Output: Final image, illumination intensities, mask, and error

Parallelism: Strategy

- One-Dimensional: Divide image along first dimension.
- Linear: Divide frames equally among all processors.
- Grid: Divide image based on 2D regions.
- Communicator
  - Communicates images as well as intermediate frames and frame corners.
  - Provides Sum, Min, Max, and handles standard and complex data.

Parallelism: Multi-GPU

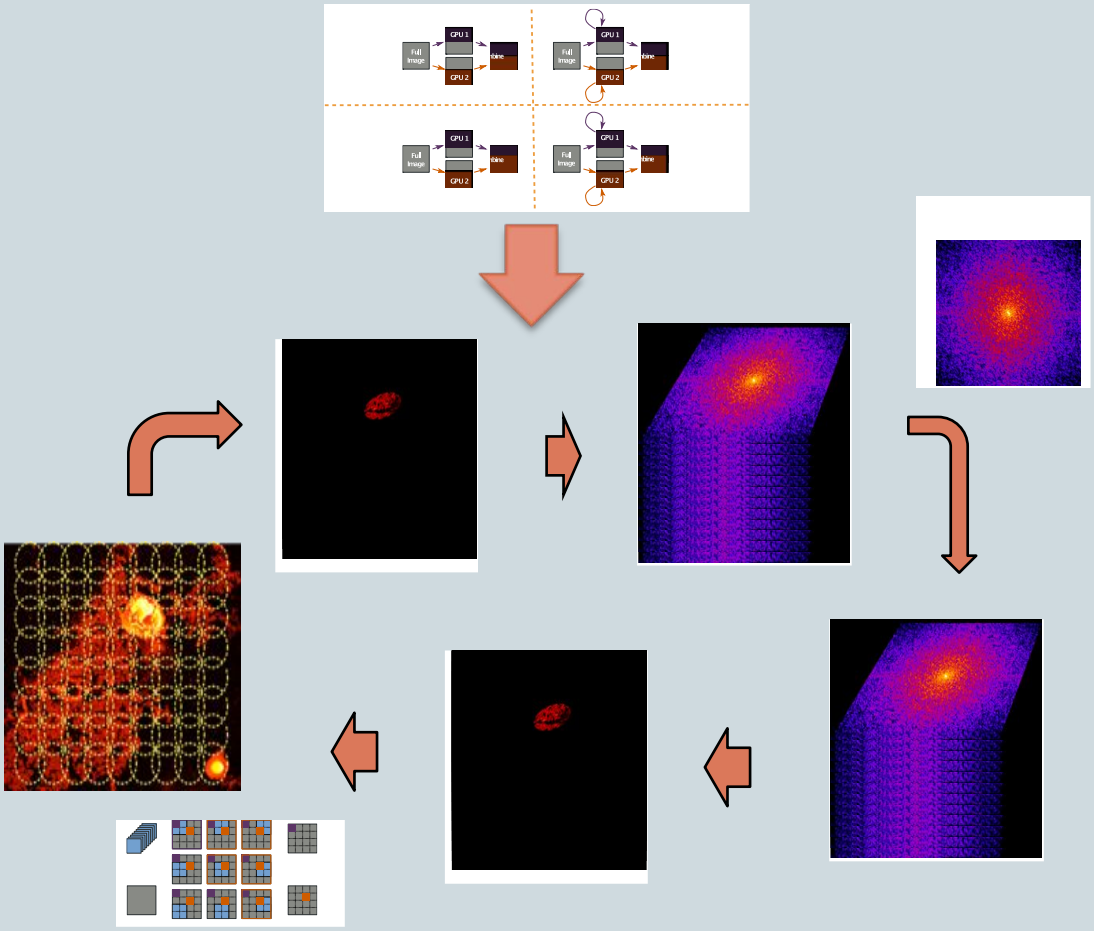


Strategy:
One-Dimensional: Divide image along first dimension.
Linear: Divide frames equally among all processors.
Grid: Divide image based on 2D regions.

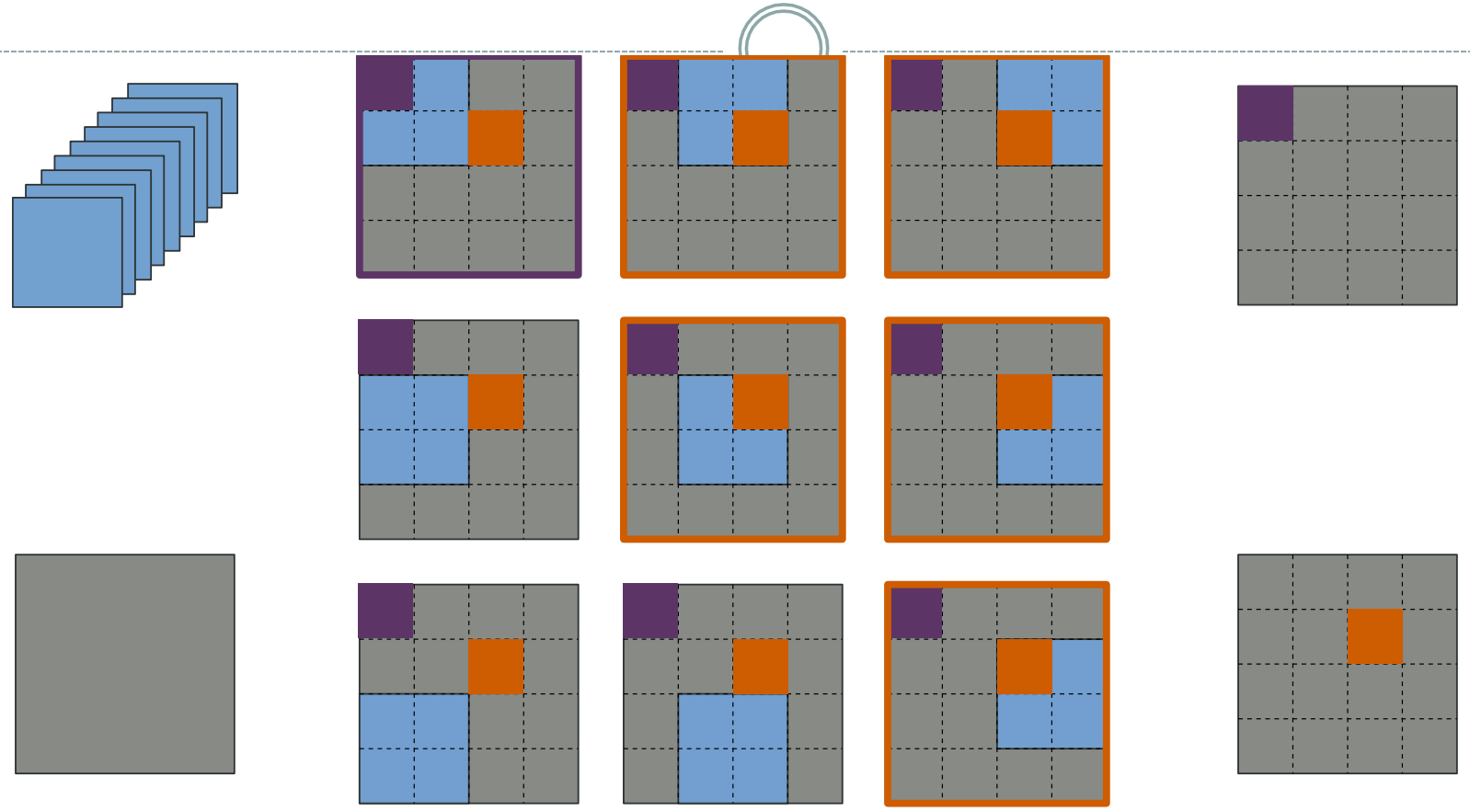
- Image Split
- Overlap
- Illumination Refinement

Thrust (CUDA/OpenMP) Engine

- Initialize: Calculates Overlapping Frames
- Iterate
- Determine Overlap, Split Image
- Apply Modulus Projection and FFT Transformation
- Refine Illumination
- Repeat…



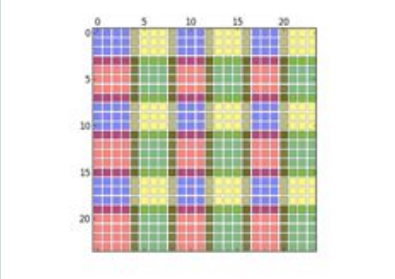
Overlapping Kernel

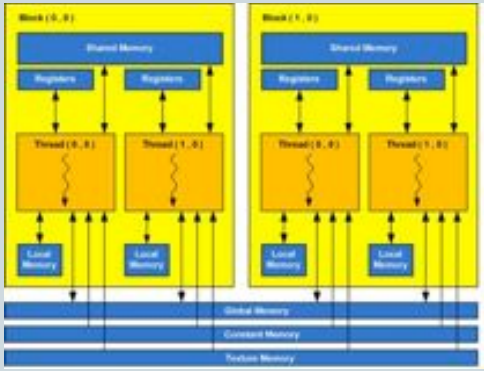


Total of 12 GPU Kernels: Overlap, Splits, Average, Multiply, and Helper Functions.

Performance Considerations

- Parallelization through domain decomposition.
- Use of shared memory.
- Global communication frequency is user controllable.





Strategy extends to out of core and streaming solutions.

Deployment

- SHARP- Stand alone Executable, Available as a Shared Library.
- Python Bindings with NumPy & Mpi4Py support
  - mpirun -n X python main.py <sharp-args> -i 20 -N 2 -R -f output.cxi input.cxi
- PyQt User Interface.
  - Demo Later in the Day.

```
# example execution: mpirun -n X python main.py -i 20 -N 2 NS_140411023.cxi -f output.cxi

import sys
import mpi4py
from mpi4py import *
import sharp

options = sharp.Driver.createOptions()
options.parse_args(sys.argv)

engine = sharp.Driver.createCudaEngine(options.wrapAround)

comm = sharp.Driver.createCommunicator(sys.argv, engine)
input_output = sharp.Driver.createInputOutput(sys.argv, comm)
strategy = sharp.Driver.createStrategy(sharp.Strategy.Grid, comm)
stitcher = sharp.Driver.createStitcher()
#solver = sharp.Driver.createSolver(engine,comm,input_output,strategy,stitcher)

engine.setInputOutput(input_output)

# load meta data
input_output.loadMetadata(options.input_file)

#initialize strategy
strategy.setFrameCorners(input_output.frame_corners())
strategy.setImageSize(input_output.imageSize())
strategy.setFramesSize(input_output.framesSize())
strategy.calculateDecomposition()

input_output.loadMyFrames(options.input_file, strategy.myFrames())

#solver.initialize()

for i in range(options.n_reconstructions):
  output_filename =  "run" + str(i);
```

SHARP-Python Interface

```
# example execution: mpirun -n X python main.py -i 20 -N 2 NS_140411023.cxi -f output.cxi

import sys
import mpi4py
from mpi4py import *
import sharp

options = sharp.Driver.createOptions()
options.parse_args(sys.argv)

engine = sharp.Driver.createCudaEngine(options.wrapAround)

comm = sharp.Driver.createCommunicator(sys.argv, engine)
input_output = sharp.Driver.createInputOutput(sys.argv, comm)
strategy = sharp.Driver.createStrategy(sharp.Strategy.Grid, comm)
stitcher = sharp.Driver.createStitcher()
#solver = sharp.Driver.createSolver(engine,comm,input_output,strategy,stitcher)

engine.setInputOutput(input_output)

# load meta data
input_output.loadMetadata(options.input_file)

#initialize strategy
strategy.setFrameCorners(input_output.frame_corners())
strategy.setImageSize(input_output.imageSize())
strategy.setFramesSize(input_output.framesSize())
strategy.calculateDecomposition()

input_output.loadMyFrames(options.input_file, strategy.myFrames())

#solver.initialize()

for i in range(options.n_reconstructions):
  output_filename =  "run" + str(i);
```

Control Options

Core Options:

- -i: number of iterations to run. Defaults to 10.
- -o: output period of the error metrics in iterations.
- -r: illumination refinement period in iterations. Defaults to off.
- -N: number of independent reconstructions to do.
- -f: output filename (if not given the output is written as a new image in the input file)

Advanced Options:

- -b: beta parameter for RAAR.
- -g: how often to go global synchronizations. Defaults to 1.
- -n: noise parameter for the illumination cutoff. Defaults to 1e-2.
- -R: round pixel step up.
- -s: silence all output.
- -B: relaxed fourier projection.
- -T: period of background retrieval.
- -I: enforce intensities when refining illumination.
- -M: enforce fourier mask when refining illumination.

Debugging Options: -D (output debug messages), -t (output the time the solver takes), -w: turn on frame wrap around.

Source Code provides –DTIMINGS flag to provide aggregate total times of every kernel & major algorithm.

Thanks!

Core Algorithm: Data Movement

- Communicator
  - Communicates images as well as intermediate frames and frame corners.
  - Provides Sum, Min, Max, and handles standard and complex data.

Core Algorithm: Image Reconstruction (Part 1)

- Strategy
  - OneDimensional: Divide image along first dimension. Each node gets frames wherever the center falls inside the slice.
  - Linear: Divide frames equally among all processors.
  - Grid: Divide image based on 2D regions
- Stitcher (remove…)
  - Stitch Frames: Accumulate reconstructed image.
  - Stitch Image: Stitch together an image with maximum magnitudes at each pixel.
- Solver
  - Set up all parameters and iterate over

Core Algorithm: Image Reconstruction (Part 2)

Engine – Thrust (Cuda/OpenMP backend)

- For Each Reconstruction
  - Split Frames, Iterate to refine, Merge Frames.
- Each iteration (pseudo code)
  - Overlap Projector – description
  - Module Projector – description
  - Perform reconstruction.
  - Refine Illumination.
  - Re-scale and Re-add
  - Compute Error
  - Write to Disk

Parallelism

- MPI Communication.
  - Communication pattern…
- GPU Kernels.
  - High level description of kernels…

Performance & Timing

- Performance numbers
  - TODO:
- -DTIMINGS
  - Timings can be enabled for all kernels and major operations.