X-RAY PTYCHOGRAPHY RECONSTRUCTION ON DISTRIBUTED GPUS

YAO-LUNG LEO FANG AND MEIFENG LIN

NEW YORK SCIENTIFIC DATA SUMMIT

AUGUST 7, 2018

Zhihua Dong
CSI

Xiaojing Huang
NSLS-II

Leo Fang
CSI/NSLS-II

Hanfei Yan
NSLS-II

Sungsoo Ha
CSI

Wei Xu
CSI

Meifeng Lin
CSI

BROOKHAVEN NATIONAL LABORATORY

OUTLINE

- Introduction
  - X-ray Ptychography
  - Data and Computation Challenges
- GPU Acceleration
  - Implementation Workflow
  - Results
- Conclusions

INTRODUCTION

MOTIVATION

- X-ray diffraction technique is an essential tool for studying the structure of materials at nanoscale.
- With brighter light sources, faster data acquisition rates, larger data volumes (higher image resolutions), high performance computing (HPC) is critical for enabling *in-situ* analysis of image data from light sources.
- Efficient interactive graphical user interface (GUI) is also necessary to allow general beam line users to analyze their data in near real time, so that they can adjust their X-ray experiments as needed.

PTYCHOGRAPHY

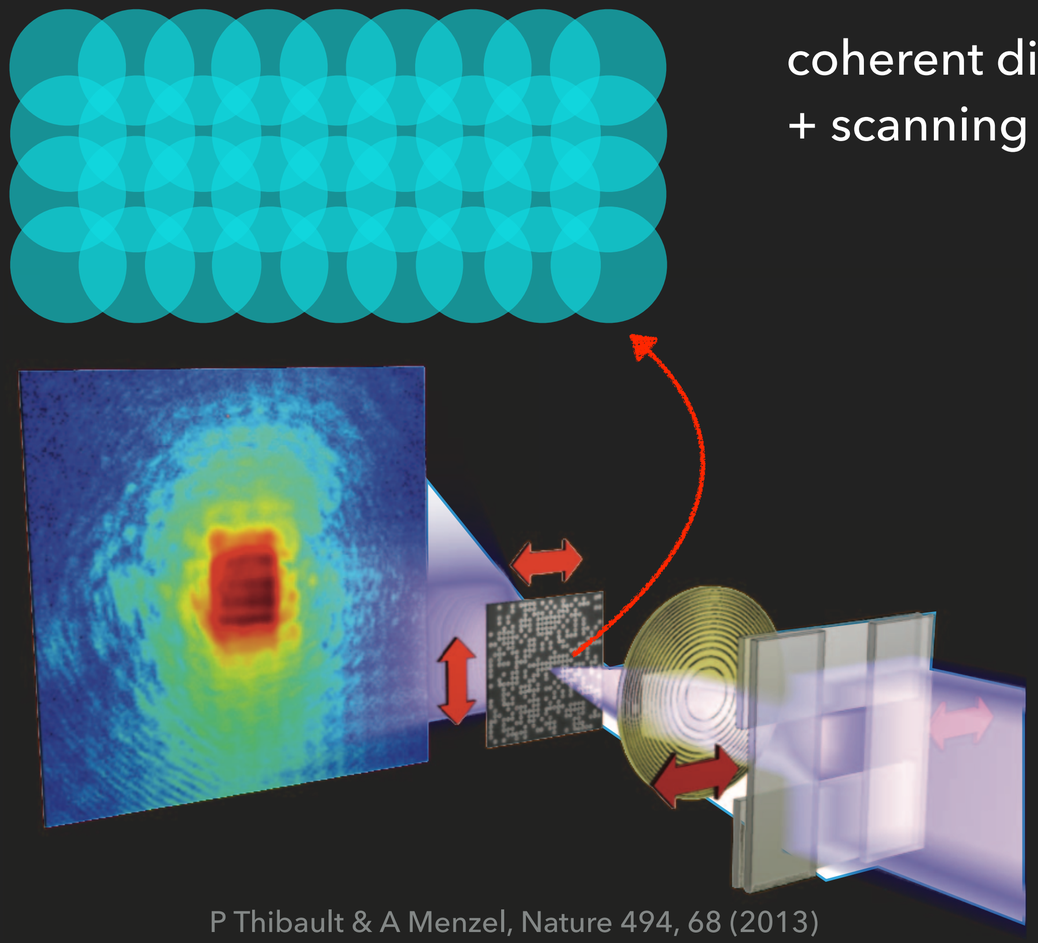
Ptychography

From Wikipedia, the free encyclopedia

Ptychography (/tɪˈtʃoʊgræfi/ ti-CHOH-graf-ee; πτυχο-γραφία, from πτυχή = fold + γραφή = writing/scripture), also known as scanning diffraction microscopy, is a type of coherent diffraction imaging invented by Walter Hoppe in the 1960s[1] that uses a beam of coherent radiation (e.g. x-rays) or particles (e.g. electrons) to illuminate a small sample of material and then the phase information from the beam is recovered by analysing the diffraction pattern produced by the scattered radiation.[2] The resulting microscopic image can show details at the atomic level.

- Ptychography, pronounced as ti-CHOH-graf-ee, from Greek for “fold”.
- A coherent diffraction imaging technique invented by Walter Hoppe in 1960s.
- Coherent diffraction images are obtained from overlapping regions of the sample.
- Sample phase information can be reconstructed from post analysis of the scanning images.

X-RAY PTYCHOGRAPHY



P Thibault & A Menzel, Nature 494, 68 (2013)

reconstruct high-resolution images from coherent diffraction data

coherent diffraction imaging (CDI)
+ scanning transmission X-ray microscopy (STXM)

✓no special optics requirement

✓wavelength-limited resolution

✓no *a priori* knowledge in illumination

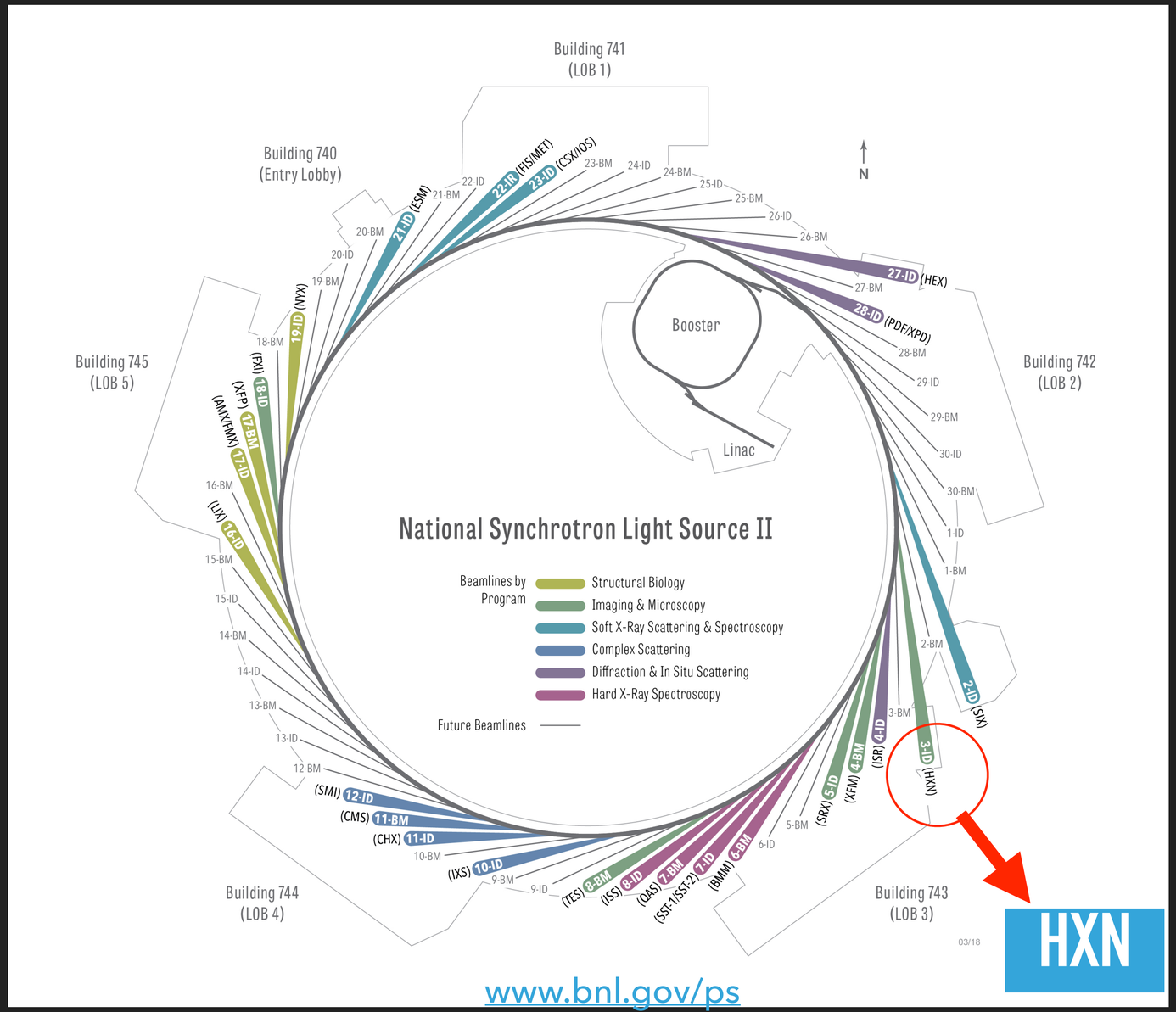
✓easy sample preparation

- diffraction intensity recorded
- overlapped scanning

➡ sufficient information for object reconstruction

NSLS–II AND THE HXN BEAM LINE

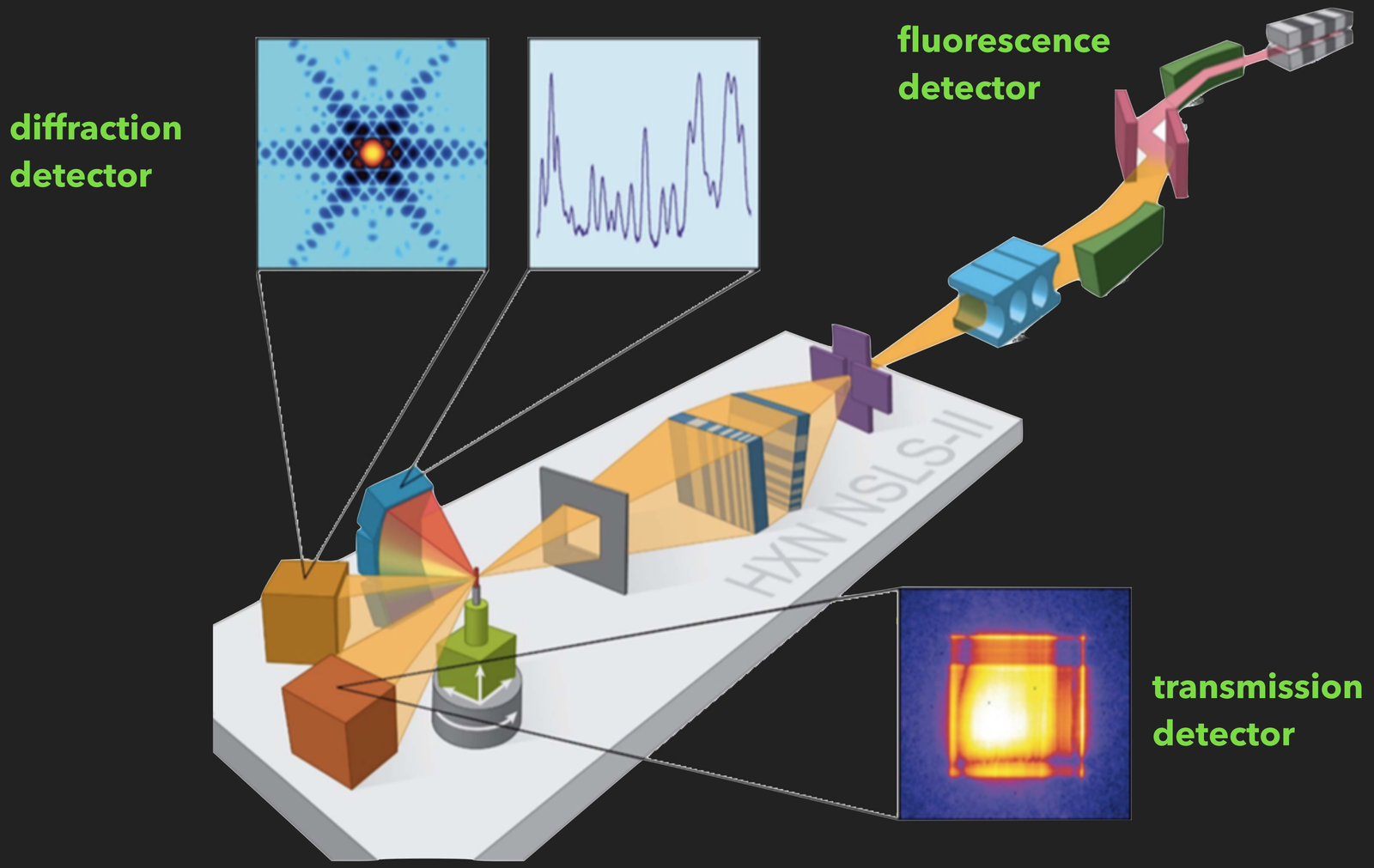
- State-of-the-art, medium-energy (3-billion-electron-volt, or GeV) electron storage ring that produces x-rays up to 10,000 times brighter than the NSLS
- 28 beam lines in operation; 1 under development
- HXN - Hard X-ray Nanoprobe: hard x-ray imaging of structure, elements, strain and chemical states with spatial resolution from 10 to 30 nm with an ultimate goal of ~1nm



www.bnl.gov/ps

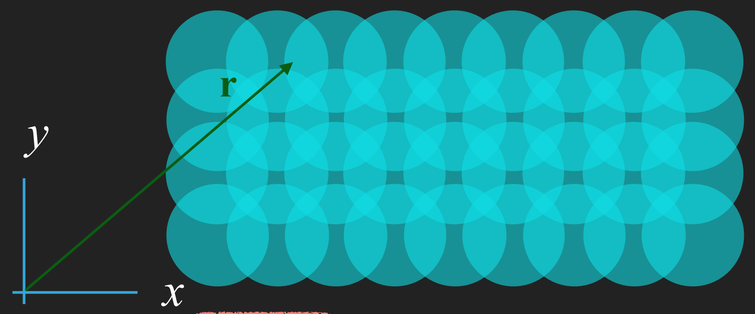
BROOKHAVEN NATIONAL LABORATORY

HXN BEAMLINE SETUP



Yan et al., Nano Futures 2, 011001 (2018)

SINGLE-MODE PTYCHO. RECONSTRUCTION



complex exit wave

"probe" profile

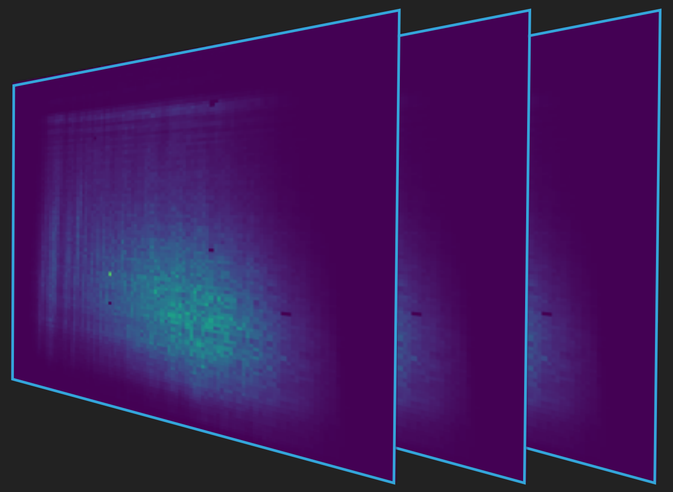
$$\psi_j(\mathbf{r}) = P(\mathbf{r} - \mathbf{r}_j)O(\mathbf{r})$$

"object" transmission function

Input $I_j(\mathbf{q}) = |\mathcal{F}[\psi_j(\mathbf{r})]|^2$

$j = 1, 2, \cdots, N$

constraints



difference map algorithm*

Output

$$P(\mathbf{r}) = \frac{\sum_j [O(\mathbf{r}+\mathbf{r}_j)]^* \psi_j(\mathbf{r}+\mathbf{r}_j)}{\sum_j |O(\mathbf{r}+\mathbf{r}_j)|^2}$$

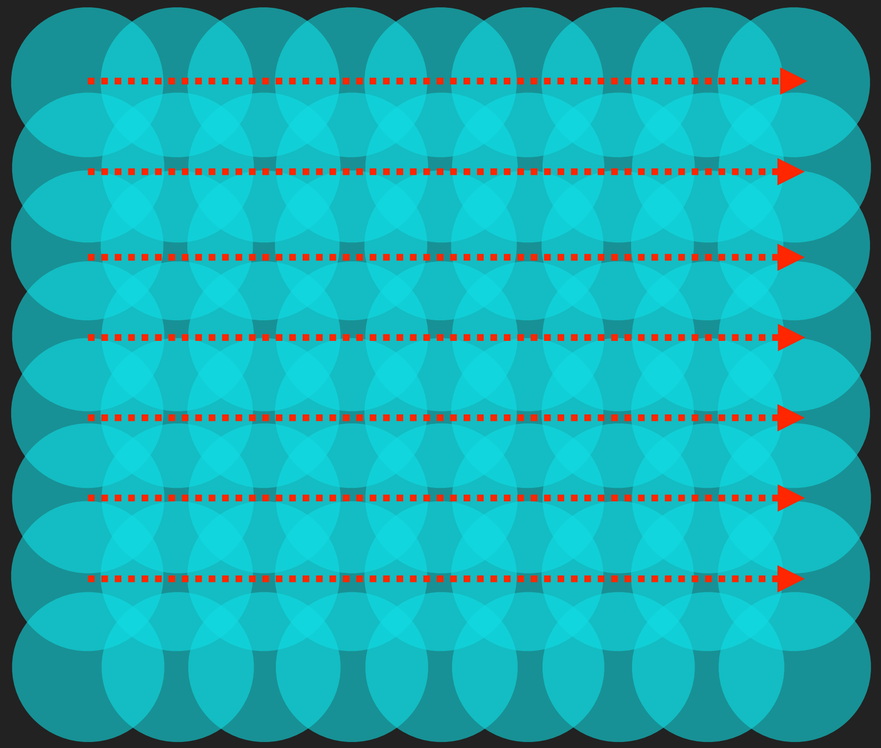
$$O(\mathbf{r}) = \frac{\sum_j [P(\mathbf{r}-\mathbf{r}_j)]^* \psi_j(\mathbf{r})}{\sum_j |P(\mathbf{r}-\mathbf{r}_j)|^2}$$

MULTI-MODE PTYCHO. RECONSTRUCTION

motivation: to reduce motor overhead by performing “fly scan”

➡blurriness due to motion can be regarded as having incoherent x-ray beam

➡need to compensate this with multiple probe modes



probe mode index

object mode index

$$\psi_j^{(k,l)}(\mathbf{r}) = P^{(k)}(\mathbf{r}-\mathbf{r}_j)O^{(l)}(\mathbf{r})$$

$$P^{(k)}(\mathbf{r}) = \frac{\sum_l \sum_j \left[O^{(l)}(\mathbf{r}+\mathbf{r}_j)\right]^* \psi_j^{(k,l)}(\mathbf{r}+\mathbf{r}_j)}{\sum_l \sum_j |O^{(l)}(\mathbf{r}+\mathbf{r}_j)|^2}$$

$$O^{(l)}(\mathbf{r}) = \frac{\sum_k \sum_j \left[P^{(k)}(\mathbf{r}-\mathbf{r}_j)\right]^* \psi_j^{(k,l)}(\mathbf{r})}{\sum_k \sum_j |P^{(k)}(\mathbf{r}-\mathbf{r}_j)|^2}$$

one more sum to do ⇨ still highly parallelizable!

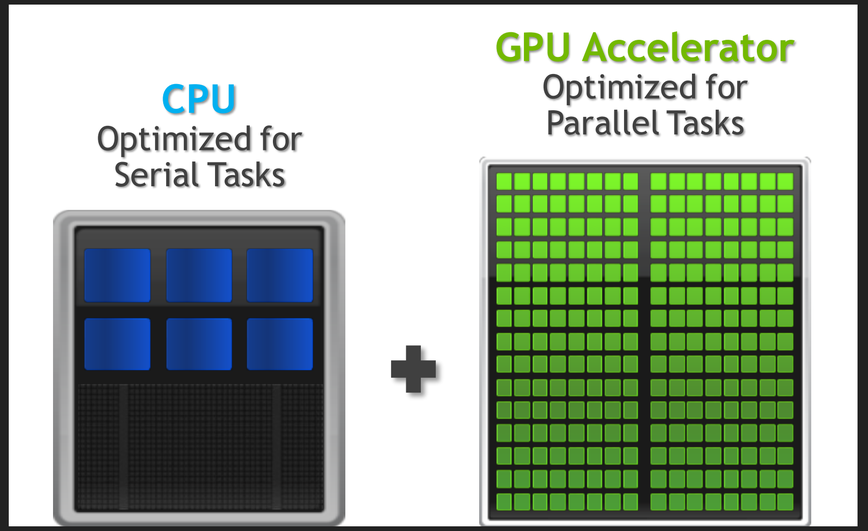
DATA AND COMPUTATION CHALLENGES

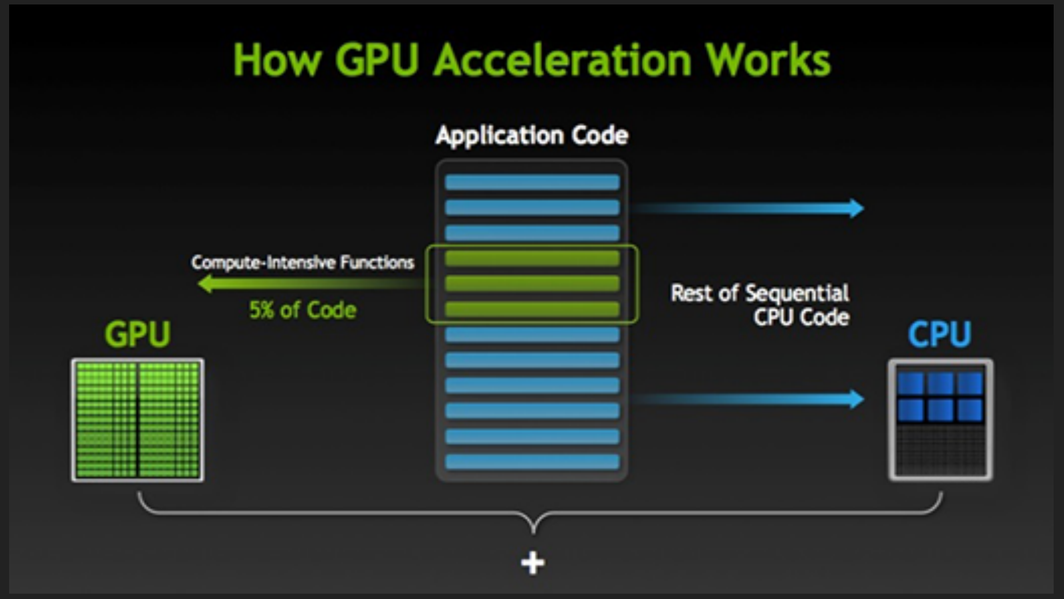
- For a high-resolution ptychography reconstruction
  - Need O(10,000) scan images
  - ~200x200 pixels in double precision per image
  - Memory requirement for input images: O(1 GB) to O(10 GB)
- DM algorithm: O(100) iterations
- Memory requirement for the DM algorithm (including temporary buffers):
    - Single-mode: ~4x input
    - Multi-mode: ~10x input
- Serial code: takes *hours* to complete one ptychography reconstruction.
- Beamline users have limited access time and possibly limited offline computing resources => Fast *in-situ* ptychography reconstruction software will improve beam line users' productivity.

GPU ACCELERATION

GPUS FOR SCIENTIFIC COMPUTING

- Modern GPUs offer massive compute power for parallel tasks
  - Thousands of compute cores vs. O(10) cores on CPUs
- Image processing: straightforward parallelism in pixels.
- GPU memory still limited: 32 GB in the latest generation of NVIDIA GPUs (Volta V100).
- Need to distribute workload to multiple GPUs





Images from nvidia.com

DATA MANAGEMENT FOR GPU COMPUTING

- Host CPUs and the GPU devices have distinct physical memories
- Data transfer between the CPU memory and the device memory could be a bottleneck.
- Want to minimize data transfer while maximizing the computation done on device.
- Difference map algorithm involves mostly local operations. Global sum (*Allreduce*) needed only at the end of each iteration. Scalable!

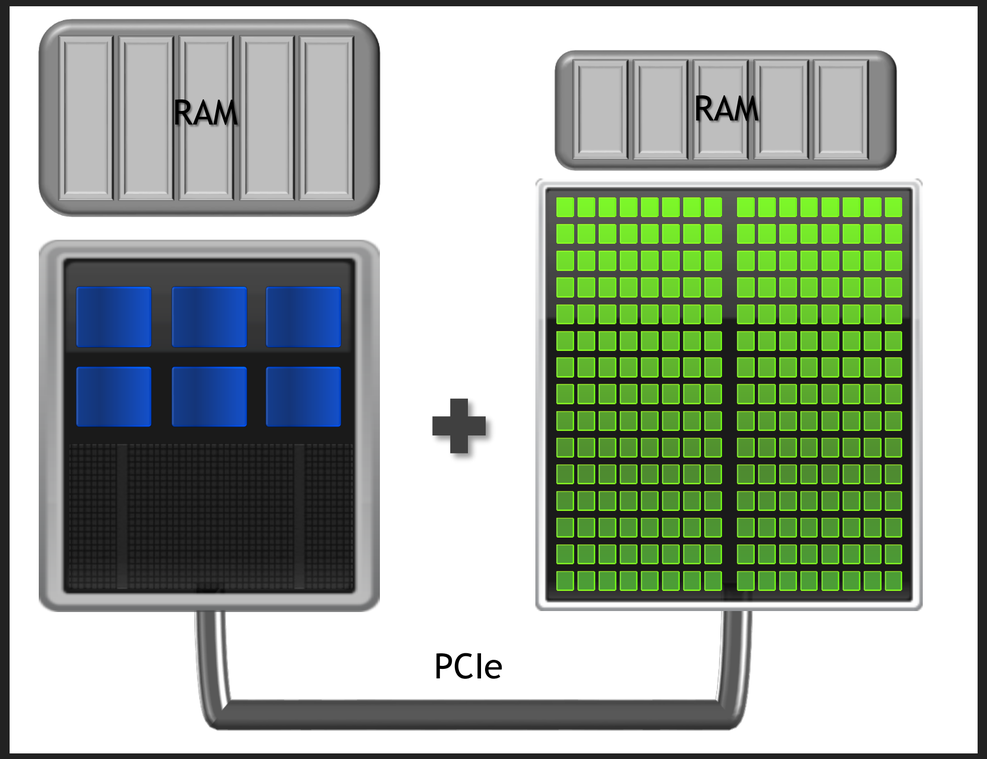
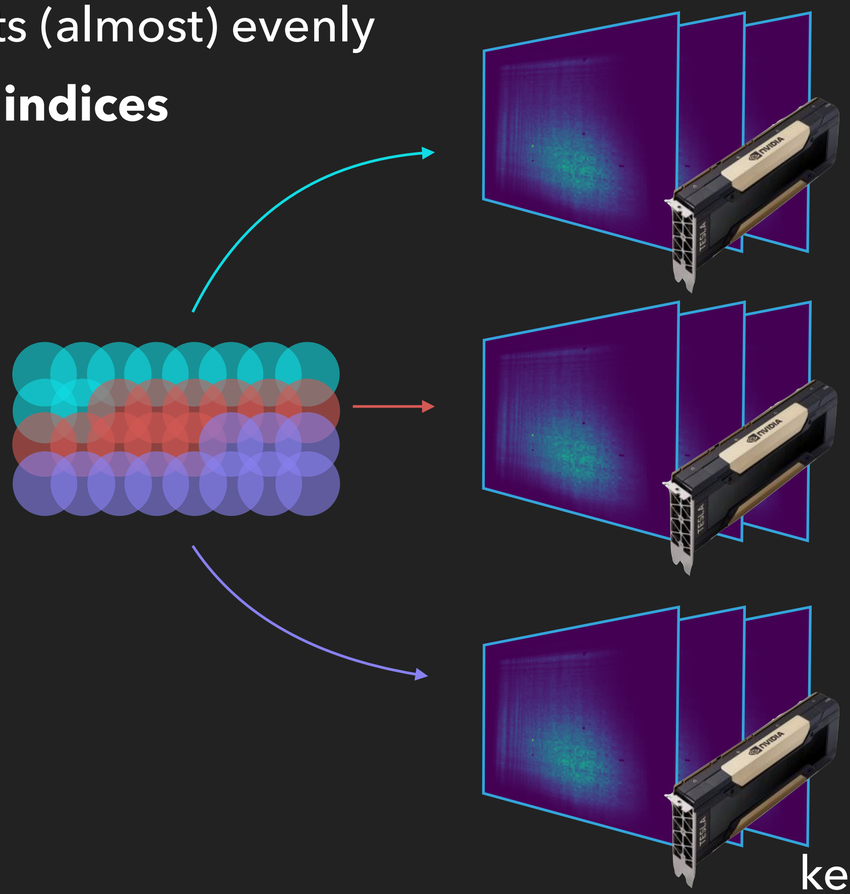


Image from nvidia.com

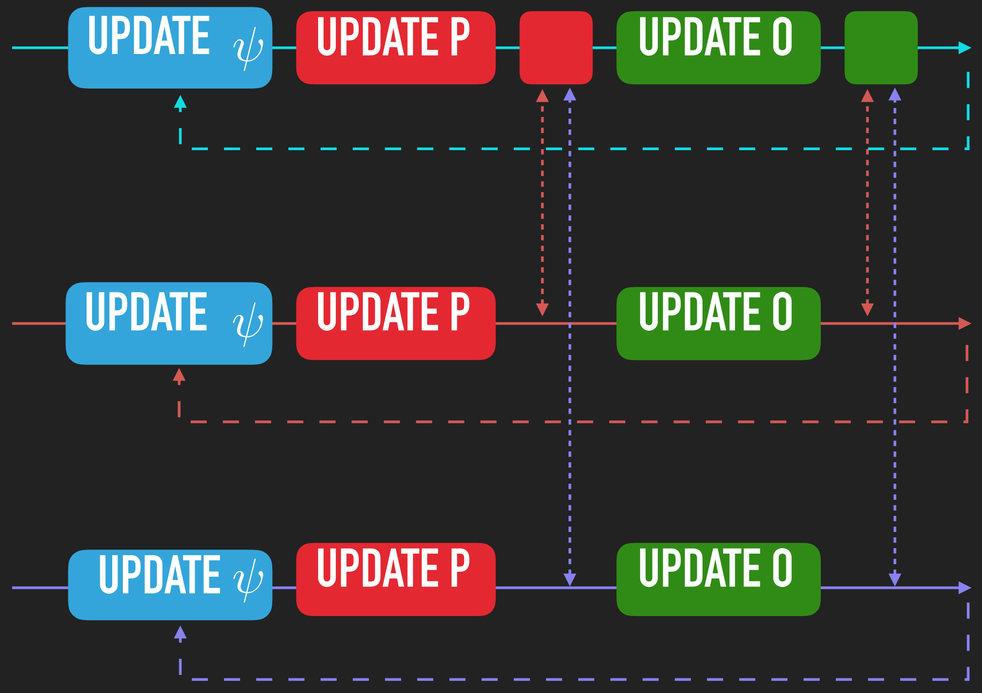
WORKFLOW FOR PTYCHO MULTI-GPU ACCELERATION

split points (almost) evenly

based on **indices**



iteration loop

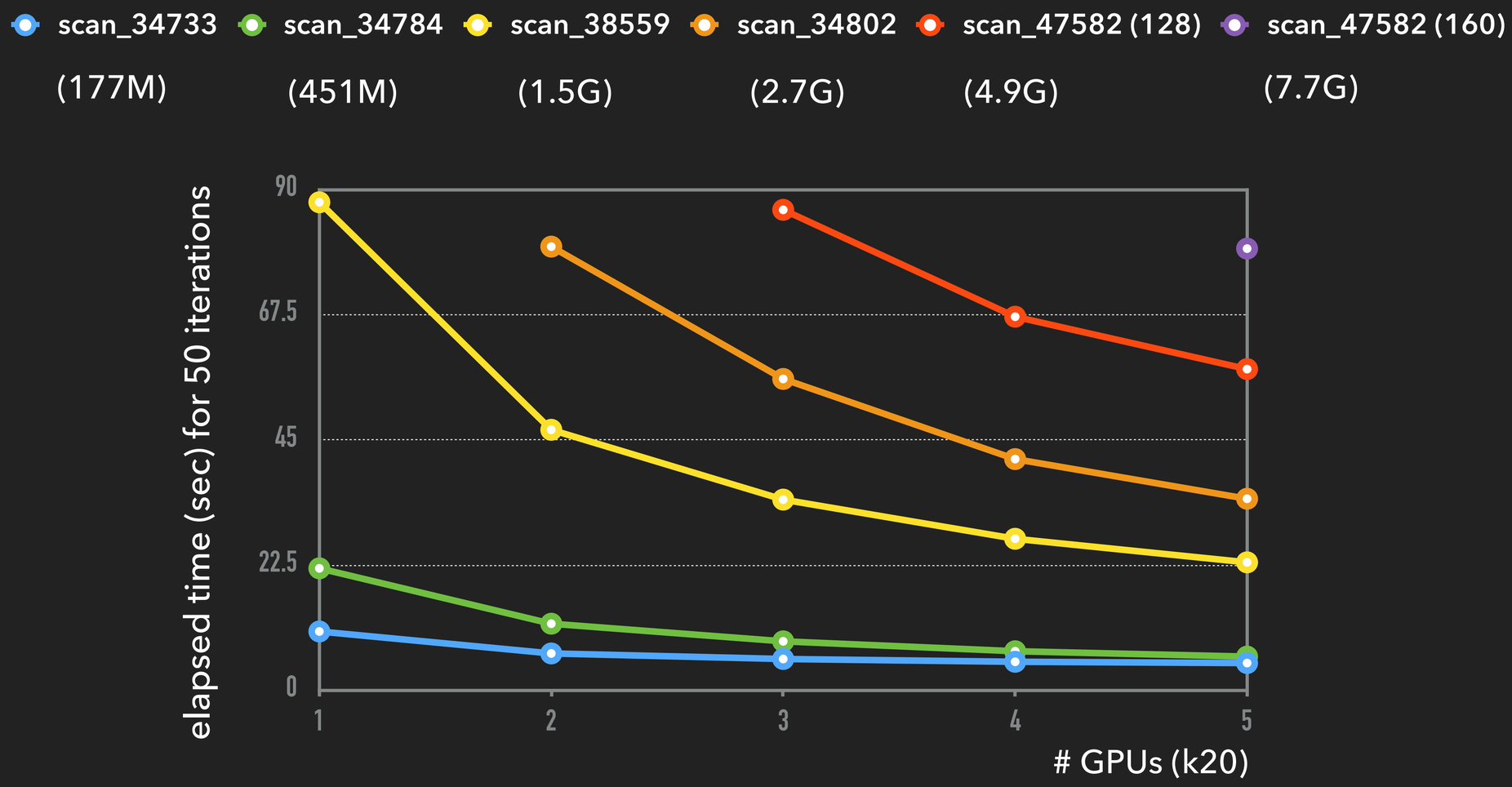


keep everything on device, only MPI_Allreduce P & O

GPU-ACCELERATED PTYCHOGRAPHY SOFTWARE

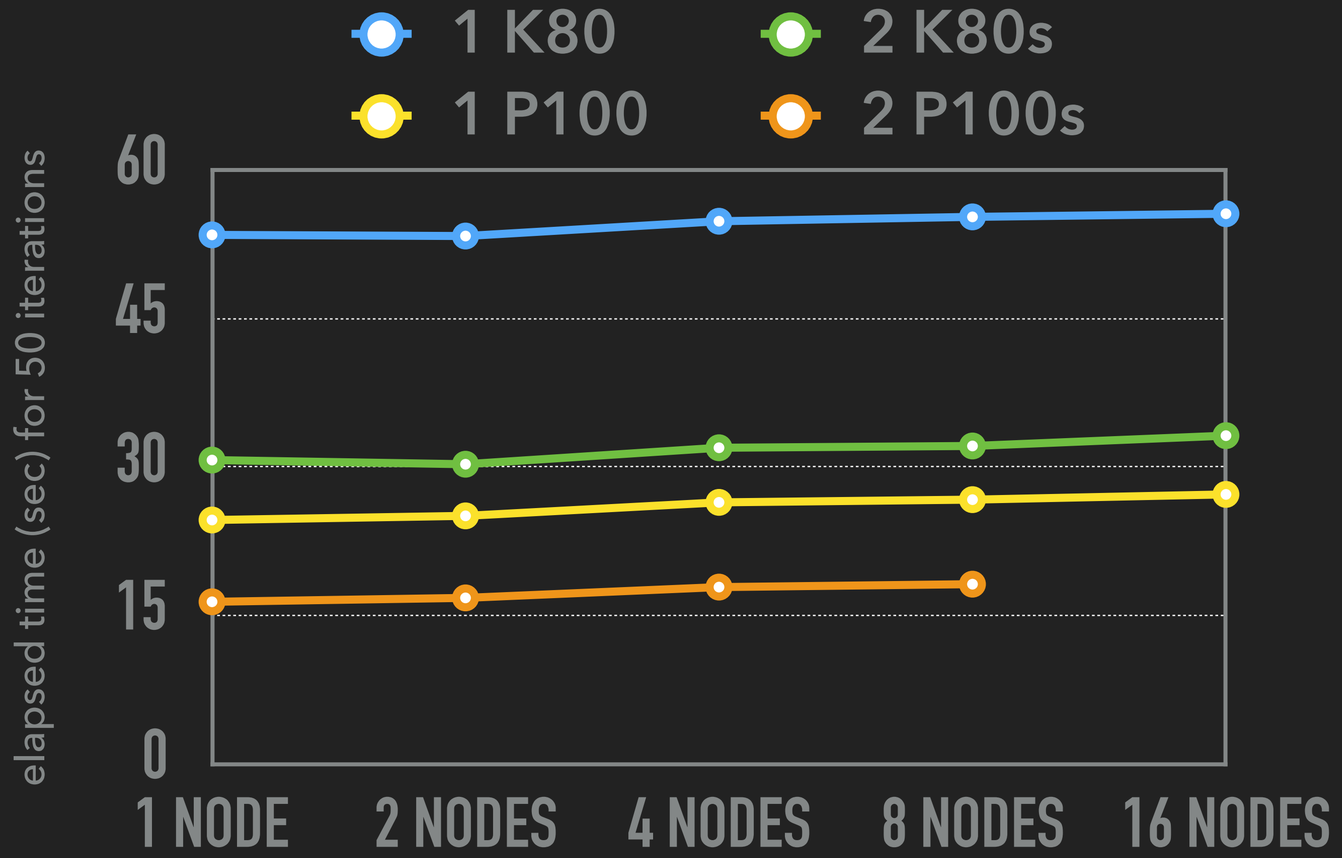
- Originally written in Python (numpy + scipy + …)
- GPU version uses PyCUDA, scikit-cuda, mpi4py
- Computational intensive functions rewritten in CUDA C
- Tremendous speedup: hours => *sub-minutes!*
- Everything in Python (except for ~600 lines of CUDA C) => integration with NSLS-II data acquisition and analysis environment (databroker)…
- Support for distributed GPUs, including GPU clusters and across different servers.
- Graphical user interface (GUI) integrated with multi-GPU backend.

RESULTS: STRONG SCALING



Tested @ hpc1

RESULTS: WEAK SCALING



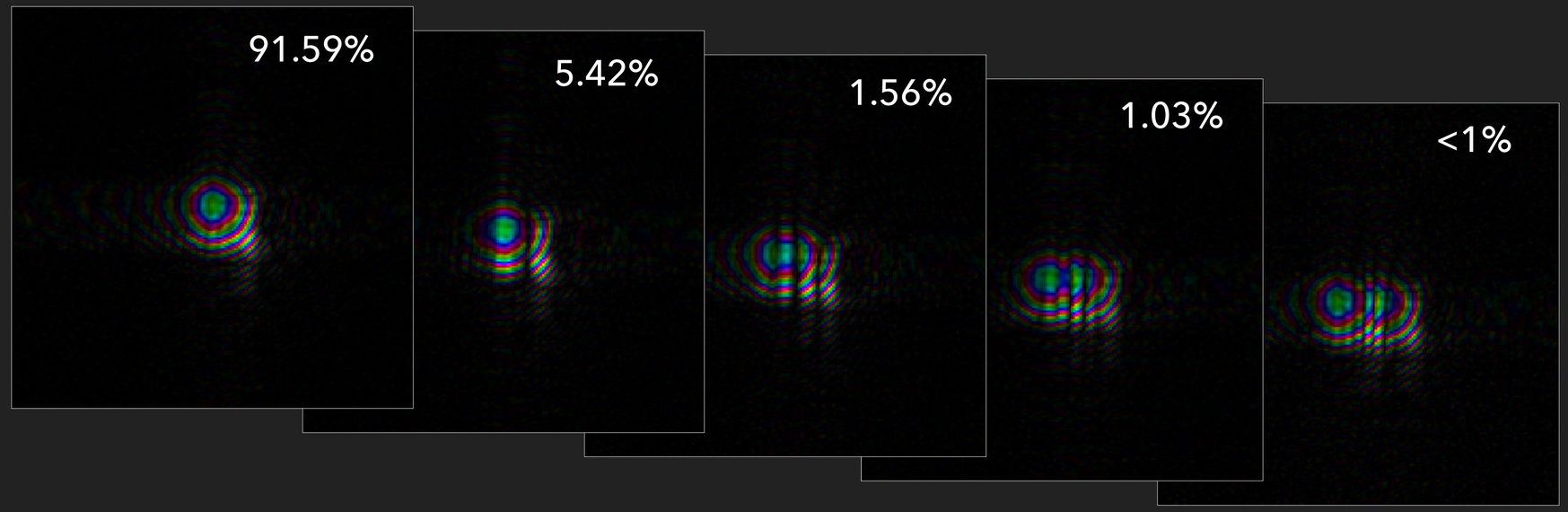
synthesized data used, each node gets ~5000 points

Tested @ BNL Institutional Cluster

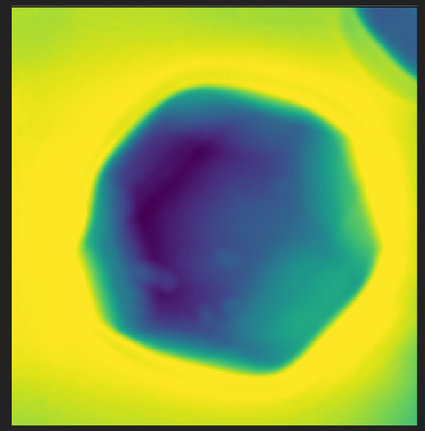
RESULTS: PERFORMANCE IMPROVEMENTS

Test machine: xf03id-srv5@HXN, Intel Xeon CPU E5-2630 v4 @2.20GHz, 256GB RAM, 4 NVIDIA Tesla V100 GPUs. 50 iterations used.

showcase: gold nano-crystal with multi-mode



obj. phase



CPU (serial): 8.8 hr → 4 V100 GPUs: 33.78s

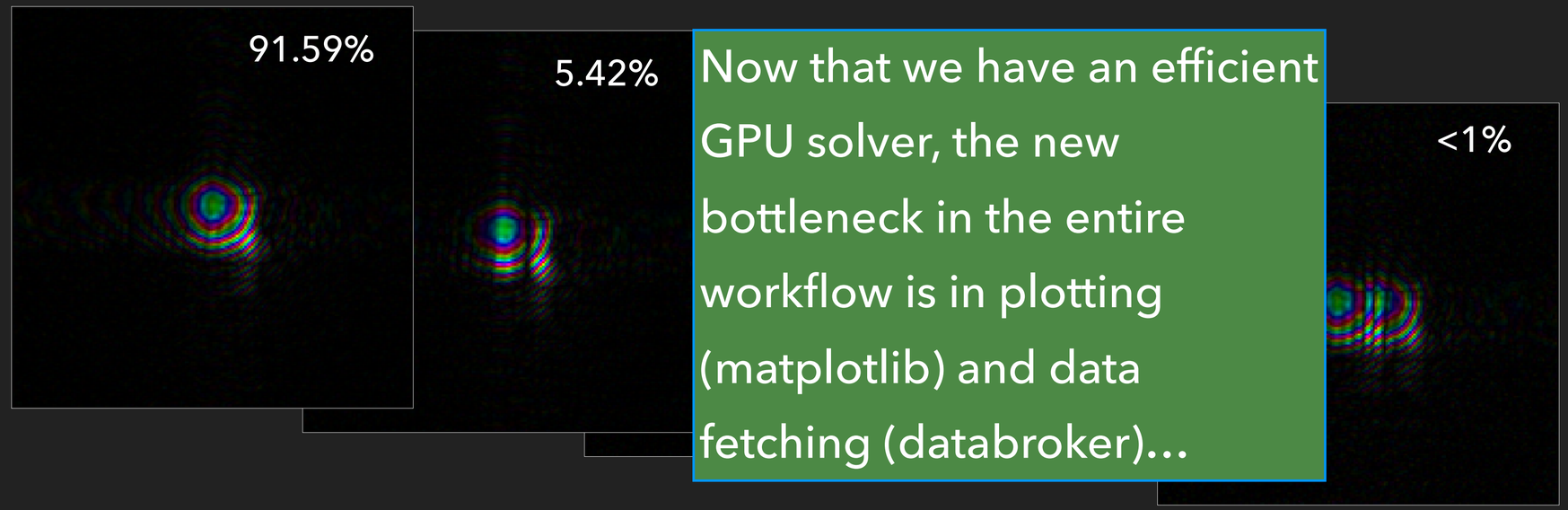
939x speedup!

New: MPI-enabled parallel CPU version also reduces the CPU time significantly 10X to 85X with 32 processes *depending on data sizes*

RESULTS: PERFORMANCE IMPROVEMENTS

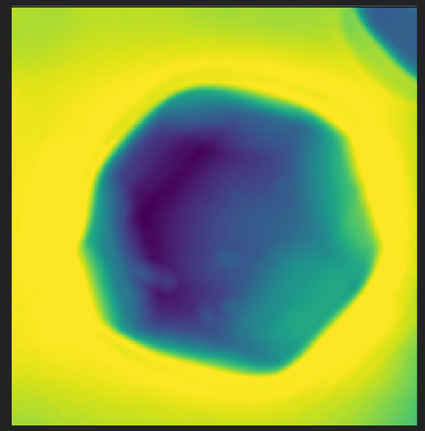
Test machine: xf03id-srv5@HXN, Intel Xeon CPU E5-2630 v4 @2.20GHz, 256GB RAM, 4 NVIDIA Tesla V100 GPUs. 50 iterations used.

showcase: gold nano-crystal with multi-mode



Now that we have an efficient GPU solver, the new bottleneck in the entire workflow is in plotting (matplotlib) and data fetching (databroker)...

obj. phase



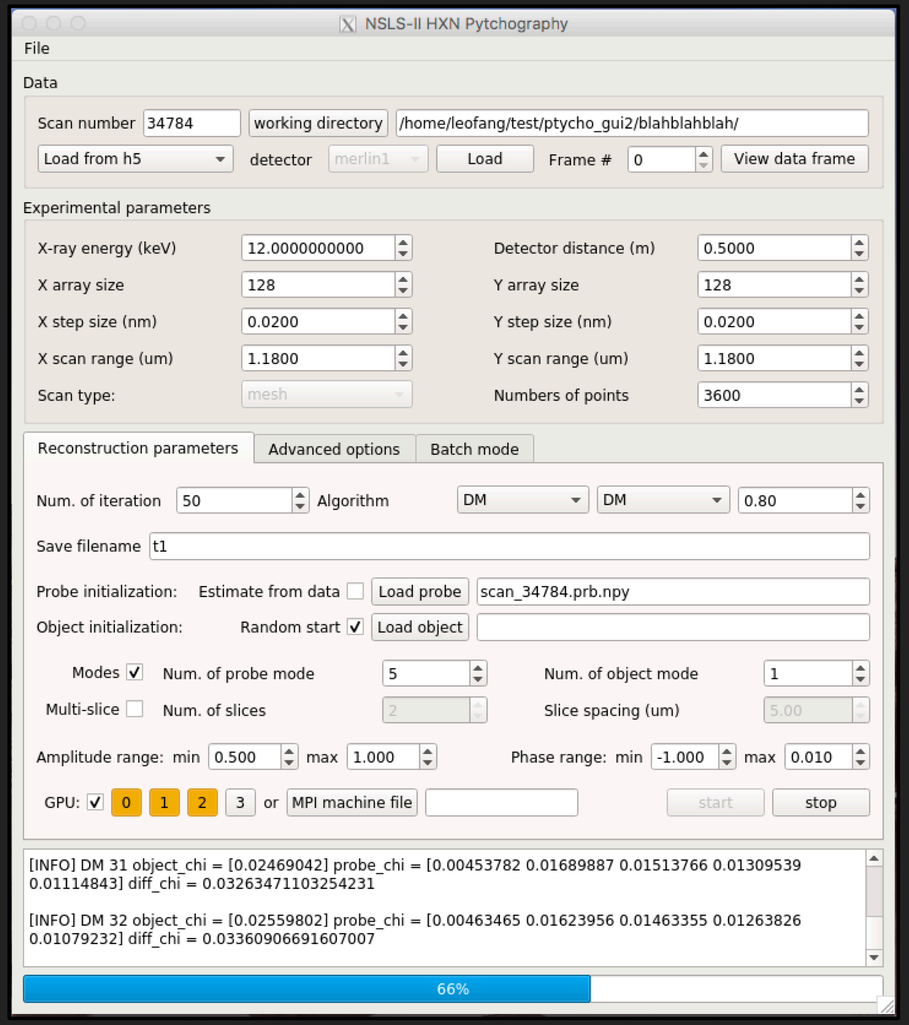
939x

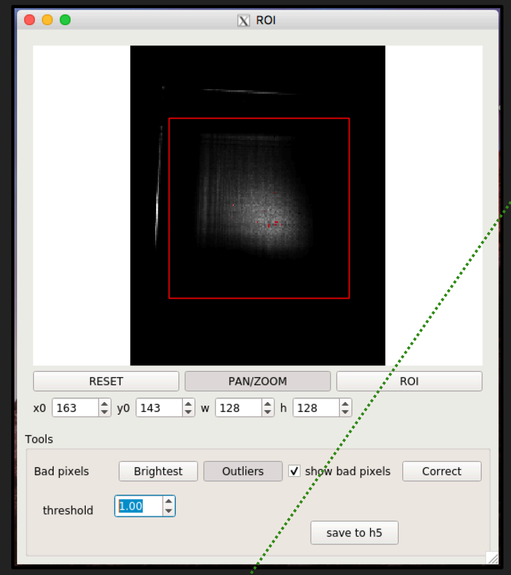
CPU (serial): 8.8 hr ⟶ 4 V100 GPUs: 33.78s

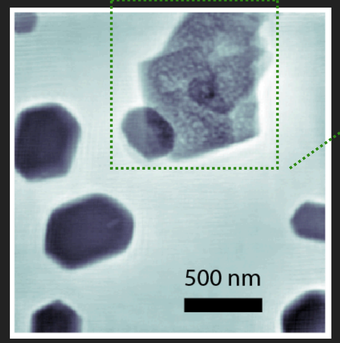
speedup!

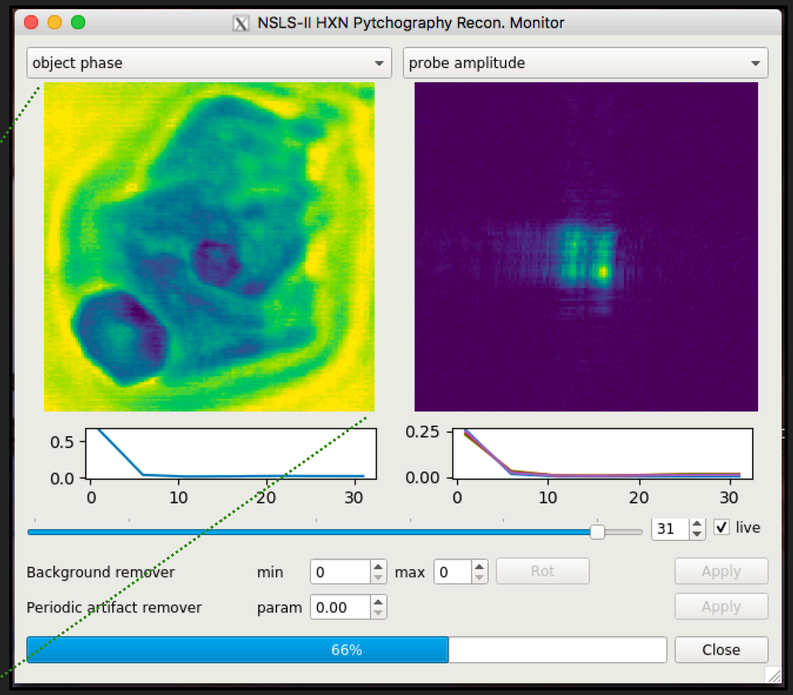
New: MPI-enabled parallel CPU version also reduces the CPU time significantly 10X to 85X with 32 processes *depending on data sizes*

GRAPHICAL USER INTERFACE









- ★ PyQt5
- ★ Customized event handler
- ★ In-situ processing of raw data
- ★ Efficient realtime monitor
- ★ Clean separation of UI logic, implementation & computation

CONCLUSIONS

CONCLUSIONS

- Fast, *in-situ*, ptychography reconstruction software is essential for beam line users' productivity.
- We have successfully implemented a GPU-accelerated version of the HXN ptychography reconstruction software with great improvement in performance compared to the serial CPU version.

FUTURE WORK

- Accelerating other solvers (maximum likelihood/error reduction/multi-slice/etc) - ongoing
- Further performance fine-tuning (single-precision floating point arithmetics, pinned memory, batch size, CUDA-aware MPI, etc) - ongoing
- Integrating post-processing utilities with GUI
- Optimizing object-update routine
- Modularizing for coupling to SHARP
- Improving quality of amplitude reconstruction

ACKNOWLEDGEMENTS

- This work is supported by BNL's LDRD program under award #17-029.
- Part of the tests used computing resources of the Center for Functional Nanomaterials, a US DOE Office of Science user facility at BNL.

THANK YOU!