Real-Time Data Analysis and Autonomous Steering of Synchrotron Light Source Experiments

Tekin Bicer*†, Doga Gursoy†, Rajkumar Kettimuthu*‡, Ian T. Foster*‡§,
Bin Ren¶, Vincent De Andrede† and Francesco De Carlo†
*Mathematics and Computer Science Division Argonne National Laboratory, Lemont, IL 60439
Email: {bicer, kettimut, foster}@anl.gov
†X-Ray Science Division, Advanced Photon Source, Argonne National Laboratory Lemont, IL 60439
Email: {dgursoy, vdeandrade, decarlo}@aps.anl.gov
‡Computation Institute, University of Chicago and Argonne National Laboratory, Chicago, IL 60637
§Department of Computer Science, University of Chicago, Chicago, IL 60637
¶Computer Science Department, College of William & Mary Williamsburg, VA 23185
E-mail: bren@cs.wm.edu

***Abstract*—Modern scientific instruments, such as detectors at synchrotron light sources, can generate data at 10s of GB/sec. Current experimental protocols typically process and validate data only after an experiment has completed, which can lead to undetected errors and prevents online steering. Real-time data analysis can enable both detection of, and recovery from, errors, and optimization of data acquisition. We thus propose an autonomous stream processing system that allows data streamed from beamline computers to be processed in real time on a remote supercomputer, with a control feed-back loop used to make decisions during experimentation. We evaluate our system using two iterative tomographic reconstruction algorithms and varying data generation rates. These experiments are performed in a real-world environment in which data are streamed from a light source to a cluster for analysis and experimental control. We demonstrate that our system can sustain analysis rates of hundreds of projections per second by using up to 1,200 cores, while meeting stringent data quality constraints.**

I. Introduction

X-ray photon sources are crucial tools for addressing grand challenge problems in the life sciences, energy, climate change, and information technology [1, 2]. Beamlines at such facilities, which are located in many countries worldwide, use various methods to collect data as samples are illuminated with an intense x-ray beam. Increasing the productivity of these instruments and improving the quality of science being done at these facilities will have far-reaching benefits.

Scientists who run experiments at beamlines want to collect the maximum information possible about the material under study, in as little time as possible. However, data analysis typically occurs only after acquisition is complete, meaning that experiments are run "blind." Thus, scientists do not know whether they have modified the specimen by beam damage or spent considerable time scanning insignificant areas while collecting insufficient detail from crucial features or at a critical time point in a dynamic specimen. The ability to analyze data produced by detectors in near-real time could enable optimized data collection schemes, on-the-fly adjustments to experimental parameters, early detection of and response to errors (saving both beam time and scientist time), and ultimately improved productivity.

Real-time analysis is challenging because of the vast amounts of generated data that must be analyzed in extremely short amounts of time. Detector technology is progressing at unprecedented rates (far greater than Moore's law), and modern detectors can generate data at rates of multiple gigabytes per second. For example, a tomography experiment at the Advanced Photon Source (APS) at Argonne National Laboratory can generate data from 8 MB/sec (e.g., imaging of cement hardening, 1 projection/sec for two days) to 16 GB/sec (fast imaging up to 2,000 projections/sec for short time intervals), where experiments with 100–200 projections/sec are common. The computational power required to extract useful information from these streaming datasets in *real time* almost always exceeds the resources available at a beamline. Thus the use of remote HPC facilities for analysis is no longer a luxury but a necessity.

A number of challenges arise in using remote HPC facilities for real-time analysis, including on-demand acquisition of compute and network resources, efficient data streaming from beamline to HPC facility, and analysis methods that can keep up with the data generation rates so that timely decisions can be taken. In this work, we focus on the last challenge because we believe that this is a crucial gap in current knowledge.

The primary contributions of this work are threefold. (1) We present an innovative distributed stream processing system for light source data analysis. Our system considers all three stages that are required for analyzing and steering light source experiments: data acquisition, real-time data analysis, and experiment control. We implement two iterative algorithms for tomographic reconstruction, a widely used data analysis application at synchrotron light sources, and scale their execution in a streaming setting. (2) We demonstrate experimental steering for light source experiments by using a control-feedback loop, with a controller that analyzes reconstructed images and applies an image quality metric to determine when to finalize an experiment. (3) We extensively evaluate our system

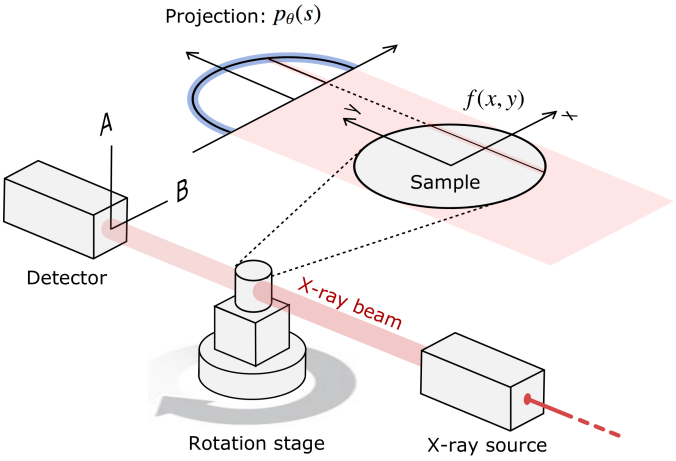


Fig. 1: Basic layout of a tomography setup and demonstration of the measurement process. The sample is placed on a rotary stage and is illuminated by an x-ray beam (shown in pink) as the stage is rotated in uniform increments. The transmitted photons are collected by using a detector, resulting in x-ray projections. The cross section of the sample is highlighted to show how a projection (shown in blue) is formed.

in a real-world light source environment and demonstrate that our methods can achieve streaming reconstruction and analysis rates sufficient to support real-time steering of light source experiments. We believe that this is the first work that enables real-time experimental steering using large-scale compute resources for synchrotron light source experiments.

The rest of this paper is as follows. We provide background information on our target data analysis problem, tomographic image reconstruction, in Section II. We introduce our system architecture and the integration of its components in Section III, and present its evaluation in Section IV. We discuss related work in Section V, and we present our conclusions in Section VI.

II. Background

We briefly explain the tomographic data acquisition and measurement process and then describe the image reconstruction problem of recovering an object from measurement data.

Figure 1 shows a typical tomography experiment. An object placed on a rotary stage is illuminated by an x-ray beam and the transmitted photons are collected by using a detector. Since photons attenuate as they pass through the object, the measurements are proportional to density: that is, measurements from dense or thicker regions lead to low detector readings, whereas less dense or thin regions lead to high readings.

The tomographic data acquisition process requires rotating the stage in multiple known increments (degrees) around a rotation axis while collecting data (projections).

A. Measurement Process

The Radon transform lies at the heart of the x-ray tomography measurement process. Its discovery dates back to Johann

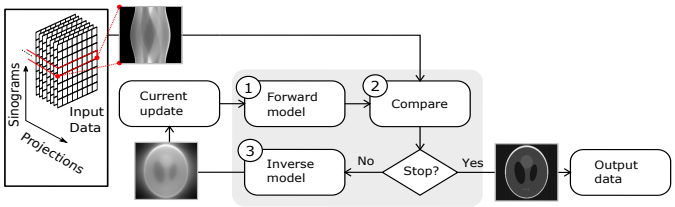


Fig. 2: Schematic of the iterative reconstruction process. (1) An initial guess for the model estimates is used to simulate data using the forward model; (2) the simulated data are compared with the measured data; and (3) each model estimate (i.e., direction and step size for each parameter) is updated, based on the employed algorithm, until a stopping criterion is met.

Radon's demonstration that a differentiable function on $\mathbb{R}^2$ can be uniquely determined from its integrals over lines in $\mathbb{R}^2$ [3]. The theory holds for three- or higher-dimensional objects, but for simplicity we present only the two-dimensional case here. Let f be a 2D function (e.g., specimen density) representing an unknown arbitrary object. That is, suppose $f(x, y)$ defines a density distribution of a sample at spatial coordinates (x, y). The two-dimensional Radon transform of f is given by

$$p_\theta(s) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta - y\sin\theta - s)dxdy, \quad (1)$$

where $p_\theta(s)$ is commonly referred to as the *sinogram* and θ and s are the sinogram parameters. In tomography, $p_\theta(s)$ is related to the measurement data through the Beer-Lambert law:

$$I_\theta(s) = I_0(s)\exp\left[-p_\theta(s)\right], \quad (2)$$

where $I_0(s)$ is the incident x-ray illumination on the sample and $I_\theta(s)$ are the collected measurements at a number of different θ angles as a result of a tomographic scan.

B. Image Reconstruction

The image reconstruction problem requires recovery of $f(x, y)$ from $p_\theta(s) = -\log\left[I_\theta(s)/I_0(s)\right]$. Numerous methods are suitable for this task; *filtered back projection* (FBP) and *iterative reconstruction* are the most commonly used.

FBP has been the traditional method of choice for reconstructing objects because of its ease in implementation and satisfactory computation speed. However, it has significant disadvantages considering real-world constraints. First, FBP requires a sufficient number of projections before it can reconstruct an object successfully. This requirement is problematic for real-time reconstruction, since only a limited number of projections may be available to process at any given time. Second, characteristics of the target specimen might prevent collection of enough projections, especially for low-dose tomography applications. Third, FBP is more susceptible to errors and noise in measured data, which are common due to experimental limitations at synchrotron light sources.

In contrast, iterative reconstruction algorithms can provide better image quality, albeit at the cost of more computing. Iterative methods converge to an optimum solution using

advanced object and data models and can provide better image quality than does FBP on a limited number of projections.

Although many variations exist, the basic iterative reconstruction method involves three major steps, as depicted in Fig. 2. First, an initial guess of the volume object, which might simply be an empty volume, is used to calculate the simulated data through a forward model. Second, the simulated data are compared with the measured data. Third, an update of each model estimate is performed based on the employed algorithm. Reconstruction of an object might require hundreds of iterations, depending on the experiment and sample.

The earliest and most basic form of iterative reconstruction is the algebraic reconstruction technique (ART), which involves solving a sparse linear system of equations in the form of $Af = p$ where p is the projection data, A is the forward projection operator, and f is the unknown 3D object to be determined. While ART provides satisfactory images and has a fast convergence rate, the iterations must be stopped before a deteriorating "salt and pepper" or "checkerboard" effect begins to degrade the object estimate. A variation of the ART method is the simultaneous iterative reconstruction technique (SIRT) in which the updates to the solution are computed by taking into account all rotation angles simultaneously in one iteration, as follows:

$$f^{k+1} = f^k + \lambda A^T(p - Af^k). \tag{3}$$

As in the ART method, a relaxation parameter λ can be used to control convergence in certain cases. SIRT typically produces better quality reconstructions than does ART and is more robust to outliers in the measurement data. In our system, we use more advanced iterative reconstruction algorithms. However, the main computational steps remain the same.

C. Organization of Tomography Datasets

A tomography dataset that is generated at synchrotron light sources typically consists of a set of 2D projections. These projections are organized similar to *input data* in Fig. 2. The parallelization of reconstruction at sinogram level is trivial, since there is no dependency between neighboring sinograms. However, once a sinogram is distributed among several processes, those processes must synchronize at the end of each iteration.

The reconstructed image dimensions are determined according to the projection size; specifically, if projection dimensions are (y,x), then the reconstructed image dimensions typically are set to (y,x,x). For instance, if the dimensions of a tomography dataset are (180, 2,048, 2,048), that is, 180 projections where each of them has (2,048, 2,048) pixels, then the reconstructed image's dimensions are (2,048, 2,048, 2,048). Notice that the size of the reconstructed image is independent of the number of collected projections.

III. System Design

Our system consists of three main components: (1) *data acquisition and distribution*, which manages data collection from detectors and the distribution of those data to analysis processes; (2) the *analysis system*, which is responsible for analysis and reconstruction of streaming data; and (3) the *controller*, which analyzes reconstructed data. In the following subsections, we explain each of these components in detail.

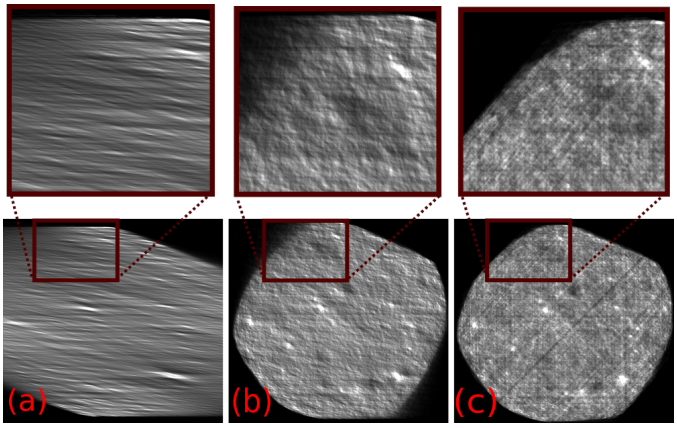


Fig. 3: Reconstructed image of a shale sample with only 30 streamed projections: (a) fixed angle, offset=1°; (b) interleaved, offset=5°; (c) optimized interleaved. The range of angles is [0,180)°.

A. Data Acquisition and Distribution

Data acquisition at current tomography beamlines is typically performed with one of two methods: *fixed angle* rotation or *interleaved*.

With *fixed-angle* rotation, acquisition starts at a specified starting point and then increments by a specified angle offset to a specified ending point. If, for example, the starting and ending angles are 0° and 180°, respectively, and the offset is 1°, this strategy results in a sequence of 180 projections at (0, 1, 2, ..., 178, 179)°.

In contrast, *interleaved* data acquisition collects data in several rounds, each involving a full rotation with a wider angle offset and with the starting angle selected to collect a disjoint set of projections. For example, with an offset of 5° and the starting angle advanced by 1° in each round, then after five rounds we have 180 projections at (0, 5, 10. ..., 175, 1, 6, ..., 174, 179)°.

If, as in most beamlines today, processing occurs only after data acquisition has completed, the choice of acquisition scheme has little impact on most analysis tasks. For real-time stream reconstruction, however, interleaved acquisition is superior to fixed angle, since it significantly improves the initial convergence rate of reconstruction.

In our system, we use an optimized version of interleaved data acquisition, in which the offset starts from the widest angle and is halved after each round. For the previous example, this strategy results in a sequence of projections at (0, 90, 45, 135, 22, 67, ..., 179)°. One potential problem with this approach is that if too many projections are collected, it collects projections with very small angles. This problem can be addressed by specifying an offset threshold below which the data acquisition strategy changes from optimized interleaved to interleaved.

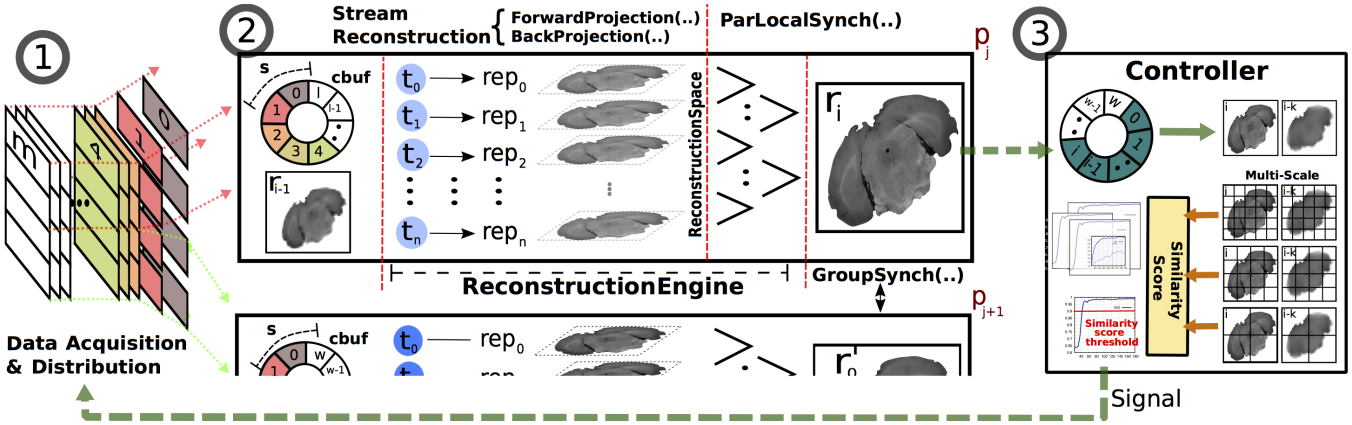


Fig. 4: Distributed stream reconstruction workflow with control feedback loop. **Data acquisition** partitions projections as they become available and streams the partitioned chunks to the reconstruction processes, p_j. The **reconstruction** processes store these chunks in a circular buffer, `cbuf`. Each time s chunks are received, the reconstruction engine performs a parallel reconstruction operation using r_{i-1} (i is the current iteration) and `cbuf`, and then sends the newly reconstructed image to the controller, which keeps w reconstructed images in its own circular buffer. The **controller** performs a similarity check on each new image; if the similarity score exceeds a user-defined threshold, a *finalize* signal is sent to the data acquisition process.

Figure 3 shows the reconstructed images that are obtained when 30 projections are collected with the aforementioned data acquisition methods. The optimized interleaved acquisition method, (c), provides the best image quality, primarily because it collects projections over multiple rounds and thus obtains a better sampling of angles, whereas the other methods perform only one partial round.

The projection generation is monitored by a data acquisition process. Each generated projection is read from the data acquisition machine's memory and distributed equally over the reconstruction processes. The distribution is performed along the y dimension of the projection. For example, if the generated projection's dimensions are 2048×2048 and there are 128 reconstruction processes, then the projection data are partitioned into 128 *chunks*, each of size 16×2048. These chunks are then streamed to the corresponding processes for reconstruction. This process is illustrated with ① in Fig. 4.

B. Analysis System

Algorithm 1 presents pseudocode for the analysis system, ② in Fig. 4. Step 1 initializes the communication structure by using the `CommInit()` function to establish two communication channels: one with other reconstruction processes (using MPI) and another with control and data acquisition processes (using the ZeroMQ distributed messaging library [4]). Step 2 sets up communication with the data acquisition machine, allocating a circular buffer `cbuf` and setting its two parameters: l, which determines the number of chunks that can be stored in the buffer, and s, which sets up the frequency of reconstruction operation. The function `SetupComm` accomplishes these tasks and returns projection metadata, `PMetadata`.

One key piece of information carried in `PMetadata`, the dimension of the projections, is used to initialize the buffers and data structures used in the intermediate processing layer (steps 3 and 4). This layer extends our MapReduce-like processing structure, exposing an API to its users for parallelization of reconstruction algorithms [5–7]. The extended processing layer overlaps data retrieval, chunk distribution, analysis, and synchronization operations to accommodate analysis of streaming data. In order to provide maximum parallelization, this layer uses *full replication*, in which each thread (map task) operates on its own buffer (image replica, i.e., $rep_{\#}$ in Fig. 4).

```
Input  : DAQAddr   // Data acq. process address
         ContAddr  // Controller process address
         t, l, s   // # threads; w. len; step size
         FProj, BProj   // Comp. kernels
         RImage    // Image with initial values
Output : RImage    // Final reconstructed image

1  CommInst ← CommInit ();
2  (TStreamInst, PMetadata ) ← SetupComm ( DAQAddr,
     CommInst.Rank, CommInst.World, l, s);
3  ReconSpace ← InitReconSpace (PMetadata);
4  ReconEngine ← InitReconEngine (ReconSpace, t);
5  while true do
6  |  CBuf ← TStreamInst.ReadCBuf () ;
7  |  if CBuf == NULL then
8  |  |  Break ;
9  |  ReconEngine.StreamRecon (CBuf, RImage, FProj) ;
10 |  ReconEngine.ParallelLocalSynch () ;
11 |  if CommInst.SharedImage (PMetadata) then
12 |  |  CommInst.GroupSynch
   |  |    (ReconSpace.MainReplica) ;
13 |  ReconEngine.Update (RImage, BProj) ;
14 |  ReconSpace.ResetReplicas () ;
15 |  CommInst.Publish (RImage) ;
16 end
```

Algorithm 1: Pseudocode for runtime system

The image replicas are allocated, set up, and managed in the *reconstruction space* (step 3). The reconstruction space is then passed to the *reconstruction engine*, where later threads are initialized and executed on their corresponding replicas (step 4). Once all buffers are set and threads are initialized, the analysis system waits for data acquisition to start streaming chunks (step 5).

The analysis system reconstructs `RImage` (r_i in Fig. 4) repeatedly until the `ReadCBuf()` function returns a null value. The number of reconstruction computations performed depends mainly on the (unknown) number of streamed chunks and the `cbuf`'s s parameter. Specifically, a reconstruction operation is triggered after receiving each s chunks. Notice that s and l also define the number of times each chunk is processed. For example, if $l = 24$ and $s = 2$, then `cbuf` contains 24 chunks at any given time during the execution. Since s is set to 2, the reconstruction operations are triggered after receiving every other chunk. Therefore, each chunk is processed 24/2=12 times before it is replaced by another.

Once a `ReadCBuf` function returns with a valid `CBuf`, the analysis system's `ReconEngine` starts scheduling threads with the `StreamRecon()` function. Each scheduled thread reads a portion of the data chunk from `CBuf` and applies the user-defined `FProj` function (shown as `ForwardProjection` in Fig. 4) to its corresponding replica. Threads use their assigned chunk data and the partially reconstructed 3D image (`RImage`) from the previous iteration to update their replicas.

After all chunks in `CBuf` are processed, threads perform parallel local synchronization, `ParallelLocalSynch()`. The replicas are then merged and reduced with a user-defined operation (*sum* in most reconstruction algorithms). The result is a single replica, `MainReplica`, in `ReconSpace`. If any 3D image slice is being reconstructed by multiple processes, a group synchronization is also performed and `MainReplica` updated at all corresponding processes. The `MainReplica` is then used with the user-provided `BProj` function to update and generate the new 3D image (`RImage`), step 13.

Replicas in the reconstruction space are then reset for the next iteration, and the newly reconstructed image is sent to the subscribed controller process.

C. Controller

The controller, ③ in Fig. 4, receives and analyzes images as they are produced by reconstruction processes. It can then *steer* experiments according to user-specified constraints. Here, we focus on a steering scenario in which the controller aims to finalize data acquisition once the reconstructed image reaches a specified quality level. This strategy can enable scientists to collect only a sufficient number of projections from the specimen, for example to minimize dose exposure, experiment time, or data analysis time.

Our approach is based on comparing each reconstructed image with those obtained from previously streamed data and observing the change in the similarity index. Since early sets of projections have more influence over the reconstructed image than later sets have, the similarity scores of consecutive reconstructed images initially show higher variability. This trend decreases as more projections are processed and the reconstructed image values converge to a refined solution.

The controller stores reconstructed images in a circular buffer. Then, for each incoming image r_i, a similarity score between r_i and r_{i-k} is calculated (where $k < i$). We denote this comparison as $s_i = \textit{compare}(r_i, r_{i-k}, \textit{scale})$, in which s_i is the similarity score and *scale* is the granularity of the comparison. A higher-scale value causes smaller features to be compared and reported in the similarity score. Depending on the s_i and user-provided similarity score constraint, the controller sends a *finalize* signal to the data acquisition process.

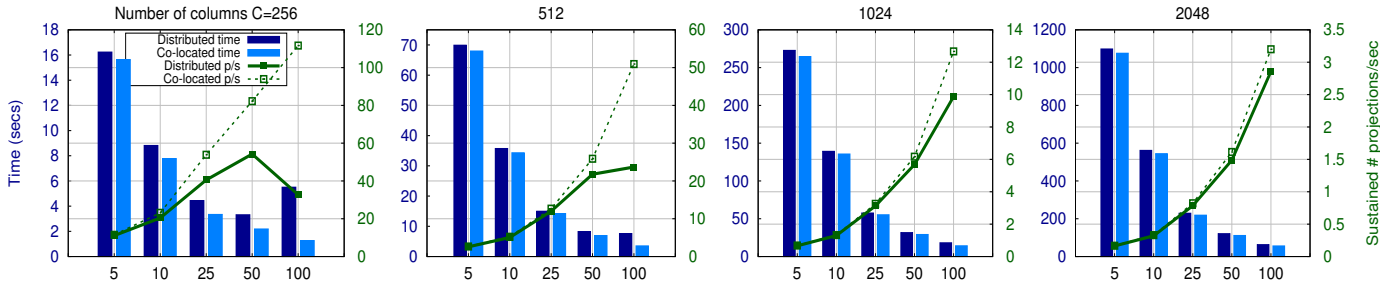
We use the Multi-Scale Structural Similarity Index (MS-SSIM) [8] to compute the similarity score. MS-SSIM considers three criteria at multiple scales that are crucial for the quantifying the quality of reconstructed tomography data [9]: luminance, structure, and contrast. The resulting similarity score s ranges over [0,1], with 1 indicating a perfect match.

The k parameter defines how similarity scores vary between compared images and thus is important to get right. For example, when $k = 1$, consecutive images are compared, which typically yields a high similarity score irrespective of the number of projections processed. This, in turn, makes it difficult to reason about improvements in image quality. When k is set to a sufficiently large number, however, the compared images will have quantifiable dissimilarities up to a point where their values are converged. The selection of k is not trivial: it depends on the data acquisition technique as well as the window length and step size used by the reconstruction process. Specifically, while the data acquisition technique determines which projections contribute to reconstruction process, the window length and step size define the number of contributing projections and the frequency of image reconstruction. In our system, we set the parameter k to the window length used by the reconstruction process (i.e., l) in order to improve the variance between consecutive similarity scores.

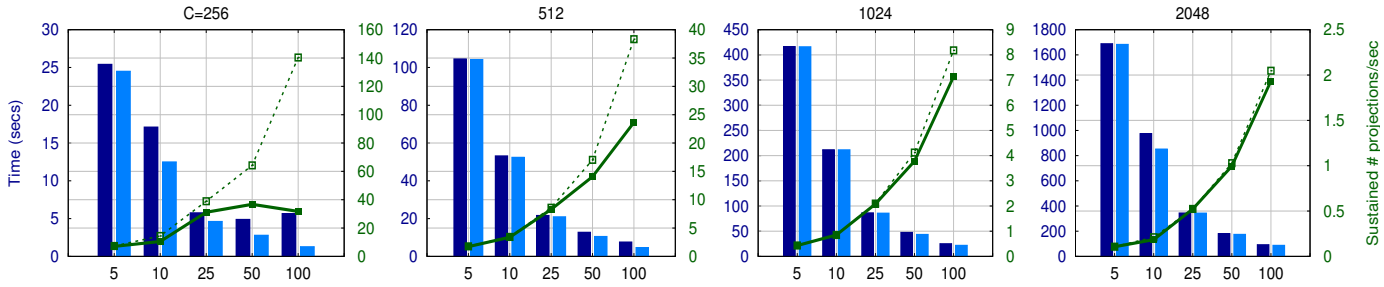
All three system components are nonblocking; thus, communication between and computation within the components can be overlapped. We use message sequence numbers (derived from projection ids and reconstruction processes' ranks) and the underlying communication library, ZeroMQ, to ensure exactly-once processing semantics.

IV. Evaluation

We conducted extensive experiments to evaluate both the computational performance of our system and the quality of the reconstructed images that it generates. We performed these experiments in a real-world environment where data are streamed from the APS at Argonne National Laboratory and analyzed at the visualization cluster of the Argonne Leadership Computing Facility (ALCF). The data acquisition machine, located at the APS, is equipped with a 10 Gb ethernet card. The ALCF analysis cluster, located 1 km distant from the APS, comprises 126 compute nodes, each with 12 cores (two 2.4 GHz Intel Haswell CPUs, each with 6 cores) and 384 GB of



(a) Performance with the MLEM reconstruction algorithm.



(b) Performance with the PML reconstruction algorithm.

Fig. 5: Reconstruction performance for (a) MLEM, top, and (b) PML, bottom, on phantoms with 180 projections, 100 sinograms, and variously 256, 512, 1,024, and 2,048 columns, from left to right. Each graph gives results for both the distributed case and the co-located case, with the x-axis giving the number of compute nodes; the bars and the left y-axis the elapsed time in seconds; and the lines and right y-axis the number of projections processed per second (p/s).

memory. The cluster nodes use FDR InfiniBand for internode communication and have a 10 Gb ethernet card for fast external data transfer.

To differentiate between the impact of computation and data transfer on end-to-end performance, we conducted experiments in two modes: a **distributed** mode, in which data are acquired at the light source, and a **co-located** mode, in which data are already in memory on the analysis cluster before reconstruction. To ease system evaluation in distributed mode, we store previously collected/generated data in the data acquisition machine's memory, from where we stream them to the analysis cluster.

We performed experiments for several datasets, both phantoms and real samples, with different dimensions and thus different computational demands. We evaluated distributed versions of two iterative reconstruction algorithms: Maximum likelihood expectation maximization (MLEM) [10] and penalized maximum likelihood (PML) [11]. MLEM performs reconstruction operations on a single pixel at a time; PML requires neighboring pixel information for its updates and thus is more computationally demanding.

A. Stream Reconstruction Performance

We measured the end-to-end performance for four phantoms, with sizes (180, 100, C), $C \in \{256, 512, 1{,}024, 2{,}048\}$, and for both the MLEM and PML reconstruction algorithms, for a total of eight algorithm-phantom combinations. In each case, we set the window length to 24 and step to 1 and thus processed each projection 24 times.

We present our results in Fig. 5. Each of the eight graphs corresponds to a different algorithm-phantom combination. In each graph, we show for each of 5, 10, 20, 50, and 100 nodes (i.e., 60, 120, 240, 600, and 1,200 cores) four values, namely, the elapsed time and the processing rate in each of the distributed and co-located modes. *Distributed time* is the total time taken, encompassing both communication and processing, when data are collected at the beamline computer and processing occurs at the compute cluster, while *colocated time* is the time taken when data are preloaded on the compute cluster and thus no communication costs are incurred. Similarly, *distributed p/s* and *co-located p/s* give the number of projections processed per second in the distributed and co-located cases, respectively. For example, we see in the upper left graph (MLEM, C=256) that when running on five nodes, our system takes $\sim$16 seconds in distributed mode, which, since all phantom datasets have 180 projections, corresponds to 180 projections/16 seconds = $\sim$11 p/s.

Figure 5(a) presents system performance when using the MLEM reconstruction algorithm. For dataset C=256, the projection consumption rate increases up to the 50-node configuration, in which the system can sustain processing 54 p/s. The 100-node configuration, however, shows a significant performance decrease. The main reason for this behavior is the increased communication cost. Each node in the 100-node configuration operates only on 180×256 ray-sum values, taking

less than 1.2 seconds in total. On the other hand, transferring these data from the data acquisition machine to the 100 nodes introduces ∼241% overhead, which can also be observed from the gap between *distributed* and *co-located* rates. To further understand the communication overhead, we performed

In the C=512 dataset, we see better scalability, as the computation time dominates the end-to-end time. The 100-node configuration, in this case, provides a 9.1x speedup relative to five nodes; while not perfect, this speedup is better than that seen for 100 nodes and C=256. For the remaining datasets, $C\in\{1{,}024, 2{,}048\}$, the performance increase is consistent with the increasing number of nodes. The strong scaling efficiencies for these datasets range from 86.1% to 98%; one exception is the C=1,024 dataset on 100 nodes, in which the scaling efficiency is 74.8%. Overall the *distributed p/s* values range from 3 p/s to 54 p/s, with the highest rates observed with the 50- and 100-node configurations.

Figure 5(b) shows PML results for the same experiments. We see a similar performance trend to that observed for MLEM, but with slightly lower reconstruction rates due to the fact that PML is computationally more demanding. For the C=256 dataset, the 50-node configuration provides the best performance with 36.7 p/s, which is 33.3% less than for the same MLEM configuration. On 100 nodes, the rate drops to 31.7 p/s, on par with the 32.7 p/s seen for MLEM, indicating that execution is communication bound.

With C=512, the scaling efficiency ranges from 81.7% to 98.2% on 10, 25, and 50 nodes relative to performance on five nodes. On 100 nodes, the total computation time is 4.69 seconds; here communication overhead becomes significant, and efficiency drops to 68.5%. Still, the system can sustain a processing rate of 23.6 p/s with 100 nodes, a speedup of 1.6 over that seen on 50 nodes.

For C=1,024 and 2,048, speedups and scaling efficiencies are higher, as computation dominates overall execution time. For C=1,024, scaling efficiencies are 82.5–98.4%. On 100 nodes, we see a reconstruction rate of 7.1 p/s: 16.5x higher than on five nodes. Similar performance results are observed for C=2,048, with a scaling efficiency of more than 87% for all configurations. The highest reconstruction rate is 1.93 p/s on 100 nodes. Here the difference between the *distributed* and *co-located* rates is only 0.11 p/s.

B. Performance Effect of Runtime Parameters

We have seen that system performance is sensitive to dataset sizes, number of compute nodes, and communication overhead. Overall system performance can also be altered with runtime parameters. Specifically, the *window length* (l) and *step size* (s) can be adjusted to match the rate at which a detector generates projections.

We show in Fig. 6a the effect on the reconstruction rate of changing the l and s parameters for the C=1,024 dataset. The contour lines show the boundaries of rates with respect to different l and s settings. For the extreme case, where l=12 and s=12, system performance is maximized at 204.1 p/s. Since l sets the number of projections that are processed for each

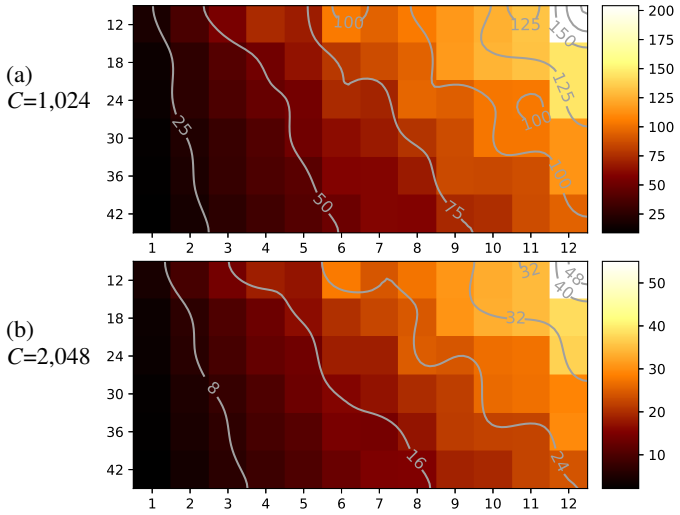


Fig. 6: Sustained p/s as a function of step size (x-axis) and window length (y-axis) for two datasets, with dimensions (180, 100, C), $C\in\{1{,}024, 2{,}048\}$, when in distributed mode and using MLEM on 100 nodes (1,200 cores). Color represents p/s, which reaches 204 for C=1,024 and 55 for C=2,048.

reconstruction iteration, increasing this parameter elevates the computational demand and thus reduces reconstruction rates. At the other end of the spectrum, where l =42 and s=1, performance drops to 8.82 p/s. Thus, setting l and s to 12 can provide a 23.1x higher reconstruction rate relative to the l=42 and s=1 configuration.

Figure 6b shows results when C=2,048. We first notice the similarity of color mapping between Figs. 6(a) and (b), indicating that the distribution of computational demands for different datasets follows the same trend. This insight can be used to determine, with minimal effort, l and s parameters as well as the amount of computational resources. Specifically, if projection generation rate and projection dimensions are known in advance (which are typically provided with detector specs), a single experiment can provide sufficient information to estimate reconstruction rates for the remaining l and s combinations.

We again see the best performance when both l and s are set to 12 for C=2,048: a reconstruction rate of 55 p/s. This setting provides a 25.6x speedup relative to l=42 and s=1. This speedup also supports the aforementioned statement on performance estimation, since it is consistent with the measured speedup for C=1,024 dataset.

While the l and s parameters can be used to adjust the reconstruction rate, these parameters also affect the image quality by determining the amount of *new* information used in each reconstruction iteration. The appropriate balance between image quality and the stream reconstruction performance depends greatly on the use case. For instance, if the ultimate goal is to reconstruct images with very small features, then larger window lengths and small step sizes can help improve achieved quality.

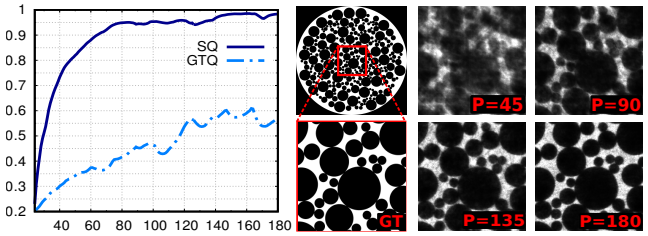


Fig. 7: Quality tests for foam phantom. The x-axis in the graph is the number of projections processed, and the y-axis is the MS-SSIM similarity score: for *SQ*, between reconstructed images only, and for *GTQ*, between the most recently reconstructed image and ground truth. The images on the right show the reconstructed images after processing 45, 90, 135, and 180 projections, respectively.

C. Quality Cutoffs for Experimental Steering

We next analyze the effect of stream processing on the quality of reconstructed images. As mentioned in Section III-C, we use MS-SSIM to compute similarity scores, in order to track how reconstructed images vary during execution.

We first consider a foam phantom which captures some common features, such as circles and pores, that occur in samples analyzed at synchrotron light sources. These features can be challenging to reconstruct: if the features in the sample are small and spatially close, more iterations and projections are required to distinguish them. We obtain 180 projections from this phantom dataset, ordered according to the *optimized interleaved* acquisition scheme. Each projection consists of a single row with *C*=1,024 columns, meaning that the dataset has dimensions (180, 1, 1,024). We denote the resulting projections as p_i, $i \in \{1, 2, 3, \ldots, 180\}$.

We set the runtime parameters as *l*=24 and *s*=1; thus, when each projection p_i is received by a worker, an image r_i is immediately reconstructed and transferred to the controller process for quality checking. Since *l*=24, the controller performs a similarity comparison between r_i and r_{i-24}. We further set *scale*=5, which causes MS-SSIM to check similarities according to the smallest features in the sample, namely, $s_i = compare(r_i, r_{i-24}, scale = 5)$.

Figure 7 shows our results. We first consider *SQ*, similarity calculated according to $s_i = compare(r_i, r_{i-24}, 5)$. This score increases rapidly at first, which is what we expect: since the difference between the reconstructed image and ground truth is initially large, the contribution of each new projection to the reconstruction is high. This trend continues while $p_{i<60}$ and then flattens off. However, subsequent projections still contribute significantly to the reconstruction, as we observe clear changes in the reconstructed image between $r_{i=90}$ and $r_{i=135}$ (labeled P=90 and P=135 in the figure, respectively). For $r_{i>140}$, the similarity scores are fairly consistent at ∼0.98.

GTQ, in contrast, is similarity relative to a *ground truth* image, r_{gt}, that is, $compare(r_i, r_{gt}, 5)$. We see that these scores fluctuate in the 0.5-0.6 band for $r_{i>140}$, indicating that the contributions of projections after $r_{i>140}$ are minimal. We conclude that for samples with foamlike features, we can reasonably have the controller send a *finalize* signal to the data acquisition process once a similarity score threshold of 0.95 is exceeded. For the foam phantom, this strategy results in the data acquisition process halting at $p_{i=140}$, providing a 1.29x speedup for both data acquisition and analysis relative to a full set of 180 projections.

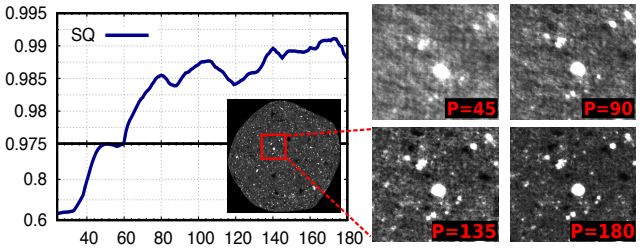


Fig. 8: Quality tests for data collected from a shale sample at the APS.

In Fig. 8, we repeat the same experiment with a real-world shale dataset. Since ground truth is not available here, only *SQ* is shown. Much as with the phantom dataset, the similarity score increases rapidly with the initial projections and then flattens off after $p_{i=60}$. Notice that after p_{60}, the scale of the y-axis in the figure is changed so that fluctuations in similarity scores are more visible. We still observe slight increases in the similarity scores up to $p_{i=100}$; but after this point the changes are minor (fluctuations), and we conclude that subsequent projections are not necessary. Since the features in this sample are much smaller and sparser than in the phantom data, MS-SSIM shows higher similarity scores, close to 0.985. Collecting 100 rather than 180 projections provides a 1.8x speedup for both data acquisition and analysis.

V. Related Work

Real-time data analysis and computational steering have been extensively studied in many fields [12–15]. Much recent work focuses on in situ analysis of simulation data, where data produced by simulation are analyzed on the same computer while the simulation is running [16, 17]. Although some of these works address problems that also occur in synchrotron data analysis, experiment-specific constraints, data generation rates, computational requirements, and available computational resources in light source facilities create unique challenges [18], as for example when working with large-scale brain imaging and analysis of data from dose-sensitive specimens [19, 20].

Real-time steering in experimental science, especially in scenarios involving large data, has received much less attention than computational steering [21, 22]. At the SC'98 conference, Laszewski et al. [21] demonstrated a system to do quasi-real-time analysis of synchrotron light source data using high-speed networks and computational grids. The National Center for Microscopy and Imaging Research developed software to integrate data acquisition from electron microscope, computational resources, and visualization in a distributed environment [23]. Although these works address some issues relevant to synchrotron light source data problems, none focuses on autonomous experimental steering using high-

performance computing resources. And much has changed in terms of detector, compute, and network capabilities, as well as the requirements and complexities within end-to-end processing pipelines, since this pioneering work. For example, the filtered back projection (FBP) technique used by Laszewski et al. [21] for tomographic reconstruction of 3D images is no longer viewed as effective [24]. Iterative reconstruction [25] is preferred because it can reconstruct the image with many fewer projections than FBP needs, reducing radiation exposure of the samples. However, iterative reconstruction is computationally more expensive and requires fine-grained parallelization for real-time processing [5, 26–28].

Stevanovic et al. [29] use FPGAs for real-time analysis and experimental steering at the ANKA synchrotron radiation facility. FPGAs can provide timely feedback for light-weight computational problems with small data, but data-intensive analysis tasks are not suitable for these types of devices.

Accelerators, including Xeon Phi [30, 31] and GPUs [32, 33], have been used extensively for high-performance reconstruction and analysis of x-ray images. Especially in medical imaging, different iterative reconstruction approaches are implemented and optimized for GPUs in order to generate high-quality 3D images [34, 35]. In more recent work, Vogelgesang et al. developed UFO, a computational framework for image-processing algorithms, which used GPUs in streaming mode to address synchrotron data analysis problems [36]. Although GPUs can provide high computational throughput, they can accommodate only small datasets and are not suitable for large-scale tomography data.

The key aspect of experimental steering (the focus of this work)—processing a data stream from a scientific instrument in near-real time and making decisions—seems to have much in common with many big data applications. According to a NIST survey [37], 80% of big data applications involve streaming, spurring the development of stream-processing systems such as Twitter's Heron [38], Googles Millwheel [39], Spark streaming [40], and IBM Stream Analytics [41]. But the streaming challenges in enterprise applications are different. For example, individual events in enterprise streams tend to be small: often only a few bytes.

Some of these tools may apply to sensor networks in scientific domains (e.g., wide-area earthquake sensor networks [42, 43], Ocean Observatories Initiative [44], urban observatories [45]) under certain conditions. However, light source instruments can generate data at rates of gigabytes per second and thus require complex, highly parallel analysis involving communication among the threads and processes used to perform the analysis.

Despite these efforts, real-time steering with control by either humans or automated processing is currently not generally available in experimental science environments.

VI. Conclusion

We have presented new methods for real-time data analysis and experimental steering at synchrotron light sources. We described an innovative distributed stream-processing system that can use remote compute resources to meet the demanding computational requirements of tomographic reconstruction tasks. We implemented a control-feedback loop that uses a similarity metric to evaluate the quality of reconstructed images and that decides, based on that metric, when to terminate data acquisition.

We demonstrated our system in a real-world environment in which projection datasets are streamed from a data acquisition machine at a synchrotron light source to a remote HPC cluster for reconstruction and analysis. We evaluated our system with two different iterative reconstruction algorithms, each of which we validated from perspectives of both performance and image quality with a variety of phantom and real datasets. We showed that our system can achieve reconstruction rates as high as 204.1 projections per second when using 1,200 cores. We further showed that our experimental steering approach can reduce data acquisition time by 22–44% for the datasets considered in our experiments by gracefully finalizing data acquisition when reconstructed image quality exceeds a specified threshold. This reduction in data acquisition time translates to more efficient utilization of both scientist and scientific instrument time.

Much of our work is retargetable to other synchrotron light source analysis tasks. For example, the data acquisition component can be used for any pixelated detector, and many modalities can be implemented using our parallel processing framework, including correlation analysis for x-ray photon spectroscopy, ptychographic reconstruction, and fitting of fluorescence data.

Acknowledgments

This material is based upon work supported by the U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research and Basic Energy Sciences, under Contract DE-AC02-06CH11357. We gratefully acknowledge the computing resources provided and operated by the Argonne Leadership Computing Facility, which is a U.S. Department of Energy, Office of Science User Facility, and experimental facilities at the Advanced Photon Source.

References

[1] "Next-generation photon sources for grand challenges in science and energy," https://science.energy.gov/~/media/bes/pdf/reports/files/Next-Generation_Photon_Sources_rpt.pdf, 2009, Accessed: 2017-06-13.

[2] "The report of the BES advisory subcommittee on future x-ray light sources," https://science.energy.gov/~/media/bes/besac/pdf/Reports/Future_Light_Sources_report_BESAC_approved_72513.pdf, 2013, Accessed: 2017-06-13.

[3] J. Radon, "On determination of functions by their integral values along certain multiplicities," *Ber. der Sachische Akademie der Wissenschaften Leipzig*, vol. 69, pp. 262–277, 1917.

[4] iMatrix Corporation, "ZeroMQ: Distributed Messaging Library," http://zeromq.org, 2014, Accessed: 2017-06-13.

[5] T. Bicer, D. Gursoy *et al.*, "Rapid tomographic image reconstruction via large-scale parallelization," in *European Conference on Parallel Processing*. Springer Berlin Heidelberg, 2015, pp. 289–302.

[6] W. Jiang, V. T. Ravi *et al.*, "A Map-Reduce system with an alternate API for multi-core environments," in *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, ser. CCGRID '10. Washington, DC, USA: IEEE Computer Society, 2010, pp. 84–93. [Online]. Available: http://dx.doi.org/10.1109/CCGRID.2010.10

[7] T. Bicer, "Supporting data-intensive scientific computing on bandwidth and space constrained environments," Ph.D. dissertation, The Ohio State University, 2014.

[8] Z. Wang, E. P. Simoncelli *et al.*, "Multiscale structural similarity for image quality assessment," in *37th Asilomar Conference on Signals, Systems, and Computers*, vol. 2, Nov 2003, pp. 1398–1402 Vol.2.

[9] D. J. Ching and D. Gürsoy, "XDesign: An open-source software package for designing X-ray imaging phantoms and experiments," *Journal of Synchrotron Radiation*, vol. 24, no. 2, pp. 537–544, 2017.

[10] J. Nuyts, C. Michel *et al.*, "Maximum-likelihood expectation-maximization reconstruction of sinograms with arbitrary noise distribution using NEC-transformations," *IEEE Transactions on Medical Imaging*, vol. 20, no. 5, pp. 365–375, 2001.

[11] J. A. Fessler and A. O. Hero, "Penalized maximum-likelihood image reconstruction using space-alternating generalized EM algorithms," *IEEE Transactions on Image Processing*, vol. 4, no. 10, pp. 1417–1429, 1995.

[12] D. M. Beazley and P. S. Lomdahl, "Lightweight computational steering of very large scale molecular dynamics simulations," in *ACM/IEEE Conference on Supercomputing*, 1996, pp. 50–50.

[13] S. G. Parker and C. R. Johnson, "SCIRun: A scientific programming environment for computational steering," in *IEEE/ACM SC95 Conference*, 1995, pp. 52–52.

[14] D. J. Jablonowski, J. D. Bruner *et al.*, "VASE: The visualization and application steering environment," in *Supercomputing '93.*, Nov 1993, pp. 560–569.

[15] J. Vetter and K. Schwan, "High performance computational steering of physical simulations," in *11th International Parallel Processing Symposium*, Apr 1997, pp. 128–132.

[16] P. Malakar, V. Vishwanath *et al.*, "Optimal scheduling of in-situ analysis for large-scale scientific simulations," in *SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2015, pp. 1–11.

[17] Y. Wang, G. Agrawal *et al.*, "Smart: A MapReduce-like framework for in-situ scientific analytics," in *SC15: International Conference for High Performance Computing, Networking, Storage and Analysis*, Nov 2015, pp. 1–12.

[18] T. Bicer, D. Gürsoy *et al.*, "Optimization of tomographic reconstruction workflows on geographically distributed resources," *Journal of Synchrotron Radiation*, vol. 23, no. 4, pp. 997–1005, Jul 2016. [Online]. Available: http://dx.doi.org/10.1107/S1600577516007980

[19] T. Bicer, D. Gürsoy *et al.*, "Trace: A high-throughput tomographic reconstruction engine for large-scale datasets," *Advanced Structural and Chemical Imaging*, vol. 3, no. 1, p. 6, 2017.

[20] D. Y. Parkinson, K. Beattie *et al.*, "Real-time data-intensive computing," in *AIP Conference Proceedings*, vol. 1741, no. 1. AIP Publishing, 2016, p. 050001.

[21] G. von Laszewski, M.-H. Su *et al.*, "Real-time analysis, visualization, and steering of microtomography experiments at photon sources," in *9th SIAM Conference on Parallel Processing for Scientific Computing*, San Antonio, TX, 22-24 Mar. 1999. [Online]. Available: http://cyberaide.googlecode.com/svn/trunk/papers/anl/vonLaszewski-siamCmt99.pdf

[22] Y. Wang, F. De Carlo *et al.*, "A high-throughput x-ray microtomography system at the Advanced Photon Source," *Review of Scientific Instruments*, vol. 72, no. 4, pp. 2062–2068, 2001.

[23] P. J. Mercurio, T. T. Elvins *et al.*, "The distributed laboratory: An interactive visualization environment for electron microscope and 3d imaging," *Commun. ACM*, vol. 35, no. 6, pp. 54–63, Jun. 1992. [Online]. Available: http://doi.acm.org/10.1145/129888.129891

[24] A. Moscariello, R. A. Takx *et al.*, "Coronary CT angiography: Image quality, diagnostic accuracy, and potential for radiation dose reduction using a novel iterative image reconstruction technique—comparison with traditional filtered back projection," *European Radiology*, vol. 21, no. 10, p. 2130, 2011.

[25] L. L. Geyer, U. J. Schoepf *et al.*, "State of the art: Iterative CT reconstruction techniques," *Radiology*, vol. 276, no. 2, pp. 339–357, 2015.

[26] D. J. Duke, A. B. Swantek *et al.*, "Time-resolved x-ray tomography of gasoline direct injection sprays," *SAE International Journal of Engines*, vol. 9, no. 2015-01-1873, 2015.

[27] D. Gürsoy, T. Biçer *et al.*, "Hyperspectral image reconstruction for x-ray fluorescence tomography," *Optics Express*, vol. 23, no. 7, pp. 9014–9023, 2015.

[28] ——, "Maximum a posteriori estimation of crystallographic phases in x-ray diffraction tomography," *Philosophical Transactions of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, vol. 373, no. 2043, p. 20140392, 2015.

[29] U. Stevanovic, M. Caselle *et al.*, "A control system and streaming DAQ platform with image-based trigger for x-ray imaging," *IEEE Transactions on Nuclear Science*, vol. 62, no. 3, pp. 911–918, June 2015.

[30] G. Teodoro, T. Kurc *et al.*, "Comparative performance analysis of Intel Xeon Phi, GPU, and CPU: A case study from microscopy image analysis," in *IEEE 28th International Parallel and Distributed Processing Symposium*, May 2014, pp. 1063–1072.

[31] E. Serrano, G. Bermejo *et al.*, "High-performance x-ray tomography reconstruction algorithm based on heterogeneous accelerated computing systems," in *2014 IEEE International Conference on Cluster Computing (CLUSTER)*, Sept 2014, pp. 331–338.

[32] F. Xu and K. Mueller, "Accelerating popular tomographic reconstruction algorithms on commodity PC graphics hardware," *IEEE Transactions on Nuclear Science*, vol. 52, no. 3, pp. 654–663, 2005.

[33] W. van Aarle, W. J. Palenstijn *et al.*, "The ASTRA toolbox: A platform for advanced algorithm development in electron tomography," *Ultramicroscopy*, vol. 157, pp. 35–47, 2015.

[34] C.-Y. Chou, Y.-Y. Chuo *et al.*, "A fast forward projection using multithreads for multirays on GPUs in medical image reconstruction," *Medical Physics*, vol. 38, no. 7, pp. 4052–4065, 2011.

[35] D. Lee, I. Dinov *et al.*, "CUDA optimization strategies for compute- and memory-bound neuroimaging algorithms," *Computer Methods and Programs in Biomedicine*, vol. 106, no. 3, pp. 175–187, 2012.

[36] M. Vogelgesang, S. Chilingaryan *et al.*, "UFO: A scalable GPU-based image processing framework for on-line monitoring," in *14th IEEE International Conference on High Performance Computing and Communication*, June 2012, pp. 824–829.

[37] NIST, "NIST Big Data Public Working Group (NBD-PWG) Home Page. 2013," http://bigdatawg.nist.gov/home.php, 2014, Accessed: 2017-06-13.

[38] S. Kulkarni, N. Bhagat *et al.*, "Twitter Heron: Stream processing at scale," in *ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '15. New York, NY, USA: ACM, 2015, pp. 239–250. [Online]. Available: http://doi.acm.org/10.1145/2723372.2742788

[39] T. Akidau, A. Balikov *et al.*, "MillWheel: Fault-tolerant stream processing at Internet scale," *Proc. VLDB Endow.*, vol. 6, no. 11, pp. 1033–1044, Aug. 2013. [Online]. Available: http://dx.doi.org/10.14778/2536222.2536229

[40] M. Zaharia, T. Das *et al.*, "Discretized streams: An efficient and fault-tolerant model for stream processing on large clusters," in *4th USENIX Conference on Hot Topics in Cloud Ccomputing*, ser. HotCloud'12. Berkeley, CA, USA: USENIX Association, 2012, pp. 10–10. [Online]. Available: http://dl.acm.org/citation.cfm?id=2342763.2342773

[41] M. Hirzel, H. Andrade *et al.*, "IBM streams processing language: Analyzing big data in motion," *IBM Journal of Research and Development*, vol. 57, no. 3/4, pp. 7–1, 2013.

[42] H. S. Kuyuk, R. M. Allen *et al.*, "Designing a network-based earthquake early warning algorithm for California: ElarmS-2," *Bulletin of the Seismological Society of America*, vol. 104, no. 1, pp. 162–173, 2014.

[43] N. Nakata, J. P. Chang *et al.*, "Body wave extraction and tomography at long beach, california, with ambient-noise interferometry," *Journal of Geophysical Research: Solid Earth*, vol. 120, no. 2, pp. 1159–1173, 2015.

[44] T. Cowles, J. Delaney *et al.*, "The Ocean Observatories Initiative: Sustained ocean observing across a range of spatial scales," *Marine Technology Society Journal*, vol. 44, no. 6, pp. 54–64, 2010.

[45] D. E. Boyle, D. C. Yates *et al.*, "Urban sensor data streams: London 2013," *IEEE Internet Computing*, vol. 17, no. 6, pp. 12–20, 2013.

LICENSE