High-Performance Parallel and Stream Processing of X-ray Microdiffraction Data on Multicores

Michael A. Bauer[1], Alain Biem[2], Stewart McIntyre[1], Nobumichi Tamura[3] and Yuzhen Xie[1]

[1]University of Western Ontario, London, Canada

[2]IBM Research, Yorktown Heights, United States

[3]Advanced Light Source, Lawrence Berkeley National Lab, United States

E-mail: bauer@uwo.ca; biem@us.ibm.com; smcintyr@uwo.ca; ntamura@lbl.gov; yxie@csd.uwo.ca

Abstract. We present the design and implementation of a high-performance system for processing synchrotron X-ray microdiffraction (XRD) data in IBM InfoSphere Streams on multicore processors. We report on the parallel and stream processing techniques that we use to harvest the power of clusters of multicores to analyze hundreds of gigabytes of synchrotron XRD data in order to reveal the microtexture of polycrystalline materials. The timing to process one XRD image using one pipeline is about ten times faster than the best C program at present. With the support of InfoSphere Streams platform, our software is able to be scaled up to operate on clusters of multi-cores for processing multiple images concurrently. This system provides a high-performance processing kernel to achieve near real-time data analysis of image data from synchrotron experiments.

1. Introduction

High-brilliance synchrotron X-ray microbeams combined with fast large-area two-dimensional detectors can produce large blocks of X-ray microdiffraction (XRD) image data at unprecedented rates. In principle, up to 1 to 10 images per second of 4MB size can be produced. This data enables scientists to study crystalline texture in polycrystalline materials using Polychromatic X-ray Microscopy (PXM), and has led to refinements that permit the mapping of crystallographic orientation and deviatoric strain of such materials [7, 11, 12]. Even at this early stage, such measurements involve the accumulation of data sets as large as 60,000 of 4MB images. Using even advanced desktop computers it would take several days to index the diffraction patterns produced using software such as XMAS (X-ray Microdiffraction Analysis Software) developed at the Advanced Light Source [2] and the 3D X-ray Microdiffraction Analysis Software Package at the Advanced Photon Source [3]. A recent work [8] attempted to use cell processors to accelerate the processes.

Funded by CANARIE [1], the Active Network Interchange for Scientific Experimentation (ANISE) project [15] aims at developing a high speed network and processing systems to achieve near real-time data analysis for synchrotron experiments. Being able to process this data in near

[1] http://www.canarie.ca/

real-time would enable users to make sound decisions about issues on time-sensitive materials under study. In this paper we report on the development of a software system for analyzing *synchrotron white-beam Laue diffractions* using parallel and stream processing on clusters of off-the-shelf multi-core processors: today's mainstream architectures. This is the result of an ANISE sub-project, serving as one of the high-performance data processing kernels for synchrotron experimental data analysis.

Stream processing is a data-centric programming model. Given a set of data which can be structured as a stream of tuples, a series of operations (processing elements) are applied to each tuple or some set of tuples in the stream. Processing elements are usually pipelined, much like an assembly line. Pipelining does not reduce the time for individual instruction execution, but instead, it improves instruction throughput. Our implementation platform is the *IBM InfoSphere Streams* [6]. InfoSphere Streams enables the development and execution of applications that process information in data streams. It provides built-in operators as well as a programming interface for end-users to create user-defined operators to operate on data streams. The runtime system and the scheduler of InfoSphere Streams help to seamlessly deploy stream applications to clusters of multi-core processors.

We consider a sequence of procedures for analyzing synchrotron white-beam Laue diffractions for the mapping of crystallographic orientation and deviatoric strain of polycrystalline materials. As reported in [7, 11, 12], the main process for analyzing Laue patterns consists of the following procedures: Image parsing, Background filtering (optional), Blob searching, Peak fitting, Indexing and Strain refinement. Following the resources of the three software packages mentioned above, we review the crystallographic principles of Laue diffraction and its mathematical model for the purpose of parallel and stream processing, which is presented in Section 2. In Section 3, we introduce our techniques for adapting the algorithms and restructure their memory access patterns for efficient parallel and stream processing. We also apply high-performance software engineering to the sequential procedures for processing images individually in order to optimize the code and improve memory usage. Our implementation in IBM InfoSphere Streams platform and benchmark results are summarized at the end of Section 3.5.

2. Review of Laue Microdiffraction Methods for Parallel and Stream Processing

Laue (white-beam) diffraction is a standard crystallographic method for determining crystal orientation without rotation of the sample. It is advantageous for micron-sized samples since current high quality diffractometers have a sphere of confusion of about 10μm and any rotation would move the sample out of the micro-beam. Synchrotron-based micro-beam combined with modern charge-coupled device (CCD) detector has made Laue microdiffraction a promising method for the study of material properties with submicron spatial resolution. In such measurement systems the sample remains fixed except for translation motion in the plane of the sample for scanning an area of interest.

Figure 1 is a schematic view of the local geometry around the sample presented in [1]. Most micro-beam Laue experiments such as [1, 13] use the following setup: The sample is inclined at approximately 45 degrees to the white X-ray beam. The Laue diffraction pattern is collected by the X-ray CCD camera located above the sample at $2\theta = 90$ degrees (angle between the incident beam and the vector from the sample to the detector centre). In general, the lower limit of the photon energy bandpass lies between 5 and 10 keV, and the upper limit between 22 and 35 keV.

When a broad-bandpass micro-beam intercepts a sample and illuminates a small number of crystal grains, the wave from the white radiation with particular wavelength that satisfies the Bragg law, $n\lambda = 2d \sin\theta$, is diffracted. In the previous formula, the parameter λ is the wavelengh of a wave, d is the spacing of the planes of crystal atoms, θ is the incident angle, and n is an integer. The diffracted beams produce Laue diffraction patterns, which are recorded by the CCD camera as reflection spots (i.e. energy-intensities and pixel positions) in a 2D image.

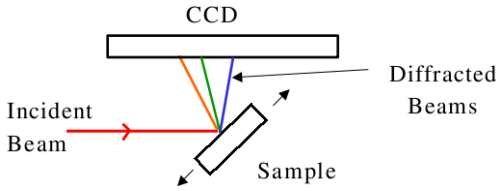

Figure 1: A schematic of X-ray microdiffraction geometry

Mathematical methods and software to obtain accurate strain information from synchrotron-based Laue patterns were first developed by a group at the Oak Ridge National Laboratory [3, 7, 12], and further improved at the Advanced Light Source [1, 2, 11]. To analyze synchrotron white-beam Laue diffractions for the mapping of crystallographic orientation and deviatoric strain of a single crystal or polycrystalline materials, the following sequence of processes takes place.

Image parsing: to convert a CCD image C in binary format (SPE, Pilatus TIFF or Mar TIFF) into a matrix E_0 of energy intensity values (integer or double). The coordinates of a pixel in C is encoded by the indices of the corresponding element in the matrix E_0.

Background filtering: to estimate the background B of E_0 (typically coming from elastic or inelastic scattering processes, fluorescence that do not contribute to the Bragg reflection peaks) and subtract B from E_0, generating an image E. This process is optional. However for some samples where the background is too high, it is necessary to filter out the background in order to accurately locate the reflection peaks.

Blob searching: to find the regions (or blobs) in E (or E_0 if no background filtering is applied) where pixel values are above a certain threshold (i.e. significant reflections) and then identify the potential peak positions of local maxima. Usually 8-way connectivity check and two-dimensional local maxima for peak searching are sufficient.

Peak fitting: to characterize the two-dimensional profile of each peak (by fitting to either Gaussian, Lorentzian or Pearson VII function) so as to find the correct positions of the reflection spots. Peak fitting is important because the accuracy of the fitted peak positions directly influence the strain results.

Indexing: to determine the Miller indices (hkl) of each reflection and the number of reflecting grains. To do so, the fitted peaks are sorted by their integrated intensities and a subset N of the most intense reflections is first considered. The value of N can be carefully chosen to allow the algorithm to index a set of overlapping diffraction patterns coming from different illuminated grains.

First, a calibration procedure is used to obtain the instrument geometrical parameters (i.e. the position of the detector), including the pixel coordinates of the center channel, the distance of the detector to the sample, angular positions and angular tilts with respect to the incoming beam direction. Given the direction of the incident beam $\vec{k}_{in}$, the position of the detector and the characteristics of the detector (i.e. the size of the active area both in number of pixels and in mm), for each reflection in N with a precise position on the CCD, its corresponding reflected-beam direction $\vec{k}_{out}$ can be computed. Assuming elastic scattering, the direction of the experimental scattering vector can thus be calculated as $\vec{q}_{exp} = \vec{k}_{out} - \vec{k}_{in}$.

The angles between each pair of these scattering vectors corresponding to the N reflections are then computed and compared with a "reference" list of angles between each pair of the theoretical scattering vectors determined from the crystalline structure of interest and the energy bandpass of the polychromatic beam. Next, the process looks for angular matches

between triplets of reflections within an adjustable angular tolerance. For each matched triplet and their corresponding (hkl) indices, a trial orientation matrix is computed as well as a complete list of reflections that should be visible on the CCD for the considered energy bandpass, CCD dimensions and geometry. The best match is the triplet that is able to index the largest number of measured reflections.

Strain refinement: to obtain only the deviatoric strain tensor (unit-cell distortion) and the stress tensor for the illuminated grains, since white-beam patterns only allow the measurement of unit-cell distortions, not changes in the unit-cell volume (the dilatational or hydrostatic strain). Therefore, only the lattice parameters a, b, c can be refined. The transformation matrix t_{ij}, which defines the transformation from the unstrained crystal vectors to the distorted unit-cell vectors, can then be derived. By eliminating the rotational component from t_{ij}, the deviatoric strain tensor ε'_{ij} within the X-ray illuminated volume can be obtained. With the knowledge of the absolute lattice value of a single reflection, the complete strain tensor ε_{ij} (i.e. the sum of the deviatoric tensor and the dilatational tensor) can in principle be determined from a single white-beam diffraction image. The stress tensor can thus be derived from the strain tensor and the literature values of stiffness constants C_{ijkl} as $\delta_{ij} = C_{ijkl}\ \varepsilon_{kl}$.

3. Parallel and Stream Processing of Laue Microdiffractions

In this section, we present on the development of a system in IBM InfoSphere Streams for parallel and stream processing of synchrotron-based Laue microdiffraction images. Starting from the CCD image files, we organize the data involved in the Laue microdiffraction analysis as data streams and implement in IBM Streams a set of user defined operators (UDOPs) to work on the data streams and perform the functionalities of the processes introduced in Section 2. These user defined operators can be used to construct pipelines for processing multiple images in streaming mode. In the Appendix, Figure 4 shows a Streams live graph captured by the Streams studio when we use one pipeline with four processing elements for parallel peak fitting for processing images. Figure 5 illustrates four XRD processing pipelines running concurrently. In the following we report on our key implementation techniques.

3.1. Efficient Source Operator for Parsing Image Files

Synchrotron X-ray experiments usually generate thousands of image files. The large volume of files may have to be stored on external disks. Thus, reading these files from disk and parsing them may be a bottleneck. For reading an image file, we first find the data strip position and the byte count of the data and then read a whole block to save on disk seeking time. We then use bit-wise operations to convert a two-dimensional XRD image in binary format to a matrix of integer or double-precision energy intensity values. This makes our Source operator in Streams about ten times faster than the present SPE and TIFF image readers [9, 14].

3.2. Cache-efficient Background Filter

Except for the fine-pipelined parabolic background filter reported in [10], we implement in Streams an operator for estimating the background of CCD images based on Bruckner's method [4] and the Fortran procedure in XMAS. In this method, the image data needs to be traversed for many times. To obtain a cache-efficient implementation, we arrange the 2D image data in row-major layout in one consecutive array. Between the two calls to the 1D Bruckner procedure, we transpose the image so that the data for 1D Bruckner to operate on will always be aligned consecutively in order to minimize cache misses. We use the cache-oblivious algorithm introduced in [5] for matrix transpositions. As a result, our operator is about ten

times faster than the Fortran procedure in XMAS. This cache-efficient 2D Brückner Filter is presented in Algorithm 1.

Algorithm 1: **Cache-efficient 2D Brückner Filter**

Input: M, an array containing a matrix $(r \times c)$ of intensity values in row-major layout
Output: the background of M

```
1  S ← ResizeImage(M, r, c, m, n); // reduce the dimension of M to m × n
   // repeat the procedure for q times
2  for k from 1 to q do
3      for each row R_i in S do
           // replace each value in R_i by the average of its 2N neighbours, and iterate for t times
4          BackgroundFitBruckner1D(R_i, N, t);
       // replace the value of each point in S by the median of its W neighbors
5      MedianFilter2D(S, m, n, W);
6      Rotate S clockwise for 90 degrees into S_r;
7      for each row R_i in S_r do
           // replace each value in R_i by the average of its 2N neighbours, and iterate for t times
8          BackgroundFitBruckner1D(R_i, N, t);
       // replace the value of each point in S_r by the median of its W neighbours
9      MedianFilter2D(S_r, m, n, W);
10     Rotate S_r counter-clockwise for 90 degrees into S;
11 S_b ← ResizeImage(S, m, n, r, c); // enlarge the dimension of S to r × c
12 return S_b;
```

3.3. Memory-efficient Parallel Peak Fitting

For each significant reflection identified during blob searching, a non-linear least square solver is used to fit to a two-dimensional Gaussian, Lorentzian or Pearson VII function, so as to estimate the correct peak position. This is a computationally intensive procedure. The number of reflections in an image varies from tens to hundreds, depending on the properties of the sample. To gain performance for this procedure, we apply a parallel scheme that utilizes a dynamic scheduler to distribute the blobs for balanced parallel peak fitting.

This scheme is demonstrated in Figure 2. In the example, the blob searching operator takes a stream u of image data and identifies interesting blobs and then arrange the data associated with each blob (i.e. the coordinates of the center, the pixel values of the blob region) as a tuple in a stream *bl*. To prepare to be distributed for peak fitting, each blob tuple is labelled with a group ID to classify the blob tuples into groups. The largest group ID is the number of distributed peak fitting elements (F) available. Next a splitter is used to split the tuples in stream *bl* by Group ID, generating F number of sub-blob streams. Each sub-blob stream will be taken by a peak fitting process element to carry out the fitting operation and organize the result into a tuple in a sub-peak stream. All the sub-peak streams are bundled together to be sorted and then used in indexing process.

In order to avoid memory contention as well as to maintain good data locality, right after a blob with pixel b as its center is found, we copy an area R of pixel values from the original image, as shown in Figure 3. The area R that we define is twice as large as the area R_b (normally a 10×10 matrix) which has b as its center and is needed for peak fitting. We associate the data set R to the blob of b and distribute this blob for parallel peak fitting. In peak fitting, the

necessary area R_b is determined from area R by coordinate mapping. A larger area like R is required because the center of a reflection can be different (i.e. slightly shifted) from b by peak fitting. The new center, say p, will define an area R_p of dimension 10×10 in the original image. The area R_p is used to compute an integrated intensity which will be used to classify (i.e. rank the priority of) a reflection for later indexing. We determine the area R_p from area R again by coordinate mapping, with no reference to the original image. In this way, we avoid the many times of references to the original image, a potential source of memory contention for concurrent execution. Good data locality is obtained by associating each peak fitting with a small working data set like R.

In addition, we align the data in the peak fitting results in one flat array, where the data associated with each peak is a record with the integrated intensity value as its key. We create two routines for sorting records of data. One implements insertion-sort algorithm for sorting small number of records and the other one uses merge-sort for sorting relatively large number of records. In these sorting routines specialized for records of data, a record is moved as a block to achieve efficient memory operations.

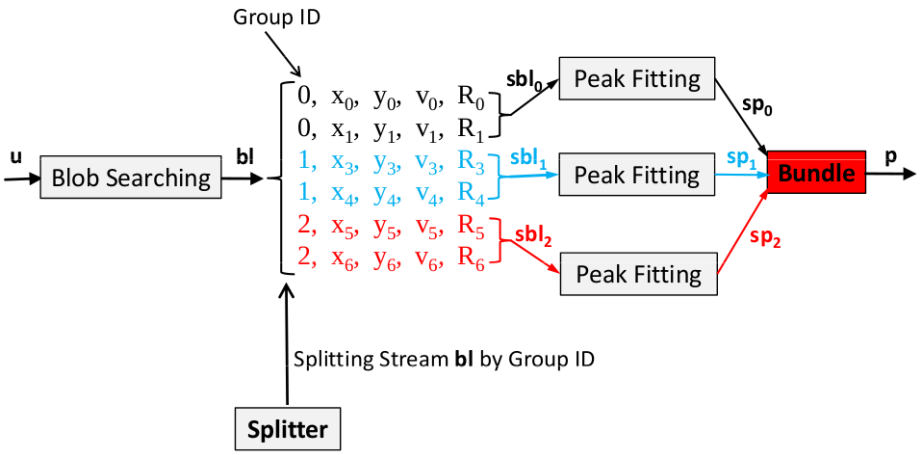

Figure 2: Dynamically scheduling blobs for parallel peak fitting

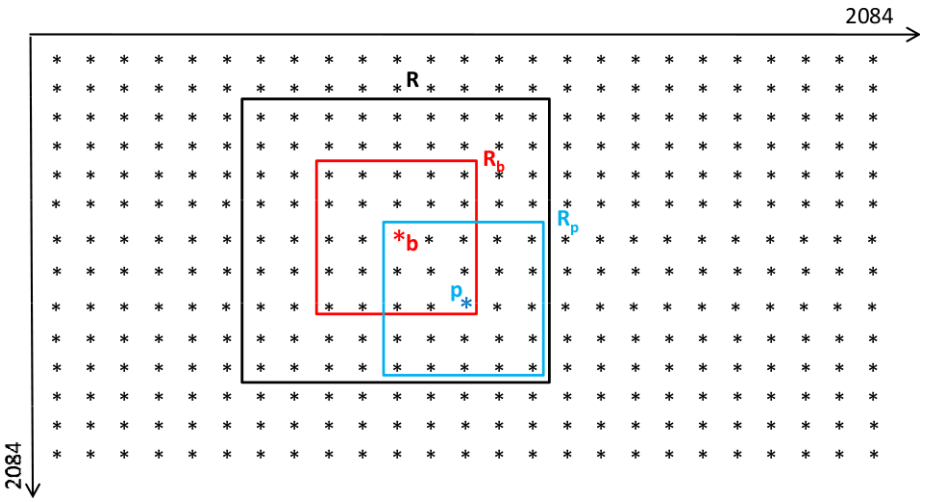

Figure 3: Efficient memory management for parallel peak fitting

3.4. Sharing Common Parameter Values in Indexing and Strain Analysis

For indexing and strain analysis, we eliminate the duplicate computations and data storage occurring in the sequential program. In InfoSphere Streams, we are able to generate streams of data which can later be shared by multiple processing elements. We take advantage of this feature and create a special operator to parse the experimental parameters, compute and then only keep the final derived parameter values necessary for indexing and strain analysis as a data stream.

More precisely, we design a Source operator for parsing the common parameter values (beamline, sample and detector geometrical parameters, unit-cell parameters, space group data and stiffness constants). Based on these inputs, three commonly used rotation matrices for conversion between different coordinate systems (beamline (lab), detector and sample) and two pairs of transformation matrices for conversion between lattice space and reciprocal space, between lattice space and direct (lab system) space are generated. Together with the space group data and stiffness constants, they from a stream which can be shared later for indexing and strain analysis for all the XRD images.

In addition, using this stream of parameter data, we create an operator for generating the theoretical reflection list which can be used for the indexing of all XRD images. We also observe that we can use matrix transposition instead of calculating the inverse for generating the inverse of a rotation matrix by Euler angles as composition of extrinsic rotations. For processing $10,000$ images, this can save 1.5 million number of multiplications of double precision values.

3.5. Benchmarks

Our benchmark system is an Intel Core2 Quad CPU Q9550 (2.83 GHz, 8 GB RAM and 6 MB L2 cache) running with Red Hat Enterprise Linux 5.4 operating system. It supports InfoSphere Streams Version 1.2. Our program can be configured to generate multiple pipelines to process multiple images concurrently in streaming mode. It takes about 2.5 seconds to process a single image of dimension size 2084×2084 in SPE format using one pipeline with four processing elements for parallel peak fitting, which is about 10 times faster than the present best C program. On this quad-core desktop, super-linear speedup is obtained for processing four images by four pipelines.

4. Concluding Remarks

We present a parallel and stream processing system on multicore processors to analyze X-ray microdiffraction images for the mapping of crystallographic orientation and deviatoric strains. It is the first streaming application in the field of XRD data analysis. Our approach and results show that stream processing is an effective model for efficiently using multicore processors for XRD image data analysis. With the support of IBM InfoSphere Streams platform, the set of user-defined operators that we have created for XRD image preprocessing and Laue pattern analysis could be generalized as part of a library of streaming operators for processing images derived from a variety of detectors. Our work-in-progress includes the configuration and deployment of this kernel to large systems with many cores for processing large set of XRD images efficiently.

5. Acknowledgments

We would like to acknowledge the support for this project from CANARIE and for the networking infrastructure that they provide to enable us to even begin to think about streaming image data. We would also like to thank all those involved in the Science Studio project and the ANISE project.

References

[1] A. A. MacDowell, R. S. Celestre1, N. Tamura, R. Spolenak, B. Valek, W. L. Brown, J. C. Bravman, H. A. Padmore, B. W. Batterman and J. R. Patel. Submicron X-ray Diffraction, http://escholarship.org/uc/item/2f47k34b, Lawrence Berkeley National Laboratory, USA, 2000.

[2] Advanced Light Source. X-ray Microdiffraction Analysis Software (XMAS), http://xraysweb.lbl.gov/microdif/user_resources.htm, Lawrence Berkeley National Laboratory, USA, 2010.

[3] Advanced Photon Source. 3D X-ray Microdiffraction Analysis Software Package, http://www.aps.anl.gov/Sectors/33_34/microdiff/downloads/, Argonne National Laboratory, USA, 2009.

[4] Sergio Brückner. Estimation of the Background in Powder Diffraction Patterns through a Robust Smoothing Procedure. *Journal of Applied Crystallography*, 33(3 Part 2):977–979, Jun 2000.

[5] M. Frigo, C. E. Leiserson, H. Prokop, and S. Ramachandran. Cache-Oblivious Algorithms. In *FOCS*, 1999.

[6] IBM. IBM InfoSphere Streams, http://www-01.ibm.com/software/data/infosphere/streams/, 2010.

[7] J. Chung and G. Ice. Automated Indexing for Texture and Strain Measurement with Broad-band Pass X-ray Micro-beams. In *Journal of Applied Physics*, number No. 9, Vol. 86, 1999.

[8] J. Qin and M. A. Bauer. Accelerated Synchrotron X-ray Diffraction Data Analysis on a Heterogeneous High Performance Computing System. In *J. Phys.: Conf. Ser.*, number 256 012007, 2010.

[9] Jon Tischler. WinView, Oak Ridge National Lab, TischlerJZ@ornl.gov, 2010.

[10] M. A. Bauer, A. Biem, S. McIntyre and Y. Xie. A Pipelining Implementation for Parsing X-ray Diffraction Source Data and Removing the Background Noise. In *J. Phys.: Conf. Ser.*, number 256 012017, 2010.

[11] N. Tamura and A. A. MacDowell and R. Spolenak and B. C. Valek and J. C. Bravman and W. L. Brown and R. S. Celestre and H. A. Padmore and B. W. Batterman and J. R. Patel. Scanning X-ray Microdiffraction with Submicrometer White Beam for Strain/Stress and Orientation Mapping in Thin Films. In *J. Synchrotron Rad.*, number 10, pages 137–143, 2003.

[12] N. Tamura and J.-S. Chung and G. E. Ice and B. C. Larson and J. D. Budai and J. Z. Tischler and M. Yoon. Strain and Texture in Al-Interconnect Wires Measured by X-ray Microbeam Diffraction. In *Mat. Res. Soc. Symp.*, number 563, pages 175–180, 1999.

[13] R. I. Barabash, G. E. Ice, N. Tamura, B. C. Valek, J. C. Bravman, R. Spolenak and J. R. Patel. Quantitative Characterization of Electromigration-induced Plastic Deformation in Al(0.5 wt%Cu) Interconnect. *Microelectron Engineering*, (75:1):24–30, 2004.

[14] Sam Leffler. libtiff 3.8.2, ftp://ftp.remotesensing.org/pub/libtiff/, Silicon Graphics, Inc., 2004.

[15] ANISE Team. Active Network Interchange for Scientific Experimentation Project, http://www.anise-project.com/, University of Western Ontario, London ON, Canada, 2010.

Appendix

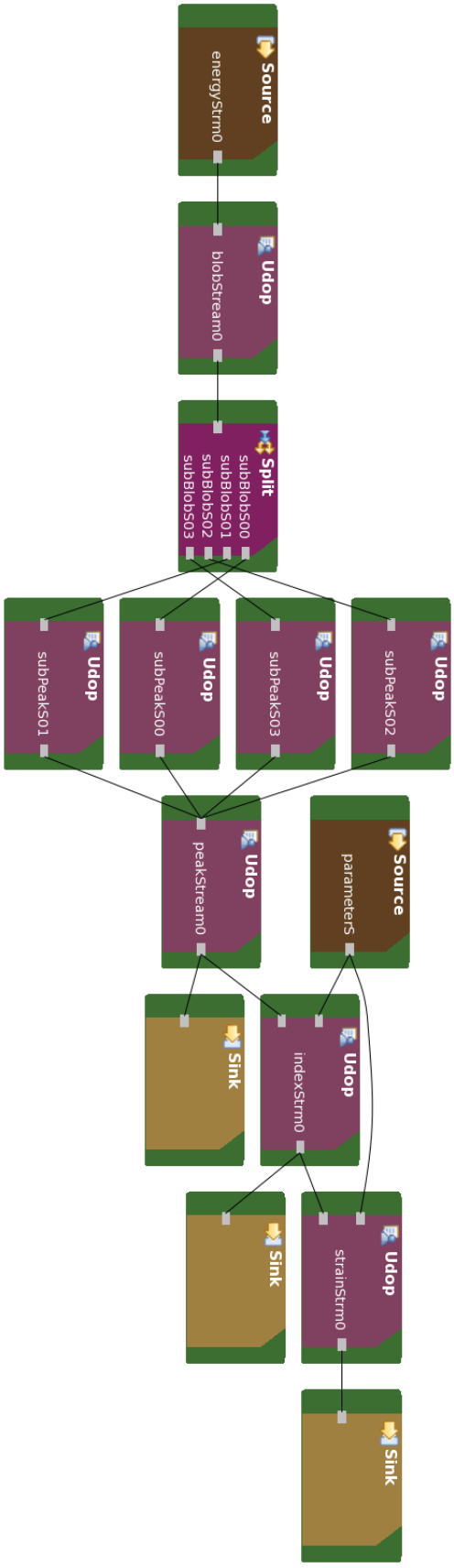

Figure 4: Streams live graph: one pipeline with 4 processing elements for parallel peak fitting

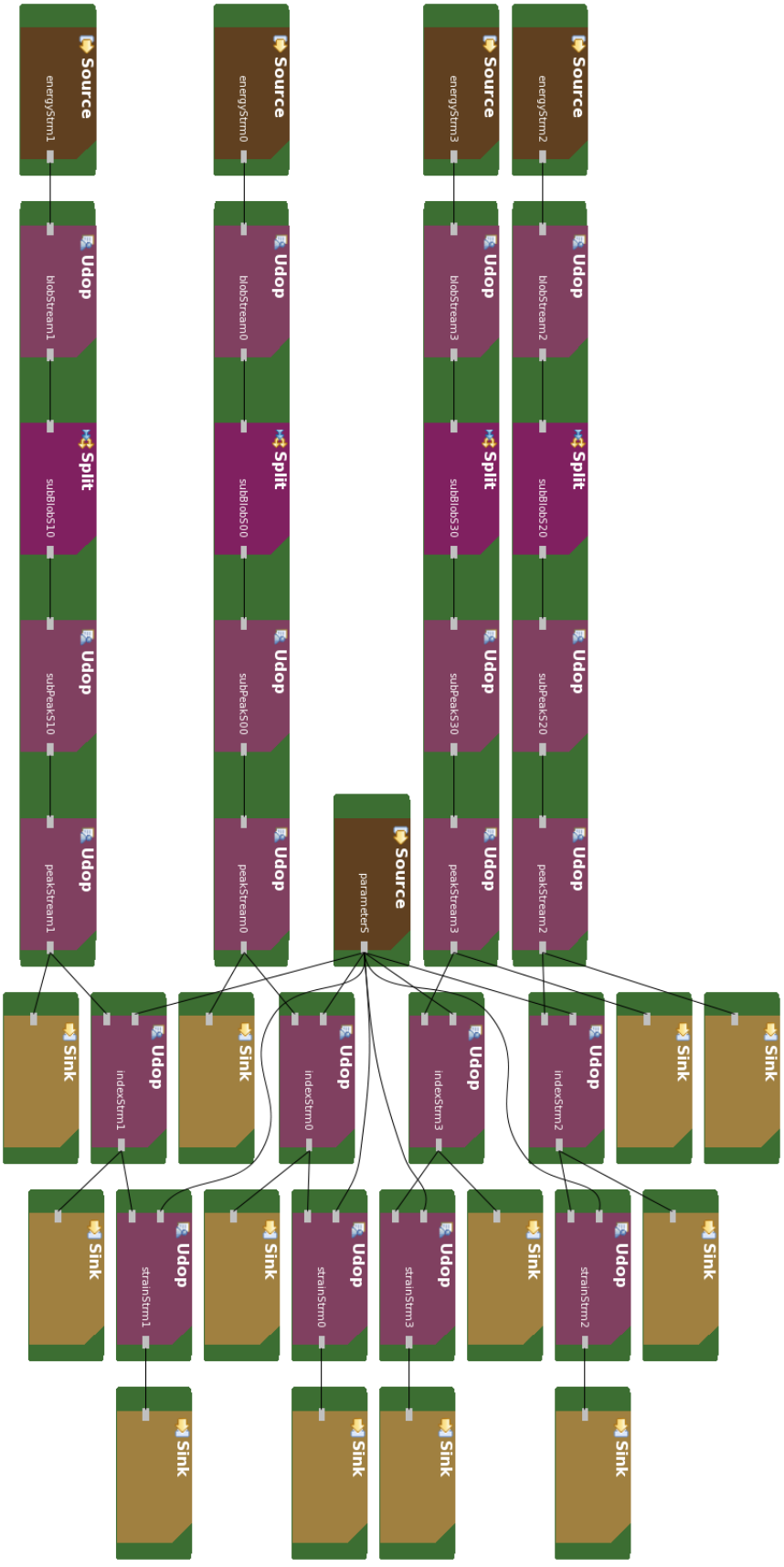

Figure 5: Four XRD processing pipelines running concurrently (the operators for blob searching and peak fitting are fused to minimize transfer latencies)