Accelerating Microstructural Analytics with Dask for Volumetric X-ray Images

Daniela Ushizima
Computational Research Division, LBNL
Berkeley Institute for Data Science, UC Berkeley
Berkeley, CA, USA 94720
dushizima@lbl.gov

Matthew McCormick
Kitware Inc.
Insight Toolkit
Carrboro, NC, USA 27510
matt.mccormick@kitware.com

Dilworth Parkinson
Advanced Light Source, LBNL
Beamline 8.3.2
Berkeley, CA, USA 94720
dyparkinson@lbl.gov

***Abstract*—While X-ray microtomography has become indispensable in 3D inspections of materials, efficient processing of such volumetric datasets continues to be a challenge. This paper describes a computational environment for HPC to facilitate parallelization of algorithms in computer vision and machine learning needed for microstructure characterization and interpretation. The contribution is to accelerate microstructural analytics by employing Dask high-level parallel abstractions, which scales Numpy workflows to enable multi-dimensional image analysis of diverse specimens. We illustrate our results using an example from materials sciences, emphasizing the benefits of parallel execution of image-dependent tasks. Preliminary results show that the proposed environment configuration and scientific software stack deployed using JupyterLab at NERSC Cori enables near-real time analyses of complex, high-resolution experiments.**

***Index Terms*—MicroCT analysis, Python, Dask, HPC**

1. Introduction

From industry to national laboratories, X-ray imaging has become fundamental to measure the function and resilience of new materials and for probing dynamic properties. The growth of X-ray brilliance and extremely quick snapshots allied to advances in HPC and machine learning create new opportunities to streamline the description of materials structures as part of the design of new compounds. However, the volumetric image analysis of these specimens at scale has necessitated several research efforts [1], [2], [3], [4] in automation that combine computational and experimental methods.

The above application is representative of a major opportunity that stems from the utilization of advanced computing at experimental facilities, namely coupling of increasing data rate experiments to new data science algorithms [5], [6], [7] in support of quantitative image analysis to automatically drive the scientific discovery, and full utilization of data acquire at high cost.

Recent implementations of deep learning models applied to image representation and structural fingerprints have made sample sorting and ranking possible [8], [9], which allows automated identification of special materials configurations from million-sized databases. These complex networks recognize events from data gathered in both experimental and simulation regimes. While such methods successfully bypass hand-engineered features [10], [11], [12], their full extension to three-dimensional imagery seldom meets standards that are comparable to the superior human labeling. Unfortunately manual curation is practically impossible for entire 3D datasets from X-ray imaging experiments, particularly when a single sample is often imaged many times, and each file can contain several gigabytes.

In order to inspect material structure using synchrotron-generated X-ray microtomography (microCT) at such large scales, this paper describes new Python-based computational schemes based on new implementations for image representation (e.g. multi-resolution pyramid), enhancing (e.g. non-linear filters, 3D Hessian-based tubeness), volume partitioning (e.g. hysteresis) [13], and measurements (e.g. counts, detection accuracy); altogether, these techniques address key, workhorse tasks in image segmentation, stereological analysis, and enumeration of particles within high-resolution microtomography imagery.

The contributions of this investigation are: (a) interactive and fully-automated processing Python code for volumetric data that stem from physical experiments and simulation; (b) the elaboration of Jupyter notebooks to enable domain scientists to access created Python codes for the analysis and constraint the parameter space, particularly when prior knowledge might be available from experimental settings; and (c) the reproducibility of experiments by recognizing the importance of open-source codes to fully extract information from scientific images coming from advanced instruments. This paper also includes scripts for visual analysis and interaction with extracted 3D geometries.

The paper is organized as follows: Sec. 2 introduces a use-case to demonstrate our advancements on inspection of hierarchical materials that consist of many individual strands, bundled within a matrix to achieve high-strength mechanical properties and durability. Sec. 3 describes the different methods composing a complex environment that allows characterization of microCT at scale using HPC. Results on microCT dense phase segmentation and fiber detection appear in Sec. 4 with concluding remarks in Sec. 5. Finally, Sec. 6 discusses new opportunities for improve-

ments and future work on microstructural analytics.

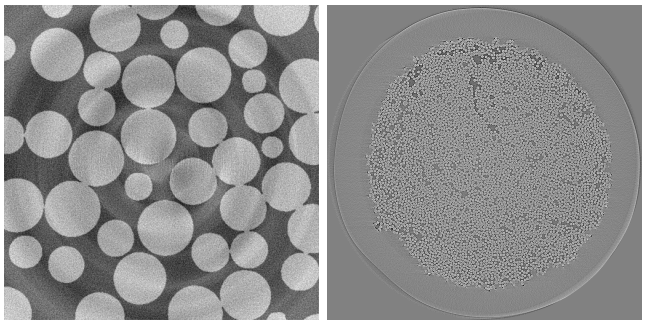


Figure 1: Raw slice from 3D stacks: (a) synthetic bead pack; (b) CMC reinforced with fibers.

2. Related Work

The search for the next generation of ceramic matrix composites (CMCs) aims to find materials that withstand higher temperatures and deformations for the construction of aerospace vehicles [1], [2], with application in military and commercial jet engines and industrial turbines. CMCs can operate at temperatures up to 500^oF higher than nickel super-alloys, and are a third of the weight, which is likely to lead to lighter aircrafts and engines that run at higher thrust more efficiently. As part of industrial power turbines, CMCs have the potential to reduce emissions and the cost of electricity. [14]. To analyze and characterize the underlying three-dimensional microstructure of CMC samples in a non-destructive fashion, a set of microCT volumes are often acquired.

Larson et al. [3], [4] investigated new manufacturing processes using polymer-to-ceramic conversion, and impregnation and curing of a pre-ceramic polymer into unidirectional fiber beds. Here, microCT was used to investigate the CMC microstructure evolution during matrix impregnation, which is key for the CMC reinforcement.

Each sample consists of 1012 tows of Hi-Nicalon silicon carbide (SiC) fibers, with 500 fibers per tow with the average fiber radius around 6.4 $\pm$ 0.9 μm (standard deviation), with diameters ranging from 13 to 20 pixels, as measured from the microCT images. The fiber beds have widths of approximately $1.5mm$, containing 5,000-6,200 fibers per stack.

Previous work on fiber detection from CMC [1], [2], [3], [4] introduced semi-supervised techniques to separate the fibers within the fiber beds, but the computational methods are opaque and still unavailable. Nevertheless, the data repository from [3], [4] has been invaluable [15] for algorithm development and reproducibility, therefore they are considered in the experiments of this paper as described in Sec. 4.

The microCT images acquired from such fiber beds are available at the Materials Data Facility [16], in a data collection of $\approx$ 6 TB in total, with each file containing about 60 GB and more than 14 billion voxels. Data acquisition took place at the Advanced Light Source (ALS) at the Lawrence Berkeley National Laboratory. This paper describes results using one of the publicly available microCT volumes in this collection, as well as through using a synthetic dataset with a simulated monodispersive bead-packing sample, contaminated by ringing artifacts, which are inherent in computed-tomography images (e.g. due to projection and reconstruction [17]).

Other works describing promising algorithms for fiber detection include [18], [19], however these approaches showed scalability issues when handling high resolution datasets. The methods in this paper avoid these scalability issues, and allow the processing of very large datasets and complex structures.

3. Methods

This section describes the different components of the software stack to enable the working environment illustrated in Fig. 2, including many components of the software stack made availabe at NERSC by default [20]. NERSC Cori is a Cray XC40 with a peak performance of about 30 petaflops, comprised of 2,388 Intel Xeon "Haswell" processor nodes, 9,688 Intel Xeon Phi "Knight's Landing" (KNL) nodes. This paper and corresponding code repository describe our ongoing efforts on deploying these analytical capabilities at NERSC Cori, although preliminary tests indicate portability to Amazon Web Services (AWS) as well.

3.1. Image Representation with `xarray` & `zarr`

By using Xarray [21] data structures, and Zarr [22] for serializing Xarray content, one can transform data into pyramidal representations for quick multi-resolution access and processing with modules compatible with n-dimensional scientific data.

Zarr is a format for storage of chunked, n-dimensional arrays, compressed with next-generation codecs, in a hierarchical organization along with their metadata. Efficient parallel processing of image regions facilitated by 1) parallel read and write of compressed sub-regions, 2) quick queries, consolidated dataset metadata coupled with lazy loading of bulk array data, 3) compatibility with modern network-based blob storage backends, and 4) interfaces with scientific Python libraries like NumPy, Xarray, and Dask along with other scientific computing tools written in languages such as C++ or JavaScript.

Zarr is a well-supported storage format in Xarray, a library for labeled arrays and datasets in Python. Xarray, originally developed for the geosciences, enables labeling the dimensions of arrays, e.g. identifying the `x`, `y`, and `z` dimensions. Xarray also preserves metadata by naming of data arrays in datasets semantically, e.g. by an experiment identifier, and retaining array spatial information through 1-d coordinate labels, which propagate well through NumPy-like slicing operations.

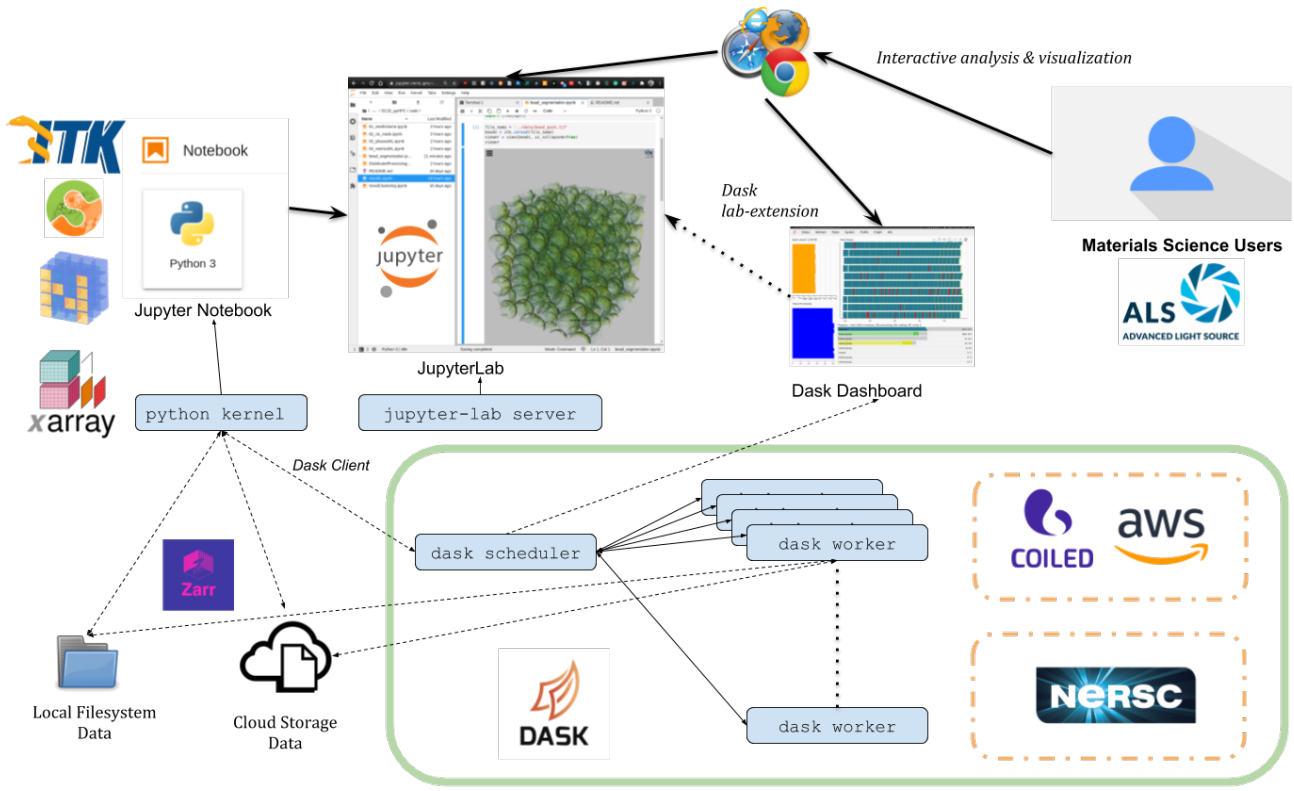


Figure 2: *Scalable, python-based, 3D microstructure interactive image analysis and visualization system architecture.* A material image scientist can interactively analyze and visualize their data through the JupyterLab browser interface. Additionally, users can monitor distributed computation in the Dask dashboard visualization in a separate browser tab. Computational processes are shown in blue. The JupyterLab server may be running locally or remotely. Computation and web-native visualizations are executed in a Jupyter notebook that runs a Python kernel. This Python kernels run standard scientific Python packages, e.g. NumPy, scikit-image, ITK, and accesses data on the local filesystem or in cloud via Zarr. To scale computation, a Dask distributed cluster can optionally be connected via a Dask client. This client sends tasks, composed of Python functions and data, to Dask workers via the Dask scheduler. The cluster can be backed by the NERSC Cori cluster, running over MPI, or a dynamic cluster on the AWS cloud managed by Coiled.

Our current work focuses on scaling key image processing routines to be amenable to Xarray-Zarr, as illustrated in a preliminary experiment [23] to demonstrate the efficiency in working with such framework. The analysis of specimens from materials science often requires exploration of multi-scale information; this chunked, compressed multi-scale pyramid storage permits parallel analysis and visualization with Dask [24], discussed in the Sec. 3.3.

3.2. Enhancement and Partitioning with `itk`

The Insight Toolkit (ITK) offers a rich assortment of 2D and 3D image analysis algorithms written in C++. Recently, many of the libraries have been Python wrapped to be easily callable and interfaced with Numpy arrays, making ITK extensible via plug-ins, macros and Python scripts. ITK has also evolved in a way that it is easier to used in conjunction with other image processing libraries, such as `scikit-image` [25] and `dask-image` [26]. Among the several functions, this paper focuses on the following:

- Preprocessing filters, e.g. an unsharp mask filter to enhance edges.
- Mathematical morphology filters that enforce shape-based constraints on segmented objects.
- Fast distance map filters, applied in this case with a morphological watershed segmentation filter to identify objects by both shape and connectivity.
- Hessian-based objectness features, useful for identifying blob-like, tube-like, and plate-like structures in volumes.

Particularly, the Hessian-based tubeness enhances fiber-like structures while employing a well-defined kernel and fiber classification function, which suppresses background noise and increases contrast of fading fibers.

3.3. Parallelizing with `dask`

Dask presents three parallel collections that can store data that is larger than RAM, namely Dataframes, Bags and Arrays. For example, Dask Arrays enable data partitioned between RAM and a hard disk as well distributed across multiple nodes in a cluster. We use Dask Arrays for Numpy workflows to enable multi-dimensional image analysis of microCT volumes.

Several libraries, such as Joblib, Multiprocessing, IPython Parallel, Concurrent.futures, and Luigi, provide parallel execution. However, Dask schedulers present unique characteristics, e.g., it is possible to specify the entire graph as a Python dictionary rather than using a specialized API. This means that the schedulers take a task graph, which is a dictionary of tuples of functions, and a list of desired keys from that graph. Another advantage is that there is a variety of schedulers ranging from single machine, single core to threaded, multiprocessing, and distributed. The Dask single-machine schedulers have logic to execute the graph in a way that minimizes memory footprint. As a result, our algorithms can be quickly extended on smaller datasets with a laptop or workstation. Then, the same code can scale to extremely large datasets by running Dask over a HPC cluster backed by MPI, a cluster with a scheduler like SLURM, or a dynamically-generated, cloud service running a Kubernetes cluster.

3.4. HPC Interaction with `itkwidgets`

`itkwidgets` provides interactive Jupyter widgets to visualize images, point sets, and meshes in 3D or 2D. We include rendering as one of the capabilities of this framework as a driver to improve user experiments, fast hypotheses testing to streamline the path to final solution, visual feedback and code debugging. It also facilitates the generation of immersive visualizations by recording flight paths through 3D renderings of multidimensional images from tomography datasets.

4. Results

This section describes details to configure the system to run the experiments as well as the results using the proposed environment to process microCT data. In an effort to make this work as reproducible as possible, information about software packages, code for figure creation from public data are available at: `https://github.com/dani-lbnl/SC20_pyHPC`

4.1. Experimental Setup

The NERSC Cori software stack has considerably evolved to accommodate intensive analyses of experimental and observational science data. For example, it has made Shifter [20] available to users to allow containerized environments (e.g. using Docker) to run on a supercomputer. In order to access the environment illustrated in Fig. 2, the user needs to log into a terminal on Cori, either via JupyterLab or SSH. We supply the Dask software infrastructure by calling `bash ./install-override-jupyter.sh` which will download the Docker software environment into Shifter and configure the NERSC JupyterHub to load the same environment for JupyterLab from a `HOME` directory configuration. This default Python 3 Jupyter kernel computational environment will be running in a Docker-Shifter image with Python, MPI, Dask, dask-image, itk, etc. installed and configured for NERSC. The same computing environment is also able to run:

- The JupyterLab server and client with custom JupyterLab extensions;
- The Jupyter kernel Python process;
- The Dask dashboard;
- The Dask scheduler;
- The Dask workers;

A software environment with specific software packages and versions is critical for successful distributed execution. We achieve a consistent software environment with the Conda package manager, combined to the Docker-Shifter containerization.

4.2. Overall Speed

Fig. 3 shows the several transformations steps, from the raw input to the final segmentation of the beads, which are colored-based on each calculated connected component. Since the synthetic data is small (200^3 voxels), the computation takes under 4 seconds to complete for 40 blocks, but there is a lot of overhead in sending the data over the network, and it is actually slightly slower to process the chucked data. On the other hand, the data-intensive, 3D Hessian computation on a 650 megavoxel fiber volume completes in under 2 minutes with 100 workers, with 200^3 chunk-sizes. Fig. 4 shows a 3D render of the stack, with a virtual cut to enable internal observation (left), a fiber chunk after Hessian-based tubeness calculation (center), and the rough fiber segmentation for later fiber detection.

4.3. Image Segmentation Metrics

We measure the performance of the fiber segmentation pipeline (Fig. 4) comparing the automated fiber detection with a manually labeled slice as in [19], which are shown in Fig. 5. The fibers detected by hand (mouse clicks) are considered the "true" objects while the automatically detected ones are the "predicted" ones. The detected fiber centers determine the center of a circle with diameter of 10 pixels, therefore narrower than the average fiber in order to avoid overlaps.

While Fig. 4 emphasizes qualitative evaluation, we quantify the difference between the manual and automated (Fig. 5) in terms of several parameters:

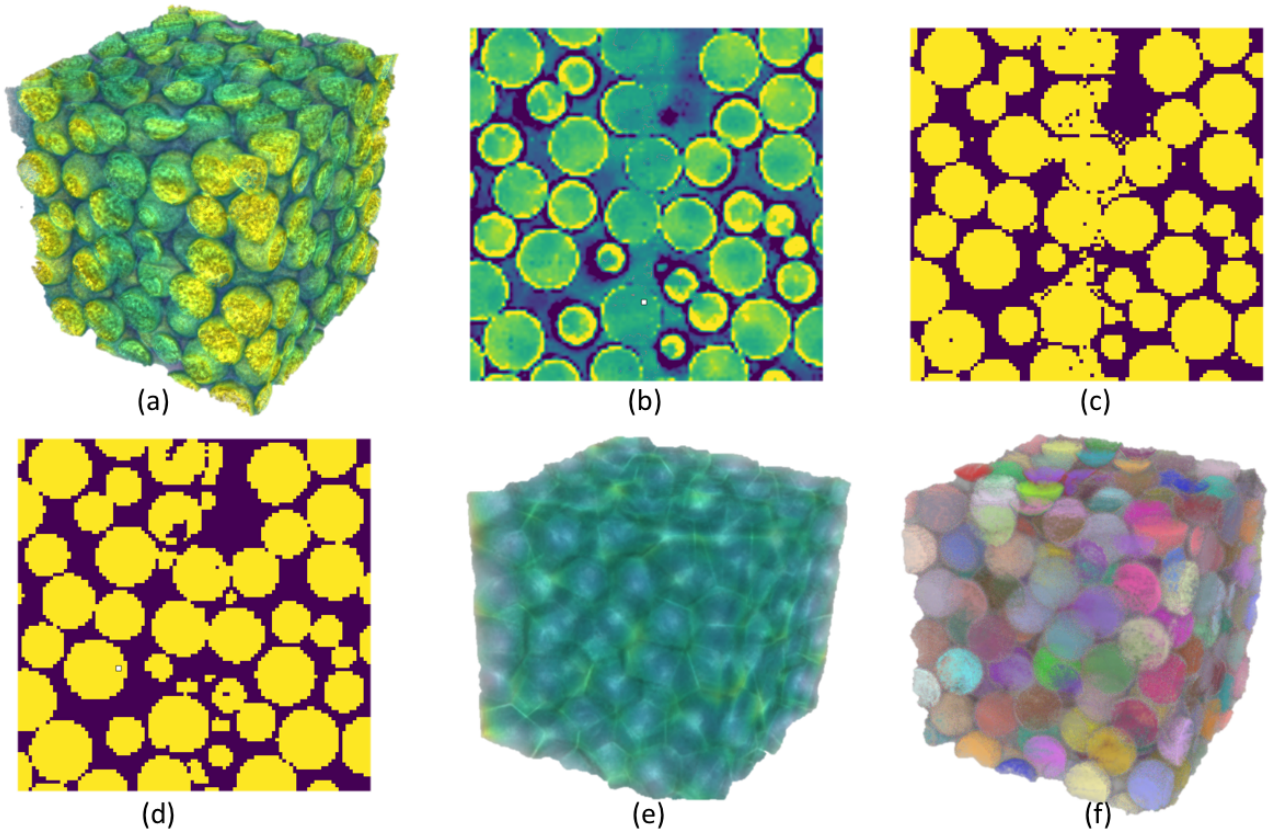


Figure 3: Bead segmentation pipeline: (a) raw data; (b) filtering; (c) rough segmentation; (d) postprocessing; (e) 3D watershed; (f) connected components.

- Dice coefficient= 0.7424,
- Sensitivity = 0.7962,
- Specificity = 0.929,
- Accuracy = 0.9065,
- AUC=0.8626,
- Precision=0.6955,
- F-score=0.7424.

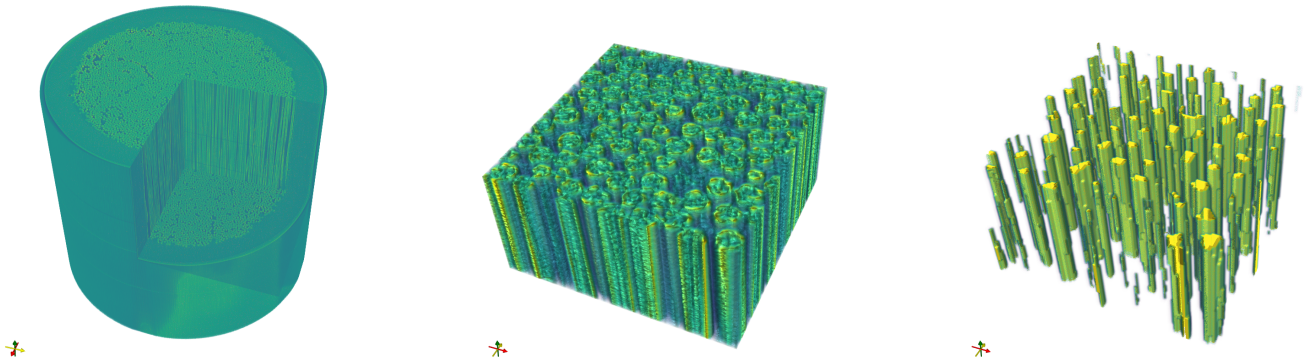
Performance measurements across computational environments are compared in Fig. 6. Fiber identification on a standard laptop takes 13 minutes. The same processing pipeline can be executed on NERSC Cori in 0.5 minutes and a Coiled.io cluster in 2 minutes.

These performance changes can be attributed to two significant differences in the standard mode of operation of these different computational environments. First, the most processors were available in NERSC Cori, where we allocated a cluster of 100 workers with 2 CPU's per worker. On the Coiled AWS cluster, 100 workers were allocated with 1 CPU per worker. Two workers were allocated on the laptop, and each worker had access to 6 cores. Second, I/O capabilities varied between the environments: for Cori, we stored the data on the Lustre file system designed for high performance temporary storage of large files. For the Coiled and laptop analysis, we accessed the data through storage on an HTTP Content Distribution Network (CDN).

More notable, however, are the similarities across the computational environments. High-memory resources were not required – the limited memory capacities of the laptop and Coiled workers were sufficient for the large dataset. The user experience, i.e. interactive Jupyter workspaces accessed through the browser on the laptop, was exactly the same. And, most importantly, the same software was used without modification; to change the computational backend, the Dask Distributed backend was simply changed to a distributed cluster resource, which resulted in a 25X acceleration when processing the larger dataset.

5. Conclusion

The ability to use remote computations and visualizations has never been so important, particularly during the current COVID-19 pandemic. While processing simulated bead-packing experiments such as those in Fig. 3 can be easily performed using a decent laptop, image processing of high-resolution high-fidelity microCT data often demands HPC computation. For a long time, switching from one system to another meant completely rewriting the analysis code. This paper proposes a scheme that allows laptop computation to HPC to cloud cluster computation with low



(a) (b) (c)

Figure 4: Fiber segmentation pipeline: (a) raw data with a virtual cut, (b) chunck after Hessian-based tubeness, (c) rough fiber segmentation.

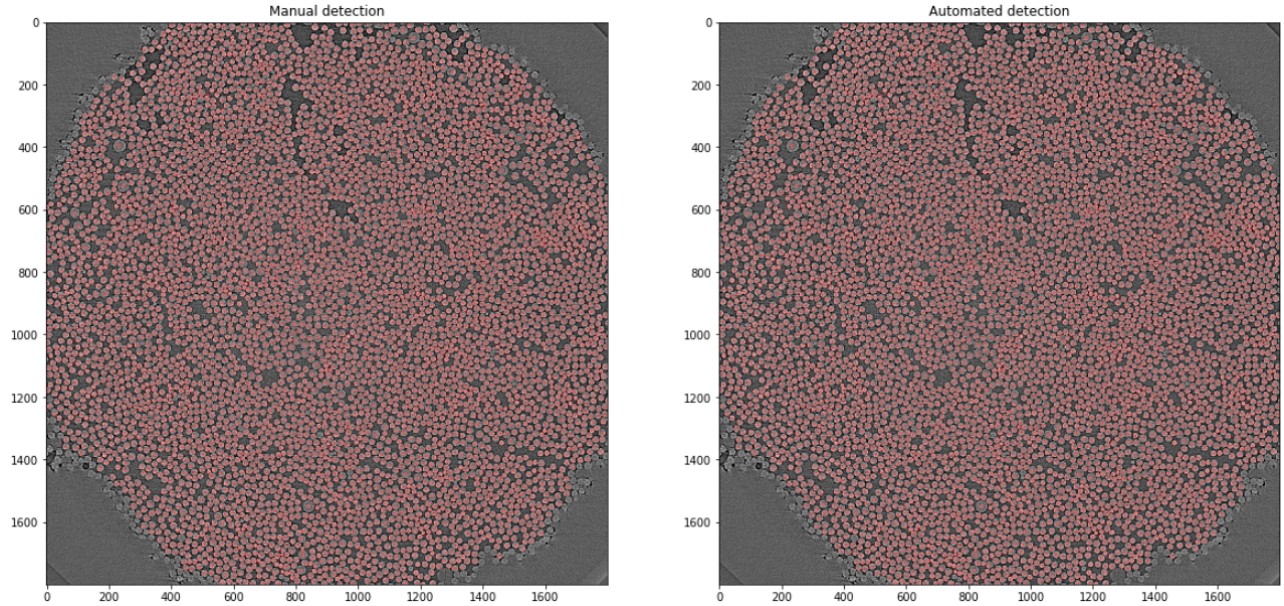


Figure 5: Fiber detection overlayed on raw slice: manual detection using mouse clicks (left) and automated detection (right)

technical barriers to the standard material scientist. Now, the user can choose different resolutions to make the computation more amenable for quick evaluation of collected data, e.g., choosing more compact data from the pyramidal representation or using sophisticated HPC infrastructure, such as NERSC, for analysis of the whole data.

Fig. 5 shows that the current fiber automated detection algorithm (few minutes) often agrees with the manual detection (few hours), and both of them include mistakes, such as missing fibers or fake fibers. The metrics in Sec. 4.3 treats the manual segmentation as the ground-truth, so the low precision does not mean failure in prediction over true objects only, but also the potential inadequacy of the ground-truth and the comparison method. Beyond validity, more automated computational methods should be considered. Here we use the center of the detected fiber to determine a circle proportional to the fiber diameter; since the diameter is narrower than the real fiber, low sensitivity was expected because a small variation on the mouse click will lead to different intersection between its circle and an accurately predicted fiber.

6. Discussion and Future Work

This paper proposes a comprehensive pathway for scaling key modules in computer vision for analyzing 3D data such as microCT with an impact on applications in materials science.

We exercised our scientific workflow using both a simulated and an observed microCT. These results are preliminary, and a more complete examination, currently underway, considers many more datasets. Our main goal here is

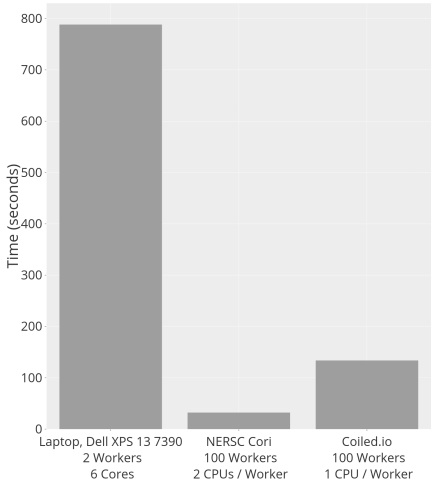


Figure 6: Fiber segmentation performance (lower is better) across Dask clients ranging from a local laptop, HPC cluster to a cloud cluster.

to emphasize the relevance of the proposed computational environment to a broad variety of users, including its potential to be applied to many other scientific domains. This approach also has excellent potential to facilitate training and education in HPC, scientific computing, and computer vision.

Acknowledgements

The authors are grateful for the support of the U.S. Department of Energy, Office of Science, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This material is based upon work supported by the U.S. Department of Energy, Office of Science, Office of Advanced Scientific Computing Research, Center for Advanced Mathematics for Energy Research Applications (CAMERA) program. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

References

[1] H. A. Bale, A. Haboub, A. A. Macdowell, J. R. Nasiatka, D. Y. Parkinson, B. N. Cox, D. B. Marshall, and R. O. Ritchie, *Nat. Mater.*, vol. 12, pp. 40–46, 2012.

[2] B. N. Cox, H. A. Bale, M. Blacklock, M. N. T. Fast, V. Rajan, R. Rinaldi, R. O. Ritchie, M. Rossol, J. Shaw, Q. D. Yang, F. Zok, and D. B. Marshall, "Stochastic virtual test systems for ceramic matrix composites," *Annual Review of Materials Research*, vol. 44, pp. 479–529, 2014.

[3] N. Larson and F. Zok, "In-situ 3d visualization of composite microstructure during polymer-to-ceramic conversion," *Acta Materialia, Publishing: Elsevier*, pp. 579–589, 2018.

[4] C. C. Larson, N. and Z. F., "X-ray computed tomography of microstructure evolution during matrix impregnation and curing in 2 unidirectional fiber beds," *Composites. Part A, Applied Science and Manufacturing, Publishing: Elsevier*, pp. 243–259, 2019.

[5] M. Emerson, V. Dahl, K. Conradsen, L. Mikkelsen, and A. Dahl, "Statistical validation of individual fibre segmentation from tomographs and microscopy," *Composites Science and Technology*, p. 208215, 2018.

[6] C. Harris, P. O'Leary, M. Grauer, A. Chaudhary, C. Kotfila, and R. O'Bara, "Dynamic provisioning and execution of hpc workflows using python."

[7] K. Odziomek, D. Ushizima, P. Oberbek, K. jan Kurzydowski, T. Puzyn, and M. Haranczyk, "Scanning electron microscopy image representativeness: morphological data on nanoparticles," *Journal of Microscopy*, vol. 265, no. 1, pp. 34–50, 2016.

[8] D. Ushizima, H. A. Bale, W. Bethel, P. Ercius, B. Helms, H. Krishnam, L. Grinberg, M. Haranczyk, M. A. A. Macdowell, K. Odziomek, D. Y. Parkinson, T. Perciano, R. Ritchie, and C. Yang, "IDEAL: Images across Domains, Experiments, Algorithms and Learning," *Journal of Minerals, Metals and Materials*, 2016.

[9] Araujo, Silva, Medeiros, Parkinson, Hexemer, Carneiro, and Ushizima, "Reverse image search for scientific data within and beyond the visible spectrum," *Expert Systems with Applications*, vol. 109, pp. 35–48, Nov 2018.

[10] Silva, Araujo, Rezende, Oliveira, Medeiros, Veras, and Ushizima, "Searching for cell signatures in multidimensional feature spaces," *International Journal of Biomedical Engineering and Technology*, 2018.

[11] Ke, Brewster, Yu, Yang, Ushizima, and Sauter, "A convolutional neural network-based screening tool for x-ray serial crystallography," *Journal of Synchrotron Radiation*, 2018.

[12] S. Liu, C. N. Melton, S. Venkatakrishnan, R. J. Pandolfi, G. Freychet, D. Kumar, H. Tang, A. Hexemer, and D. M. Ushizima, "Convolutional neural networks for grazing incidence x-ray scattering patterns: thin film structure identification," *MRS Communications*, pp. 1–7, 2019.

[13] L. Ibanez, B. Lorensen, M. McCormick, B. King, D. Blezek, H. Johnson, B. Lowekamp, jjomier, I. Kitware, G. Lehmann, A. Gelas, D. Zuki, M. Malaterre, karthikkrishnan, B. Hoffman, S. R. Aylward, Will, X. Liu, stnava, N. Dekker, M. Popoff, N. Tustison, gabehart, Sean, alex gouaillard, andinet enquobahrie,

B. Helba, T. Vercauteren, P. Hernandez-Cerdan, and J. H. L. Gorroo, "Insightsoftwareconsortium/itk: Itk 4.13.3," Aug. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.3971206

[14] G. Gardiner, "The next generation of ceramic matrix composites," 2017. [Online]. Available: https://www.compositesworld.comblog/post/the-next-generation-of-ceramic-matrix-composites

[15] N. M. Larson and F. W. Zok, "Ex-situ xct dataset for x-ray computed tomography of microstructure evolution during matrix impregnation and curing in unidirectional fiber beds," http://dx.doi.org/doi:10.18126/M2QM0Z, 2019.

[16] K. C. J. P. R. S. T. Blaiszik, B. and I. Foster., "The Materials Data Facility: Data services to advance materials science research."

[17] D. Gürsoy, F. De Carlo, X. Xiao, and C. Jacobsen, "TomoPy: a framework for the analysis of synchrotron tomographic data," *Journal of Synchrotron Radiation*, vol. 21, no. 5, pp. 1188–1193, Sep 2014. [Online]. Available: https://doi.org/10.1107/S1600577514013939

[18] MacNeil, Ushizima, Panerai, Mansour, Barnard, and Parkinson, "Interactive volumetric segmentation for textile microtomography data using wavelets and non-local means," *Journal of Statistical Analysis and Mining*, Sep 2019.

[19] S. Miramontes, D. Ushizima, and D. Parkinson, "Evaluating fiber detection models using neural networks," in *Advances in Visual Computing. ISVC 2019. Lecture Notes in Computer Science*, vol. 11845. Springer, 2019.

[20] Z. Ronaghi, R. Thomas, J. Deslippe, S. Bailey, D. Gursoy, T. Kisner, R. Keskitalo, and J. Borrill, "Python in the NERSC exascale science applications program for data," in *Proceedings of the 7th Workshop on Python for High-Performance and Scientific Computing*, ser. PyHPC'17. New York, NY, USA: Association for Computing Machinery, 2017. [Online]. Available: https://doi.org/10.1145/3149869.3149873

[21] Xarray developer team, "Xarray," http://xarray.pydata.org/, 2020.

[22] Zarr developer team, "Zarr," https://zarr.readthedocs.io/, 2020.

[23] M. McCormick and D. Ushizima, "Microct pyramids," https://fiber-bed-zarr.netlify.app/, 2020.

[24] Dask developer team, "Dask," https://dask.org/, 2020.

[25] E. Gouillart, J. Nunez-Iglesias, and S. van der Walt, "Analyzing microtomography data with python and the scikit-image library," *Advanced Structural and Chemical Imaging*, vol. 2, no. 1, p. 18, 2017.

[26] John Kirkham, "Dask image," https://image.dask.org/, 2020.