OpenMSIStream: A Python package for facilitating integration of streaming data in diverse laboratory environments

Margaret Eminizer [1¶], **Sam Tabrisky** [2,3,4], **Amir Sharifzadeh** [1], **Christopher DiMarco** [4], **Jacob M. Diamond** [4,6], **K. T. Ramesh** [4], **Todd C. Hufnagel** [4,5,6], **Tyrel M. McQueen** [4,5,7,8], **and David Elbert** [1,4]

1 Institute for Data Intensive Engineering and Science (IDIES), The Johns Hopkins University, USA **2** Department of Biology, Dartmouth College, USA **3** Department of Computer Science, Dartmouth College, USA **4** Hopkins Extreme Materials Institute (HEMI), The Johns Hopkins University, USA **5** Department of Materials Science and Engineering, The Johns Hopkins University, USA **6** Department of Mechanical Engineering, The Johns Hopkins University, USA **7** Department of Chemistry, The Johns Hopkins University, USA **8** Institute for Quantum Matter (IQM), William H. Miller III Department of Physics and Astronomy, The Johns Hopkins University, USA ¶ Corresponding author

DOI: 10.21105/joss.04896

Software

- Review
- Repository
- Archive

Editor: Amy Roberts

Reviewers:

- @lucask07
- @SergeyYakubov

Submitted: 07 October 2022
Published: 09 March 2023

Summary

OpenMSIStream provides seamless connection of scientific data stores with streaming infrastructure to allow researchers to leverage the power of decoupled, real-time data streaming architectures. Data streaming is the process of transmitting, ingesting, and processing data continuously rather than in batches. Access to streaming data has revolutionized many industries in the past decade and created entirely new standards of practice and types of analytics. While not yet commonly used in scientific research, data streaming has the potential to become a key technology to drive rapid advances in scientific data collection (e.g., Brookhaven National Lab (2022)). This paucity of streaming infrastructures linking complex scientific systems is due to a lack of tools that facilitate streaming in the diverse and distributed systems common in modern research. OpenMSIStream closes this gap between underlying streaming systems and common scientific infrastructure. Closing this gap empowers novel streaming applications for scientific data including automation of data curation, reduction, and analysis; real-time experiment monitoring and control; and flexible deployment of AI/ML to guide autonomous research.

Streaming data generally refers to data continuously generated from multiple sources and passed in small packets (termed messages). Streaming data messages are typically organized in groups called topics and persist for periods of time conducive to processing for multiple uses either sequentially or in small groups. The resulting flows of raw data, metadata, and processing results form “ecosystems” that automate varied data-driven tasks. A strength of data streaming ecosystems is the use of publish-subscribe (“pub/sub”) messaging backbones that decouple data senders (publishers) and recipients (subscribers). Popular message-focused middleware solutions such as RabbitMQ (VMware, 2022), Apache Pulsar (Apache Software Foundation, 2022b), and Apache Kafka (Apache Software Foundation, 2022a) all provide differing capabilities as backbones. OpenMSIStream provides robust and efficient, yet easy, access to the rich data streaming systems of Apache Kafka.

Statement of Need

The majority of scientific research today relies on semi-automated collection, reduction, and analysis of data. Vast improvements in instrumentation and computational modeling, however, have rapidly increased the volume and quality of that data, providing an opportunity for transformative acceleration of science. Such transformation will require scalable integration of data resources for development and real-time deployment of AI and machine learning, so as to facilitate pervasive laboratory automation and the development of autonomous decision-making in research.

OpenMSIStream development was driven by needs in materials science research with its central goal of discovering novel materials to meet urgent societal needs in fields as diverse as energy, health, the environment, and security. It provides a tool to accelerate materials research through the development and integration of data and data platform resources in the Materials Innovative Infrastructures prioritized by the Materials Genome Initiative (MGI) (U.S. White House Office of Science and Technology Policy, 2021). Specifically, OpenMSIStream simplifies the process of standing up streaming systems by abstracting details while still providing full functionality and configurability. It provides file-oriented tools to align with the prevailing paradigms of scientific instrumentation and data analysis. The suite of tools manage chunking of data files of any type to form manageable messages in independently-configured topics. Records can be read back to reconstruct the original data file contents and trigger flexible processing code to run as entire files become available from the stream in real time.

The messaging backend for OpenMSIStream is provided by the `confluent_kafka` Python wrapper (Confluent, 2022) around Apache Kafka. A Kafka "broker" persistently stores messages in ordered, append-only logs called "topics". "Producer" programs send messages to be appended to topics, while "consumer" programs read messages stored in those topics.

OpenMSIStream producers provide flexibility to upload single files to Kafka topics, and persistently watch directory trees on file systems for files to upload. OpenMSIStream consumers can download files uploaded to topics to disk, or transfer files to object stores through S3 API compliance (Amazon Web Services, 2022). OpenMSIStream also includes base classes that users can extend to invoke individualized Python code on the contents of reconstructed files and save processing results locally or produce them as messages to different topics, including a specific implementation for automated extraction and re-production of metadata keys and values.

OpenMSIStream programs (or extensions thereof) can be run from the command line, in Docker containers, or installed to run persistently in the background as Windows Services or Linux daemons, all using the same simple interface. Producer and consumer programs and the central Kafka broker exist independently of one another, so they can run on computers where data are being generated by instruments, on machines hosting data storage, or on more powerful servers for analysis as necessary. The Kafka backend is fully customizable through a simple configuration file interface, and data uploaded to topics can optionally be encrypted on the broker using KafkaCrypto (McQueen, 2022).

OpenMSIStream was designed for deployment in diverse science laboratory environments. Lab scientist or student users need only minimal computing experience to set up a directory on an instrument computer to watch for data files and start running another program on a different computer to automate backups or transfers to local disks or cloud storage solutions. Slightly more advanced users can adapt their existing analysis codes to automatically perform analyses in real time and save results locally or send them off to another Kafka topic.

In materials science projects, it is common to see iterative scientific design workflows integrating contributions from several different labs that focus on material production, simulation, and characterization. Using data streaming to pass raw data, metadata, and analysis or simulation results automatically between these groups increases interoperability to tighten the materials design loop.

OpenMSIStream has been adopted as the streaming solution for the Open Material Semantic Infrastructure (OpenMSI) within the NSF-sponsored Designing Materials to Revolutionize and Engineer our Future (DMREF) collaboration at the Hopkins Extreme Materials Institute (HEMI). OpenMSIStream is also currently used to automate data transfer and analysis between electron microscopy laboratories at Cornell University and Johns Hopkins University as part of the NSF-sponsored PARADIM Materials Innovation Platform, and for similar purposes for X-ray instruments at the Materials Characterization and Processing (MCP) Facility at Johns Hopkins University. Further, OpenMSIStream will be deployed in the near future as part of the data streaming solutions for the ARL-sponsored High-Throughput Materials Discovery for Extreme Conditions (HTMDEC) project, as well as in the Integrated Materials Design and Processing Applications to Recycled Plastics DMREF project (NSF).

There are many existing open source Python libraries that implement streaming data workflows. For example, Bytewax (Bytewax, 2022) is a library for managing data flow with streaming components that can integrate with Kafka, providing an interface to different data sources and generalized operators at the single-message level. The Bluesky data collection framework (Brookhaven National Lab, 2022) implements a DataBroker class that can access local and remote files, registering incremental updates to them as data are collected. Other libraries like Streamz (Rocklin, 2022) can be used to build components of local streaming architectures that don't access or publish data remotely. While OpenMSIStream can integrate with any of these Python tools, it also fills their common gap by providing full stream processing capability using a remote broker while remaining lightweight and accessible to laboratory investigators accustomed to writing Python code to analyze files stored on disk. OpenMSIStream also interfaces seamlessly with other existing components of scientific software stacks such as `NumPy` (Harris et al., 2020), `SciPy` (Virtanen et al., 2020), and pandas (McKinney, 2010; The pandas development team, 2020). The use of the Kafka backend allows users even more familiar with the Kafka ecosystem to take full advantage of non-Python tools like Kafka Streams (Apache Software Foundation, 2022c) for further data handling outside of OpenMSIStream.

Acknowledgments

The development of OpenMSIStream has been financially supported by NSF Awards #1921959 and #2129051. Tyrel M. McQueen and the development of KafkaCrypto were supported by the Platform for the Accelerated Realization, Analysis, and Discovery of Interface Materials (PARADIM), a NSF Materials Innovation Platform, under cooperative agreement #1539918.

References

Amazon Web Services. (2022). *Amazon Simple Storage Service*. https://aws.amazon.com/s3/

Apache Software Foundation. (2022a). *Apache Kafka* (Version 3.2). https://kafka.apache.org/

Apache Software Foundation. (2022b). *Apache Pulsar* (Version 2.10.1). https://pulsar.apache.org/

Apache Software Foundation. (2022c). *Kafka Streams*. https://kafka.apache.org/documentation/streams/

Brookhaven National Lab. (2022). Bluesky — an experiment specification & orchestration engine. In *GitHub repository*. GitHub. https://github.com/bluesky/bluesky

Bytewax, Inc. (2022). Bytewax - Python stateful stream processing framework. In *GitHub repository*. GitHub. https://github.com/bytewax/bytewax

Confluent, Inc. (2022). Confluent-kafka-python: Confluent's Kafka Python client. In *GitHub repository*. GitHub. https://github.com/confluentinc/confluent-kafka-python

Harris, C. R., Millman, K. J., Walt, S. J. van der, Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., Kerkwijk, M. H. van, Brett, M., Haldane, A., Río, J. F. del, Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357–362. https://doi.org/10.1038/s41586-020-2649-2

McKinney, Wes. (2010). Data Structures for Statistical Computing in Python. In Stéfan van der Walt & Jarrod Millman (Eds.), *Proceedings of the 9th Python in Science Conference* (pp. 56–61). https://doi.org/10.25080/Majora-92bf1922-00a

McQueen, T. M. (2022). kafkacrypto: Message layer encryption for Kafka. In *GitHub repository*. GitHub. https://github.com/tmcqueen-materials/kafkacrypto

Rocklin, M. (2022). Streamz. In *GitHub repository*. GitHub. https://github.com/python-streamz/streamz

The pandas development team. (2020). *Pandas-dev/pandas: pandas* (latest). Zenodo. https://doi.org/10.5281/zenodo.3509134

U.S. White House Office of Science and Technology Policy. (2021). *Materials genome initiative strategic plan*. https://www.mgi.gov/sites/default/files/documents/MGI-2021-Strategic-Plan.pdf

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261–272. https://doi.org/10.1038/s41592-019-0686-2

VMware, Inc. (2022). *RabbitMQ* (Version 3.10.7). https://www.rabbitmq.com/