A High-level Design for Bidirectional Data Streaming to High-Performance Computing Systems from External Science Facilities

Michael J. Brim
Anjus George
Amir Shehata
Corwin Lester
David Rogers
Patrick Widener
Ross Miller
Gustav Jansen
Rafael Ferreira da Silva
Sarp Oral

Approved for public release.
Distribution is unlimited.
This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor any of their employees, makes any warranty, express or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.
A High-level Design for Bidirectional Data Streaming to High-Performance Computing Systems from External Science Facilities

Michael J. Brim, Anjus George, Amir Shehata, Corwin Lester, David Rogers, Patrick Widener, Ross Miller, Gustav Jansen, Rafael Ferreira da Silva, Sarp Oral

March 30, 2024

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, TN 37831-6283
managed by
UT-Battelle LLC
for the
US DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725
CONTENTS

LIST OF FIGURES ................................................................. v
LIST OF TABLES ............................................................... vii
ACKNOWLEDGEMENTS ........................................................... ix
ABSTRACT .................................................................................... xi
1 INTRODUCTION ........................................................................ 1
2 MOTIVATION AND RELATED WORK ............................................ 3
   2.1 Representative Use Cases for Data Streaming to HPC ................. 3
      2.1.1 FRIB Greta .................................................................. 3
      2.1.2 SNS Neutrons .......................................................... 3
      2.1.3 SLAC LCLS ............................................................. 4
      2.1.4 INTERSECT AGILE .................................................... 4
   2.2 Data Streaming Use Case Characterization ................................... 5
   2.3 Generalized Data Streaming to HPC ......................................... 6
   2.4 General-purpose Data Streaming Frameworks .............................. 8
      2.4.1 Apache Kafka .......................................................... 8
      2.4.2 RabbitMQ ............................................................. 8
      2.4.3 Apache ActiveMQ ................................................... 8
      2.4.4 Redis ................................................................... 9
      2.4.5 NATS .................................................................. 9
      2.4.6 Nanomsg ............................................................. 9
3 SCIENTIFIC EDGE TO HPC DATA STREAMING REQUIREMENTS .......... 10
4 PROPOSED ARCHITECTURE FOR A DATA STREAMING TO HPC CAPABILITY ... 11
   4.1 Components of the Proposed Architecture ............................... 11
      4.1.1 OLCF Components .................................................. 11
      4.1.2 User Components ................................................... 12
   4.2 Orchestration Sequence for the Proposed Architecture ............... 12
   4.3 Containerized Data Streaming Services ..................................... 13
5 FUTURE PLANS FOR DEMONSTRATING THE CAPABILITY ............... 14
   5.1 Phase 1: Prototype Demonstration ......................................... 14
   5.2 Phase 2: Science Use Case Demonstration ............................... 14
REFERENCES ................................................................................. 15
LIST OF FIGURES

4-1 OLCF Architecture for Data Streaming to HPC .......................... 11
LIST OF TABLES

2-1  Data Streaming Characteristic Definitions ........................................... 5
2-2  Data Streaming Characteristics of Representative Use Cases ....................... 7
ACKNOWLEDGEMENTS

The authors would like to thank all the participants of the working group that contributed to the design and motivating use cases for the proposed Data Streaming to HPC capability. Besides the authors, this working group included representation from many organizational groups within the National Center for Computational Sciences:

- Data Lifecycle Technologies: Olga Kuchar and Dale Stansberry
- HPC Cybersecurity and Information Engineering: Ryan Adamson and Carl Bai
- HPC Infrastructure and Networking: Jeffrey Miller and Joel Comeaux
- Software Services Development: Ryan Prout, A.J. Ruckman, and Paul Bryant
- Technology Integration: Chris Zimmer
- Workflow and Ecosystem Services: Ketan Maheswari

This research used resources of the Oak Ridge Leadership Computing Facility, which is a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725.
ABSTRACT

Cutting-edge science is increasingly data-driven due to the emergence of scientific machine learning models that can guide scientists toward fruitful areas of exploration. Experimental science facilities such as light and neutron sources, particle colliders, and radio astronomy telescopes are also producing raw measurement data at rates that exceed available data storage and computing capacity at those facilities. As a result, scientific workflows are being developed that concurrently couple experiments at science facilities with high-performance computing (HPC) facilities to enable analysis of experimental data while the experiment is ongoing, and where analysis results are potentially fed back to the experiment for use in experimental control and/or steering in a time-sensitive manner. Our goal is to design, prototype, and deploy a new capability for the Oak Ridge Leadership Computing Facility (OLCF) that enables such workflows through support for bidirectional, memory-based streaming of data from external experiments into and out of OLCF HPC systems. This high-level design document describes the related work and motivating use cases that inform our understanding of the technical requirements for this capability, and describes a proposed architectural solution that meets these requirements and our plans for demonstrating the capability.
1 INTRODUCTION

Cutting-edge science is increasingly data-driven due to the emergence of scientific machine learning models that can guide scientists toward fruitful areas of exploration. Experimental science facilities such as light and neutron sources, particle colliders, and radio astronomy telescopes are also producing raw measurement data at rates that exceed available data storage and computing capacity at those facilities. As a result, scientific workflows are being developed that concurrently couple experiments at science facilities with high-performance computing (HPC) facilities to enable HPC-based analysis of experimental data while the experiment is ongoing, and where analysis results are potentially fed back to the experiment for use in experimental control and/or steering in a time-sensitive manner. Our goal is to design, prototype, and deploy a new Oak Ridge Leadership Computing Facility (OLCF) capability that enables such workflows through support for bidirectional streaming of data from external experiments into and out of OLCF HPC systems. For brevity, we refer to this capability as Data Streaming to HPC. For this capability we want to focus on memory-to-memory data movement (i.e., moving data directly from the memory of producer applications to the memory of consumer applications) and avoid requiring data transfers between file systems as the basis for moving data between experimental and HPC systems. File-based data movement is already well-supported by the U. S. Department of Energy (DOE) Advanced Scientific Computing Research (ASCR) user facilities and has been shown to be a bottleneck for certain use cases [2, 20].

The urgent need for such a capability is motivated by several programmatic factors. First and foremost, the DOE ASCR Facilities Division has officially launched the Integrated Research Infrastructure (IRI) program and named it as agency priority goal for the current government fiscal year (i.e., FY24). This program seeks to enable new integrated research that accelerates discovery and innovation through seamlessly leveraging the wide array of scientific capabilities (i.e., research tools and infrastructure) provided by DOE user facilities, both experimental and computational. The basis for establishing the IRI program was set by a year-long effort from the ASCR IRI Task Force’s Architecture Blueprint Activity (IRI ABA). The IRI ABA produced a final report [16] that identifies three integrated science blueprint patterns important to the success of DOE integrated scientific research. Of these patterns, two encompass potential use cases for the Data Streaming to HPC capability. In the "Time-sensitive" IRI architectural pattern, an experimental feedback loop may require low or near real-time latency for data streaming into and out from the HPC system. In the "Data Integration-intensive" IRI pattern, analysis performed on HPC systems is focused on integration of large quantities of data from disparate sources. Use cases matching this pattern may involve multiple independent data stream sources, such as widely distributed sensors or a set of unique instruments/detectors. Another programmatic motivator is Interconnected Science Ecosystem (INTERSECT), an Oak Ridge National Laboratory (ORNL) Laboratory Directed Research and Development (LDRD) initiative that seeks to design and prototype a common architectural framework for interconnected science ecosystems that enable automated or autonomous scientific workflows spanning computational and experimental systems [14]. INTERSECT has several domain science use case projects driving the architectural design and ecosystem prototypes. A few of these use cases have explicit requirements for Data Streaming to HPC, while others may be able to incorporate such a capability in future extensions. Finally, there is a timely need to better understand how such a capability impacts decisions related to the upcoming acquisition of the next leadership computing system for OLCF (i.e., OLCF-6), where IRI enablement and interoperability has been included as a mission need [9].

There are several potential benefits to addressing this need now. The most pragmatic benefit is that by leading the design and prototyping of the capability, we are much more likely to land on a design and
technical solutions that meet the needs of our expected users and can be eventually deployed in the OLCF production environment. The resulting solution may also be suitable for adoption by other DOE facilities. Furthermore, by using OLCF Advanced Computing Ecosystem (ACE) development resources such as the IRI testbed [8] we are free to be agile in trying diverse technical approaches without fear of impacting delivery of production science. By making these development resources available to external collaborators, we can foster new partnerships with DOE science programs.

Based on the programmatic motivators and perceived benefits and risks, we conclude that prompt action to design and demonstrate a Data Streaming to HPC capability is warranted. This high-level design (HLD) document describes the related work and motivating use cases that inform our understanding of the technical requirements for this capability, and describes a proposed architectural solution that meets these requirements. The HLD document concludes with a discussion of future plans for demonstrating the capability.
2 MOTIVATION AND RELATED WORK

A recent report [7] from the ESNet program within the DOE Office of Science (DOE-SC) analyzes the prevalence of IRI blueprint patterns across the breadth of DOE-SC programs, including Biological and Environmental Research (BER), Basic Energy Sciences (BES), Fusion Energy Sciences (FES) High Energy Physics (HEP), and Nuclear Physics (NP). The report’s findings indicate that for all the use cases studied by ESNet, a mere 1% do not include a time-sensitive or data integration-intensive component in their scientific workflows, and 39% exclusively use one or both of these patterns. Since data streaming is identified as a key gap for both patterns in the IRI ABA report [16], our expectation is that a Data Streaming to HPC capability will have a broad user base across DOE’s science portfolio.

In this section, we focus on use cases for which we have had direct discussions with potential users. First, we review four distinct representative use cases that capture the breadth and diversity of potential scientific workflows that can benefit from Data Streaming to HPC. We then present a set of criteria we have developed to help characterize the data streaming needs of these use cases. Finally, we discuss related efforts to architect general data streaming approaches for scientific edge to HPC environments and a survey of general-purpose data streaming frameworks.

2.1 REPRESENTATIVE USE CASES FOR DATA STREAMING TO HPC

2.1.1 FRIB GRETA

The Gamma-Ray Energy Tracking Array (GRETA) [13] is a new scientific instrument for low-energy nuclear science that will be deployed at the Facility for Rare Isotope Beams (FRIB) [19], a DOE-SC Office of Nuclear Physics user facility located at Michigan State University. GRETA enables real-time study of the energy and three-dimensional position of gamma rays within atomic nuclei with "up to 100 times greater sensitivity than existing detectors" [12], and FRIB provides a large catalog of rare short-life radioactive isotopes to study. The proposed GRETA science workflow named Deleria involves continuous streaming of experimental data during live measurements through ESNet to over one hundred data analysis processes running within an HPC system. The initial experimental data rate to be streamed to HPC is estimated at 40 Gbps, although future work may enable 100 Gbps data streams. The data analysis uses signal decomposition to transform waveform data into detected events that are then streamed to an external event aggregation and ordering process which generates the final experimental result data. The outbound event stream is estimated at 240 MB/sec (i.e., roughly 2 MB/sec per compute process). Incremental results from the aggregation process can be used to monitor and visualize experiment progress. The results from each experiment are used by scientists to help debug and optimize device configurations for the next experiment. Deleria is currently built using the nanomsg network communication software and uses containerized data transfer and data analysis processes.

2.1.2 SNS Neutrons

The "Edge-to-Exascale with Neutrons" project is an ORNL LDRD project that couples an experiment running at ORNL’s Spallation Neutron Source (SNS) with a transformer-based artificial intelligence (AI) model for Bragg peak detection concurrently trained on an HPC system using hundreds to thousands of compute nodes. The scientific workflow involves streaming experiment data to a local data reduction process at SNS before streaming the reduced data to HPC system for use in refining the AI model. The refined model, whose data size is on the order of a few MB, is sent back to a compute resource at SNS
where it is used for inferencing to determine if the experiment has reached its goal state. The data streaming to the HPC system is bursty, sending approximately 10 GB of reduced data every five minutes during a total experiment runtime of up to two hours. The raw experiment data does not need to be persisted, but both the reduced data used for training and the refined image result data needs to be retained.

2.1.3 SLAC LCLS

The Linac Coherent Light Source (LCLS) at SLAC National Accelerator Laboratory provides X-ray scattering for the study of the molecular structure and function properties of materials and substances. Past work [2] has demonstrated a scientific workflow that forwards experiment data from LCLS to the National Energy Research Scientific Center (NERSC) at Lawrence Berkeley National Laboratory (LBL) to provide quick data analysis that enables scientists to make informed decisions when configuring subsequent experiment runs. The existing workflow uses file-based transfers of experimental image data from the SLAC analysis file system to NERSC shared storage systems, using a clustered data transfer service known as XRootD that runs at both sites. Currently, the site-to-site data transfer over ESNet reaches peak speeds of around 2.6 GB/sec (accounting for both the network and storage I/O) when data is stored to the NERSC Cori scratch file system. At SLAC, the Apache Kafka event streaming platform is currently used to notify the data movement infrastructure that new files are available for transfer to NERSC and to signal completion of each file’s transfer. The analysis application is containerized and runs on NERSC compute systems within job reservations. It is MPI-based and uses a single orchestrator rank to distribute analysis tasks to other ranks that directly pull the relevant data from the storage system. Each analysis rank stores results to its own file, while also recording per-image statistics in a shared database that can be accessed by users continuously during the analysis. For the current LCLS, this workflow is sufficient for the accompanying data rates and desired analysis turnaround time of roughly 30 minutes. However, LCLS-II is scheduled to come online in 2025 and produce experimental data at 400 times the rate of LCLS. It is thus expected that data streaming at rates up to 100 GB/s to the HPC analysis will be necessary to maintain responsive result generation for experiment refinement decisions between runs. A future approach to data streaming, informally referred to as LCLStream, seeks to stream events while the experiment is running to enable online analysis and potential experiment steering, rather than waiting until the experiment has completed and the events are written to the analysis file system.

2.1.4 INTERSECT AGILE

The Automation for Grid Interconnected-Laboratory Emulation (AGILE) domain science project within the INTERSECT initiative aims to evaluate the impacts of deploying new types of power electronics (PE) such as photovoltaics within the electric grid through a coupling of detailed hardware emulation of a local PE environment with computing-based simulations of near and far grid behavior [15]. The far grid environment represents a large, widely distributed physical area and requires HPC-based simulation incorporating both physics-based and AI-based models. The hardware emulation operates continuously with a 100 Hz interval. At the completion of each interval, a small set of parameters (on the order of KB) are measured and streamed to the coupled simulations to perform state updates, and a similar set of parameters are streamed back from each simulation to the hardware emulation. The scientific workflow is currently in a prototype state where streaming of parameters occurs via a direct network socket connection from the control machine for the hardware emulation to the near grid simulation. Streaming to the far grid simulation has yet to be incorporated. The parameter updates do not need to be persisted for later analysis,
but there may be interest in saving traces of the parameters for later replay to enable a digital twin of the emulation environment.

### 2.2 DATA STREAMING USE CASE CHARACTERIZATION

We have chosen eight characteristics of the data streaming use cases to help demonstrate their diversity. These characteristics are defined in Table 2-1 along with their associated values or metrics. Table 2-2 summarizes the characteristics of each of our four representative use cases.

**Table 2-1. Data Streaming Characteristic Definitions**

<table>
<thead>
<tr>
<th>Use Case Characteristic</th>
<th>Description</th>
<th>Values or Metric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment Concurrency with HPC</td>
<td>When the data analysis or processing on HPC occurs in relation to the scientific experiment that produces the data.</td>
<td><em>Concurrent</em> - Analysis or processing occurs during the runtime of the experiment. <em>Sequential</em> - Analysis or processing occurs after the experiment has completed.</td>
</tr>
<tr>
<td>Workflow Pattern</td>
<td>How the results of the data analysis or processing on HPC are used in the scientific workflow.</td>
<td><em>Monitoring</em> - Results are used to monitor the progress of an ongoing experiment. <em>Steering</em> - Results are used to modify the configuration or actions of an ongoing experiment. <em>Design</em> - Results are used for design or configuration of subsequent experiments.</td>
</tr>
<tr>
<td>Data Production Periodicity</td>
<td>Whether experiment data is streamed to the HPC system continuously or in periodic bursts.</td>
<td><em>Continuous</em> - Data is produced at a relatively constant rate across the lifetime of the experiment. <em>Bursty</em> - Data is produced by the experiment in bursts separated by periods of no data production.</td>
</tr>
<tr>
<td>Data Consumption Semantics</td>
<td>How data consumers on the HPC system receive the data.</td>
<td><em>Push-to-HPC</em> - Data is streamed to the HPC consumer(s) immediately upon production. When there are multiple consumers, this method requires a pre-determined data distribution among consumers (e.g., round-robin). <em>Pull-from-HPC</em> - The HPC analysis is notified when data is available on the stream and consumers choose when and how to pull the data.</td>
</tr>
<tr>
<td>Data Stream Elements</td>
<td>The type of data element that is produced and consumed via the data stream.</td>
<td><em>Events</em> - Each element is a unique event. The data contained within an event must provide the required information for establishing uniqueness from all other events. <em>Files</em> - Each element corresponds to a file. <em>Messages</em> - Each element is a message containing arbitrary data.</td>
</tr>
<tr>
<td>Use Case Characteristic</td>
<td>Description</td>
<td>Values or Metric</td>
</tr>
<tr>
<td>-------------------------</td>
<td>-------------</td>
<td>-----------------</td>
</tr>
<tr>
<td>Data Stream Persistence</td>
<td>Whether data from the stream should be persisted, and for how long.</td>
<td>None - Stream data is available until consumed. Space-limited - Stream data is buffered within a durable storage area of a specified size where it remains available until being overwritten by more recent data. Time-limited - Stream data is buffered in a durable storage area where it remains available for a specified duration after it is first consumed. Persistent - Stream data is persisted to a durable storage area where it remains available until it is explicitly deleted.</td>
</tr>
<tr>
<td>Data Stream Bandwidth</td>
<td>The desired data throughput for a data stream. The stream will deliver at least the desired throughput when data is produced at an equivalent or higher rate.</td>
<td>megabytes/second (MB/s) gigabytes/second (GB/s)</td>
</tr>
<tr>
<td>Data Stream Latency</td>
<td>The desired maximum latency for end-to-end delivery of a data element from a producer to a consumer. The individual data elements are expected to be small (e.g., less than a kilobyte).</td>
<td>seconds (s)</td>
</tr>
</tbody>
</table>

### 2.3 GENERALIZED DATA STREAMING TO HPC

SciStream introduces the challenges involved in secure, memory-to-memory data streaming between scientific instrument endpoints that exist in disjoint network domains [4]. SciStream defines a middlebox architecture utilizing separate control and data planes and a well-defined protocol for data stream orchestration. A SciStream user agent service deployed on the public wide-area network (WAN) serves as the primary contact for users to orchestrate creation of data streams. The orchestration includes authentication of the user with each endpoint to generate certificates used for securing control and data connections. A persistent control plane server is deployed on a Science DMZ gateway node bridging the public WAN and HPC system network at each endpoint to manage data stream resource allocation. Upon receiving a request from the user to establish a new data stream, the user agent negotiates the stream requirements such as desired bandwidth, number of streaming channels, and stream lifetime with each endpoint control server, and then responds to the user with a unique id for the request and a data stream configuration that can be established at both endpoints. Because the stream configuration provided to the user may differ from the original request, the user has the option of declining or accepting the configuration. For accepted stream configurations, the user agent directs the control servers to establish the negotiated stream by launching a data plane server on a gateway node for each channel. Each data server reports its channel port back to the control server for use in creating a connection map that fully details the data streaming path from producers to consumers. A user application or third-party controller at each
<table>
<thead>
<tr>
<th>Use Case Characteristic</th>
<th>FRIB Greta</th>
<th>SNS Neutrons</th>
<th>SLAC LCLS [current]</th>
<th>SLAC LCLS [future]</th>
<th>INTERSECT AGILE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment Concurrency with HPC</td>
<td>Concurrent</td>
<td>Concurrent</td>
<td>Sequential</td>
<td>Concurrent</td>
<td>Concurrent</td>
</tr>
<tr>
<td>Workflow Pattern(s)</td>
<td>Design, Monitoring</td>
<td>Monitoring</td>
<td>Design</td>
<td>Steering</td>
<td>Steering</td>
</tr>
<tr>
<td>Data Production Periodicity</td>
<td>Continuous</td>
<td>Bursty (5 minutes)</td>
<td>Bursty (30 minutes)</td>
<td>Continuous</td>
<td>Continuous</td>
</tr>
<tr>
<td>Data Consumption Semantics</td>
<td>Pull-from-HPC</td>
<td>Pull-from-HPC</td>
<td>Pull-from-HPC</td>
<td>Pull-from-HPC</td>
<td>Push-to-HPC</td>
</tr>
<tr>
<td>Data Stream Elements</td>
<td>Events (images)</td>
<td>Files (images)</td>
<td>Files</td>
<td>Events</td>
<td>Messages (parameters)</td>
</tr>
<tr>
<td>Data Stream Persistence</td>
<td>None</td>
<td>Time-limited (experiment data)</td>
<td>Persistent (experiment data and analysis results)</td>
<td>Persistent (experiment data and analysis results)</td>
<td>Space-limited (experiment data)</td>
</tr>
<tr>
<td>Data Stream Bandwidth</td>
<td>15 GB/s to HPC</td>
<td>5 GB/s to HPC</td>
<td>100 GB/s to HPC</td>
<td>10 GB/s to HPC</td>
<td>N/A</td>
</tr>
<tr>
<td>Data Stream Bandwidth</td>
<td>240 MB/s from HPC</td>
<td>10 MB/s from HPC</td>
<td>10 MB/s from HPC</td>
<td>10 MB/s from HPC</td>
<td>N/A</td>
</tr>
<tr>
<td>Data Stream Latency</td>
<td>≤ 10.0 sec end-to-end (from data source to storage)</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>≤ 1.0 sec to/from HPC</td>
</tr>
</tbody>
</table>

Table 2-2. Data Streaming Characteristics of Representative Use Cases

The endpoint uses the unique request id to provide connection information for the producer or consumer and to request information about the local data server ports. When the full connection map has been established and shared amongst the control and data servers, the data channels are established between the hops in the data plane using reverse transport layer network connections (i.e., receivers connect to senders via UDP or TCP sockets). When the data stream is no longer required, the user submits a termination request to the user agent with the unique id, which starts the process of terminating the data servers at each endpoint.

Key contributions of the SciStream architectural design include support for dynamic orchestration of data streams from concurrent applications, allowing for third-party orchestrators such as workflow engines, the
separation of control and data planes to enable flexibility in the choice of data movement technologies, and explicit consideration for managing federated user identities and secure communication. Limitations of SciStream include an assumption of unidirectional single-host producer and single-host consumer streams, no support for multiplexing of streams or topics over data channels, and reliance on gateway nodes to directly bridge between the public WAN and the HPC system interconnect rather than permitting multiple network hops within an endpoint facility.

2.4 GENERAL-PURPOSE DATA STREAMING FRAMEWORKS

A general-purpose data streaming framework is a software infrastructure designed to facilitate the reliable and scalable exchange of data between different components, systems, or applications in a real-time or near-real-time manner. It enables the seamless flow of messages or events between producers and consumers, allowing for efficient communication and processing of data. Data streaming frameworks are used in various scenarios, such as real-time analytics, monitoring, logging, IoT data processing, and communication across distributed systems. Such frameworks provide a flexible and scalable infrastructure for handling continuous data streams, making them applicable across various use cases. These frameworks are designed to handle messaging patterns including point-to-point and publish-subscribe and are often used for building distributed and decoupled systems. Here, we survey widely adopted general-purpose streaming frameworks including Kafka, RabbitMQ, ActiveMQ, Redis, NATS, and nanomsg.

2.4.1 Apache Kafka

Apache Kafka is a distributed streaming platform for managing real-time data on a large scale [10]. Its publish-subscribe model facilitates the creation of scalable and fault-tolerant data pipelines, making it ideal for use cases such as real-time analytics, log aggregation, and event sourcing. Kafka’s unique log-based architecture ensures durability, allowing for reliable data storage and retrieval. Its architecture consists of producers that generate records and consumers that subscribe to topics. Brokers, the core servers manage topics and their partitions, ensuring parallel processing and scalable throughput, providing a resilient foundation for handling real-time data. While Kafka excels in complex scenarios, its learning curve and infrastructure requirements may be perceived as high for simpler use cases.

2.4.2 RabbitMQ

RabbitMQ is a reliable and versatile messaging broker [3]. It supports multiple producers and consumers, making it adaptable to various communication patterns, from task distribution to microservices communication. RabbitMQ ensures reliability through features like delivery guarantees, message persistence, and acknowledgment modes. Its architectural framework involves exchanges that function as sophisticated message routing agents, dictating the logic of message distribution to queues. The system’s push model, where the producer controls the message flow, suits applications that demand specific sequences and delivery guarantees. RabbitMQ supports dynamic scaling and clustering for enhanced fault tolerance. However, it lacks inherent ordering guarantees across multiple queues.

2.4.3 Apache ActiveMQ

Apache ActiveMQ is a feature-rich messaging broker known for its reliability and scalability [11]. Offering features such as message persistence, acknowledgment modes, and transactional messaging, ActiveMQ ensures reliable data exchanges. The system excels in handling large messages, even with limited memory,
providing a robust solution for scenarios requiring high-throughput message processing. Supporting both push and pull mechanisms, ActiveMQ can dynamically scale through the addition or removal of broker nodes in a cluster. ActiveMQ’s internal architecture is based on the Java Message Service standard and relies on a modular and extensible design. While it offers powerful features, configuring and managing complex setups may be challenging for some users.

2.4.4 Redis

Redis is a versatile data store and message broker valued for its speed and efficiency [18]. Widely used for caching, session storage, and real-time analytics, Redis employs a key-value data model. Its unique features include support for various data structures (strings, hashes, lists, sets, and more) and pub/sub capabilities for messaging. Redis is renowned for its low-latency responses and high throughput. Redis’s internal architecture features an in-memory data store with support for versatile data structures, leveraging a single-threaded, event-driven model for high throughput. However, its limitation lies in handling datasets that exceed available memory, as it may impact performance.

2.4.5 NATS

NATS (Neural Autonomic Transport System) is a lightweight and high-performance messaging system designed for simplicity and speed [1]. Employing a publish-subscribe model, NATS is commonly used for microservices communication and IoT scenarios. Its simplicity and low-latency communication make it stand out, but it may lack some advanced features found in other messaging systems such as complex routing and support for advanced queuing models. NATS is highly scalable and fault-tolerant, supporting clustering for horizontal scalability. NATS’s internal architecture is designed for lightweight and high-performance messaging. At its core, a NATS server acts as a central message broker orchestrating communication among clients. However, it may provide less strict ordering guarantees compared to some other systems.

2.4.6 Nanomsg

Nanomsg is a socket library offering various communication patterns and supporting custom transports with an improved threading model [6]. Implemented in C and compatible with a wide range of operating systems without additional dependencies, nanomsg enables scalable, lightweight, and brokerless messaging for fast and easy networking. It supports messaging patterns like one-to-one, many-to-many, request-reply, publish-subscribe, pipeline, and survey. However, its brokerless nature presents challenges, as nodes bear the responsibility for communication, potentially leading to inefficiencies and difficulties in network scaling, debugging, and monitoring. In dynamic environments, managing direct connections can be challenging, affecting consistency and message ordering. nanomsg’s throughput limitations for block sizes larger than one megabyte have been noted in previous studies, attributed to internal library constraints. While it guarantees ordered data within a single socket, ensuring order between different streams or sockets is uncertain, posing a fundamental challenge. An advanced version, nng, retains the brokerless architecture of its predecessor and offers enhanced scalability, reliability, and security, but its production readiness remains uncertain.
3 SCIENTIFIC EDGE TO HPC DATA STREAMING REQUIREMENTS

As highlighted above in Table 2-2 that summarizes the data streaming characteristics of our representative use cases, the technical requirements and usage models for data streaming vary widely according to the specific science needs of the workflow. Further, based on the abundance of general-purpose data streaming frameworks, it is clear that any given technological solution for data streaming cannot be a one-size-fits-all solution that meets the needs of all use cases. Therefore, it is our belief that an architectural approach that is flexible and supports the deployment of many technological solutions offers the best opportunity for providing a useful general-purpose Data Streaming to HPC capability.

The core requirements of this architectural approach for Data Streaming to HPC have already been identified by SciStream [4]. First, to minimize latency for use cases with near-real-time constraints the approach must enable data streaming from the memory of data producers to the memory of data consumers, rather than relying on intermediary file systems near producers and consumers. Second, the approach must enable the creation of secure data streams that span multiple domains, each with a possibly distinct security context that controls user authentication and access to network and computing resources. Third, the approach must support delegation of stream management and monitoring to third-party workflow orchestrators that are not stream endpoints through separation of stream control and data planes. Fourth, the approach should support both advanced reservation of network and computing resources as well as best-effort, on-demand resource allocations.

Besides the four core requirements identified by SciStream, we introduce four additional architectural requirements that were not previously addressed. First, for experimental steering use cases the architecture must support bidirectional data streaming where each endpoint may be both a producer and a consumer. Second, to support parallel data processing techniques the approach must permit the use of many endpoints (i.e., data producers and/or consumers) with a single established stream. Third, the approach should provide scientific workflows with the flexibility to choose the best technological solution for the streaming data plane, including the use of commodity-off-the-shelf (COTS) streaming frameworks popular in cloud applications. Finally, the approach must allow for multiple network hops within a single security context, as may be required when HPC systems do not provide data streaming gateway nodes that bridge the public WAN and the HPC system interconnect.

For ease of reference, we summarize the eight requirements of our proposed architectural approach for a general-purpose Data Streaming to HPC capability here:

- **R1:** Data streaming from the memory of data producers to the memory of data consumers.
- **R2:** Secure data streams that span multiple domains, each with a possibly distinct security context.
- **R3:** Separation of data streaming control and data planes.
- **R4:** Adaptability to both advanced reservation and on-demand allocation of data streaming resources.
- **R5:** Data stream endpoints may be both a data producer and a data consumer.
- **R6:** Data streams may be utilized by multiple concurrent endpoints.
- **R7:** Flexibility to deploy alternative technologies for the streaming data plane solution.
- **R8:** Streaming data plane paths may traverse multiple networks within a single security domain.
4 PROPOSED ARCHITECTURE FOR A DATA STREAMING TO HPC CAPABILITY

The proposed architectural approach to support Data Streaming to HPC for the OLCF is shown below in 4-1. This architecture focuses on providing the necessary capabilities for streaming workflows to establish secure, bidirectional, memory-to-memory data streams between application processes at external science facilities and application processes running within an OLCF HPC system.

![OLCF Architecture for Data Streaming to HPC](image)

Figure 4-1. OLCF Architecture for Data Streaming to HPC

4.1 COMPONENTS OF THE PROPOSED ARCHITECTURE

Here, we briefly describe the key components of this architecture, which includes both OLCF and user-supplied components.

4.1.1 OLCF Components

**Facility API and Gateway Services** - The *Facility API* is modeled after similar computational facility gateway service APIs such as those provided by NERSC [17] and CSCS [5]. Currently under development, the Facility API will provide authenticated users with a Representational State Transfer (REST) interface for controlling and monitoring OLCF compute and storage systems. A unique feature of the Facility API is support for *Workflow Token Management*. A workflow token provides secure access to a set of API gateway services for a specific time period. Each API Gateway Service provides a uniform API for managing specific classes of OLCF resources, such as HPC systems or storage systems.
**Data Streaming Nodes** - A Data Streaming Node (DSN) is a specialized gateway node that bridges the public WAN and internal OLCF networks using high-speed network adapters and incorporates fast non-volatile local storage (e.g., NVMe solid-state drive (SSD)) for use by streaming services. DSNs are user-allocatable resources managed by a container orchestration system (e.g., Kubernetes). Multiple DSNs may be allocated to a single workflow to meet specific network bandwidth, message processing rate, or resilience needs of a deployed streaming service. DSNs are independent from OLCF Data Transfer Nodes (DTNs) to avoid interference from file-based data transfers.

**HPC Systems** - High-Performance Computing systems provide advanced computational and network hardware to enable parallel processing of application data. HPC systems often include two user-accessible node types: Compute Node (CN) and Service Node (SN). CNs are intended for running compute-intensive and/or data-intensive user applications, while SNs provide resources for interactive user tasks such as code development or compute job management. When the HPC system employs a private high-speed network (HSN), CNs are connected only to the HSN, while SNs are typically connected to both the HSN and the facility’s infrastructure network. In such systems, a DSS may need to utilize processes running on SNs to proxy or forward data stream traffic between CNs and DSNs.

**Storage and Archive Systems** - Storage and Archive systems provide durable, long-term file or data storage for scientific workflows. For data streaming workflows, such storage systems primarily serve to help with transitioning from existing workflows utilizing file transfers between facilities. A DSS may also use storage systems to provide advanced features such as data stream persistence or materialization of data streams from stored data.

### 4.1.2 User Components

**Workflow Orchestrator** - A Workflow Orchestrator (WO) manages the tasks, data flows, and dependencies in scientific workflows that span distributed resources such as those involving multiple facilities. Typical orchestration activities include initiating and monitoring tasks on computational resources and scientific instruments, moving data between distributed sites or resources, and managing user identities and credentials across all involved security contexts.

**Data Streaming Service** - A Data Streaming Service (DSS) is a set of one or more coordinating processes that provide the data plane for streaming data between external and internal Application Stream Endpoints. The data streaming functionalities, abstractions, and semantics provided by a DSS may vary across technological solutions.

**Application Stream Endpoints** - An Application Stream Endpoint (ASE) is an application process that communicates with a DSS to produce and/or consume streaming data. The interface and semantics used for data production or consumption are dependent upon the technology used for the associated DSS.

### 4.2 ORCHESTRATION SEQUENCE FOR THE PROPOSED ARCHITECTURE

The expected sequence of actions necessary to utilize the proposed Data Streaming to HPC architecture is as follows:

1. **Authentication** - The WO (or an external ASE) uses the OLCF Facility API to authenticate the user and establish a secure API session.
2. Establish a Workflow Token - The WO (or an external ASE) uses the Workflow Token Management service to request a workflow token for the necessary lifetime and API services. For Data Streaming to HPC workflows, the Data Streaming Orchestration and Compute Orchestration services are required.

3. Allocate Resources - The WO (or an external ASE) uses the workflow token to request an allocation of DSNs from the Data Streaming Orchestration service and an allocation of HPC system CNs from the Compute Orchestration service. The resource allocation requests should indicate a specific time period that falls within the token lifetime. We envision two classes of resource allocations: reserved and on-demand. Reserved allocations guarantee that all requested resources are exclusively available to the user for the given period. On-demand allocations, on the other hand, provide best-effort elastic behavior based on the DSN and HPC resources currently available when deploying a DSS or running a HPC compute job. Each allocation (i.e., DSN and HPC) should be associated with a unique id that is returned to the WO/ASE for use in later steps.

4. Deploy a Data Streaming Service - The WO (or an external ASE) uses the workflow token and DSN allocation id with the Data Streaming Orchestration service to deploy a pre-built containerized data streaming service to the allocated DSNs (and possibly a HPC system SN when necessary). The service captures and returns the DSS-specific information about its runtime configuration and coordinates (e.g., DSN IP address and port) to the WO/ASE. This information can be shared with external ASEs to allow them to connect to and use the deployed DSS.

5. Run Compute Jobs - The WO (or an external ASE) uses the workflow token and HPC allocation id with the Compute Orchestration service to run one or more compute jobs within the allocation. The expected input to the run request is a job script suitable for the target HPC system. The DSS information captured by the Data Streaming Orchestration service in step 4 will be made available for use (e.g., as job environment variables) when the job script executes.

6. Release Resources - The WO (or an external ASE) uses the workflow token and previously obtained allocation ids to release any DSNs or HPC resources via the Data Streaming Orchestration and Compute Orchestration services.

4.3 CONTAINERIZED DATA STREAMING SERVICES

For security and infrastructure management reasons, all containers that will be used to deploy a DSS on DSNs will need to be configured and built within the OLCF environment. Appropriate documentation will be provided to users to ease this process. We expect to support both COTS streaming frameworks and custom user solutions.

COTS Data Streaming Container - A containerized DSS based on a COTS streaming framework that has been pre-configured for easy deployment to OLCF DSNs and fully integrated with the OLCF gateway services providing Data Streaming Orchestration and Compute Orchestration.

Custom Data Streaming Container - A containerized DSS based on user-supplied data streaming technology that is based on an OLCF DSN deployment template. Such custom solutions may require additional integration effort with the OLCF gateway services providing Data Streaming Orchestration and Compute Orchestration or the use of out-of-band communication mechanisms to enable stream connections from ASEs running on OLCF HPC systems.
5 FUTURE PLANS FOR DEMONSTRATING THE CAPABILITY

The demonstration for the new OLCF *Data Streaming to HPC* capability is planned to occur in two phases using OLCF Advanced Computing Ecosystem (ACE) resources located within the Open Development security enclave. These resources include the Olivine OpenShift services cluster and the IRI testbed computational and storage resources [8]. In the first phase, the *Prototype Demonstration*, we will demonstrate a full end-to-end realization of the capability using mock data producers and consumers and a COTS streaming framework such as RabbitMQ or Kafka. In the second phase, the *Science Use Case Demonstration*, we will collaborate with external partners to demonstrate a science use case that employs a custom data plane technology.

5.1 PHASE 1: PROTOTYPE DEMONSTRATION

1. Integrate *Data Streaming Orchestration* service within the FacilityAPI.
2. Deploy a COTS data streaming infrastructure on Olivine using the orchestration service and verify the functionality of external and internal endpoint clients.
3. Integrate the *Data Streaming Orchestration* service with the *Compute Orchestration* service of the FacilityAPI to enable endpoints running within Slurm compute jobs on the Defiant HPC system to connect to and use a data streaming service deployment on Olivine.
4. Demonstrate end-to-end application usage of the *Data Streaming to HPC* capability.

5.2 PHASE 2: SCIENCE USE CASE DEMONSTRATION

1. Pick a target science use case based on ACE testbed progress to date.
2. Develop and deploy a custom data streaming infrastructure for the use case on Olivine.
3. Develop and deploy the HPC compute portion of the use case to Defiant.
4. Demonstrate end-to-end science application usage of the *Data Streaming to HPC* capability.
# Bibliography


