Diaspora: Resilience-Enabling Services for Real-Time Distributed Workflows

Bogdan Nicolae, Justin Wozniak, Tekin Bicer, Hai Nguyen, Parth Patel, Haochen Pan, Amal Gueroudji, Maxime Gonthier, Valerie Hayot-Sasson, Eliu Huerta, et al.

Diaspora: Resilience-Enabling Services for Real-Time Distributed Workflows

Bogdan Nicolae*, Justin M. Wozniak*, Tekin Bicer*, Hai Nguyen*, Parth Patel*, Haochen Pan*, Amal Gueroudji*, Maxime Gonthier*, Valerie Hayot-Sasson*, Eliu Huerta*, Kyle Chard*, Ryan Chard*, Matthieu Dorier*, Nageswara S. V. Rao†, Anees Al-Najjar†, Alessandra Corsi‡, Ian Foster*

*Argonne National Laboratory, †Oak Ridge National Laboratory, ‡Johns Hopkins University
Email: {bnicolae, woz, tbicer, hai.nguyen, parth.patel, hpan, agueroudji, mgonthier, vhayotsasson, elihu, chard, rchard, mdorier, foster}@anl.gov, {raons, alnajjaram}@ornl.gov, acorsi2@jh.edu

***Abstract*—The need for real-time processing to enable automated decision making and experimental steering has driven a shift from high-performance computing workflows on a centralized system to a distributed approach that integrates remote data sources, edge devices, and diverse compute facilities. Under this paradigm, data can be processed close to the source where it is generated, thus reducing latency and bandwidth usage. System resilience is thus a key challenge, requiring distributed workflows to survive component failures and to meet stringent quality-of-service requirements, which results in the need to mitigate anomalies such as congestion and low availability of resources. To address these challenges, we propose Diaspora, a unified resilience framework that is inspired by event-driven communication patterns used in public clouds. Specifically, we propose an event fabric that extends across sites, facilities, and computations to provide timely, reliable, and accurate information about data, application, and resource status. On top of the event fabric, we build resilience-enabling services that combine QoS-aware data streaming, resilient data views, resilient compute and data resources, and anomaly detection and prediction, all of which collectively enhance workflow resilience for these scientific cases.**

***Index Terms*—real-time distributed HPC workflows, resilience, high-availability, data streaming, elasticity, anomaly detection and prediction**

I. Introduction

The increase in real-time processing requirements of modern science workflows has disrupted the traditional, centralized approach where large amounts of data are collected, stored, and *later* processed on a single high-performance computing (HPC) machine, to a distributed approach that aggregates remote data sources and compute facilities: instruments, data stores, edge devices, compute nodes belonging to different data centers, etc. This increase in system complexity is motivated by a confluence of factors, including increases in experimental data volume and velocity, the emergence of science-informed AI agents capable of rapid goal-oriented decision-making, and advanced hardware that enables both of these.

Distributed workflows have several advantages: (1) *performance and scalability*, thanks to techniques such as running the workflow tasks close to the data sources, organizing the computations into several overlapping intermediate stages, and streaming the data between stages, which ultimately reduce the latency compared with a centralized approach; (2) *improved resource utilization*, thanks to better availability of resources and load balancing, which minimizes idle time and maximizes throughput; (3) *privacy and compliance*, which can be enforced by requiring the data to be processed at the edge or within a specific jurisdiction; (4) *time-sensitive decision-making and steering of experiments*, which is possible due to the reduced latency.

Resilience beyond failures as a key challenge: To unlock these advantages, distributed workflows need to be *resilient*. In this case, resilience is not limited to surviving failures, but instead needs to be complemented with new resilience requirements due to the need to meet often stringent quality of service constraints (e.g., for uninterrupted processing of data streams). For example, data rates that are too low (e.g., due to temporary interruption or congestion on the network links) or reduced availability of HPC systems are anomalies that have a significant negative impact, which under certain circumstances is even more severe than failures. Achieving resilience requires methods for, on the one hand, detecting anomalies and monitoring service levels, and on the other, preparing for or responding to anomalies, for example via fault tolerance, scalability, redundancy, and self-healing protocols, and via adaptive management and partitioning of resources. Thus, there is a need for a unified resilience framework that hides the complexity of implementing and combining such methods, while meeting the desired quality-of-service constraints that the HPC workflows need for real-time processing.

Contributions: In this position paper, we propose the foundations for such a unifying framework. Experience in other domains, such as public clouds, has demonstrated the power of event-driven communication patterns as a foundation for a wide range of resilience solutions. However, such approaches remain virtually unexplored in the scientific computing community due to different scales, scopes, and service level requirements. We seize this opportunity to argue in favor of several integrated resilience solutions that leverage information provided by an underlying event fabric to meet broad classes of resilience needs. We summarize our contributions as follows: 1) We illustrate the need for surviving and performing well

during **both failures and anomalies** in the context of time-sensitive HPC applications that have an end-to-end urgency, such as the need to provide real-time feedback to an experiment or to train AI models that are deployed at an instrument for steering. We use ptychographic reconstruction as a science driver (§III).

2) We introduce a **hierarchical event fabric** to distribute events regarding task and resource status and configuration data, application data, and other information, as well as aggregations of such data, with controllable availability, consistency, and performance characteristics (§IV-A).
3) We propose resilience services that build on event streams to implement resilience capabilities. Specifically, we discuss the need for **QoS-aware data streams** that adapt to the flow between producers and consumers (§IV-B); **resilient data views** that automatically combine, cache, and reuse data items from multiple data streams in a single resilient abstraction (§IV-B); **resilient compute and data**, which handles aspects of data and task replication, restart from checkpoints, and elasticity (§IV-D); and **anomaly detection and prediction**, which is instrumental in informing the other services as much as possible in advance about failures or anomalies likely to happen in the near future, thus giving them a chance to prepare and take proactive actions (§IV-E).
4) We describe the integration of these services into a comprehensive, unified, resilient framework (§V).

II. Related work

Real-time HPC workflows: are increasingly popular in various scientific domains, such as high energy physics [1], [2], epidemic modeling [3], ptychographic reconstruction [4], [5], tomography [6], and Laue diffraction [7]. These applications follow a common pattern involving: (i) data acquisition from scientific instruments (e.g., ANL's Advanced Photon Sources) generating TB to PB per day [8], [9], (ii) real-time processing [10], [11], and (iii) result publishing for experiment adjustment [12]–[16], secondary analysis [1], [17], [18], and offline uses [19]–[21].

Traditional HPC resilience: at the core of any scientific computation are the data needed to make progress and/or that must be persisted for later reuse. Especially when the tasks are tightly coupled (e.g., MPI processes), the most widely used technique is *coordinated* checkpoint-restart [22], which requires that processes agree on a globally consistent state to be independently captured piece-wise by the processes and restarted from in case of failures. Without adequate data resilience support to keep the checkpoints and other reusable data safe, workflows may not complete successfully. In this regard, parallel file systems [23] can provide reliability, for example via replication or erasure coding, but have limited aggregate I/O bandwidth and geographical distribution. I/O libraries and middleware seek to overcome these challenges via, e.g., multi-level caching (node-local memories [24], [25]), burst buffers [26], key-value stores with versioning support [27]), and asynchronous flushing and prefetching to hide I/O latency [28]. Other optimizations include multi-level resilience strategies adapted to the failure rate of each level [29]), I/O aggregation [30], and forward recovery in case a certain loss is acceptable (e.g., some missing contributions to gradient averages can be tolerated in case of data-parallel AI training [31]).

Resilience of loosely coupled tasks: While effective for tightly coupled patterns, traditional HPC resilience techniques are insufficient in distributed workflows. In this case, the tasks running at different sites tend to be loosely coupled and failures need to be contained at local level due to global synchronization being prohibitively expensive. To address this challenge, alternative strategies have been proposed. For example, Condor logs tasks on stable storage and resubmits failed tasks by using two-phase commit [32], [33]. Spark [34] keeps track of intermediate resilient distributed data sets (RDDs), which are reconstructed on-the-fly in case of failures by re-executing the chain of tasks (recorded in a lineage) originally used to obtain them. KerA [35] uses a replicated virtual log structured approach for data streaming, which can be used to separate stream partitioning from durability and resilience, effectively enabling high throughput under concurrent access without sacrificing resilience in face of failures. Globus Flows records status on cloud storage [36]. Virtual data abstractions (re)generate data from specifications as needed [37]. Individual tasks may employ their own resilience strategies: while short-running tasks can be simply re-executed from scratch in case of failures, long-running tasks can be further protected using checkpoint-restart strategies.

Enforcing real-time QoS constraints: beyond fault tolerance, distributed HPC workflows need to enforce real-time QoS constraints, which is arguably even more challenging than failures, because of an inherent difficulty of detecting, predicting, and fixing anomalies that cause QoS violations [38]–[40]. LSTMs have been used to predict anomalies in electric grid networks [41], [42],hard drive disks [43], and even earthquakes [44]. Lightweight deep neural networks have been implemented inside operating system kernels [45], [46], and have been successful at predicting more difficult latency-sensitive anomalies, such as those appearing in 4G networks for cloud gaming [47]. Once detected and predicted, mitigating these anomalies involves trade-offs among consistency, availability, and partition tolerance [48], [49]. Current efforts, such as event-driven methods [50]–[52], micro-services [53]–[55], specialized queuing [56]–[58], scheduling [59], [60], and resource management systems [61], focus on preventing incorrect outputs due to failures but often neglect maintaining QoS despite anomalies. Additionally, limited attention is given to integrating different workflow patterns with varying resilience requirements in research infrastructures.

III. Motivating Example: Ptychography

Ptychography is an advanced data-intensive imaging technique that provides high-spatial resolutions—sub-10 nm—on large field-of-view [4], [5]. It consists of two main stages: (i) 2D ptychography, in which a sample is exposed to X-ray beam and scanned, while overlapping 2D diffraction patterns are

collected from high-speed detectors; and (ii) 3D tomography (or laminography), in which the sample is rotated on a rotation stage. 3D ptychography experiments at synchrotron radiation facilities can take extended periods and result in hundreds of TBs to PB-scale datasets [62], [63]. Processing of such datasets, e.g., 2D and 3D reconstruction tasks coupled with AI/ML-based image enhancement or science-specific steps (denoising, segmentation, anomaly detection, etc.), requires not only advanced computing techniques [64]–[68], and long-running workflows [18], [69] but also the usage of large-scale resources [70]–[73].

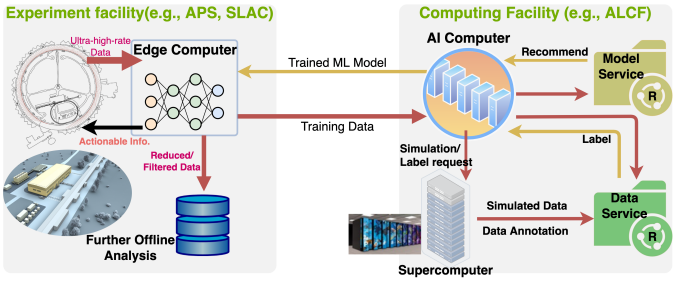

Fig. 1. An HPC machine is used to train ML models for fast data pre-processing close to a scientific instrument.

Since the time slot allocated for the use of the beamline is limited, real-time reconstruction of ptychography data is critical, e.g., for timely decision-making, adjusting and steering experiments, and ensuring science-relevant data acquisition. For example, in a quest to reduce latency to a level that allows for real-time 2D reconstruction, AI models (e.g., PtychoNN) are a promising avenue [74], [75], as they are much faster and consume less resources than traditional techniques. Since each object is unique and the beam cannot be stopped, it is not possible to use a pre-trained model or pause the workflow until we trained one. Instead, we need to apply an online solution consisting of the following steps: (1) *training warm-up*: transfer the full diffraction patterns to the HPC machine, use a classic (but expensive) algorithm to reconstruct the images, while at the same time training a learning model using these images as ground truth; (2) *switch to inferences:* as soon as the learning model is accurate enough, transfer it to the edge and use it to pre-process the diffraction patterns; (3) *fine-tuning:* continue receiving a subset of the full diffraction patterns, refine the learning model (continue the training), and periodically send a checkpoint of the model to the edge to improve the quality of the inferences [76], [77]. This pattern is illustrated in Figure 1.

From a resilience perspective, there are multiple simultaneous constraints that need to be satisfied. First, if the data flow between the scientific instrument and the HPC machine is bottlenecked or temporarily interrupted, alternatives such as data replay or reduction (compression, decimation, aggregation, interpolation) are needed that trade off quality for reduced latency. Second, data from multiple sources (diffraction patterns and reconstructed images) needs to be combined and streamed as atomic and resilient data units in order to enable the AI model training. Third, if the traditional or AI-enabled reconstruction fails, the affected processes need to be restarted from a checkpoint and they need to catch up with the rest of the workflow by revisiting and recomputing the lost data items. This may require more resources (i.e. elastic scale-up) and a buffer to retain a recent history of the streamed data items. Fourth, the sooner an anomaly can be predicted, the better domain experts can plan their experiments in order to optimize the performance-quality trade-off.

IV. Resilient Abstractions

To address the limitations of state-of-art approaches, we envision a series of composable abstractions that build on top of each other and can be selectively combined by distributed HPC workflows to achieve a desired trade-off between performance, quality of service/results and resilience requirements.

A. Event Fabric Foundation

Given the need to complement resilience beyond traditional checkpoint-restart with support for containing local failures within loosely coupled tasks and with support to enforce QoS constraints, a possible starting point is to identify a common denominator. The proven robustness of event-driven architectures adopted in the cloud computing community to address resilience and elasticity has prompted us to try to emulate a similar foundation for distributed HPC workflows. Events are central to event-driven architectures. They record messages of interest in the workflow and are organized into topics, each of which may have an arbitrary number of publishers and subscribers. By making the streaming of events resilient and matching publishers and subscribers around topics on a need-to-know basis, failures and QoS anomalies can be contained to local groups of interest, enabling self-healing without excessive synchronization.

As a consequence, we propose a specialized hierarchical event fabric to be uses as a resilience foundation. It extends from HPC at the edge to the globally accessible cloud and combines a high-speed HPC event fabric, capable of exploiting extreme transfer rates in HPC centers, with a cloud-hosted event fabric that spans administrative domains, which extends the reach of topics to publishers and subscribers belonging to remote tasks.

Mofka, a high-speed HPC event fabric: is a distributed event streaming service, designed using Mochi [78], a collection of methodologies and components to enable the rapid development of HPC data services. Mofka is analogous Kafka [50], and offers similar functionalities while leveraging HPC platform capabilities such as high-throughput and low-latency networks, remote direct memory access (RDMA), multicore CPUs and local NVMe storage devices. Mofka provides an infrastructure for task decoupling within HPC workflows and a better impedance matching between publishers and subscribers, which is geared towards high performance access under concurrency. Specifically, publishers and subscribers do not need to explicitly synchronize with each other anymore, or even live within the same job. Mofka makes use of local

persistent storage such as local SSDs to be able to restart after a crash, avoinding when possible the use of files on the PFS for persisting events, thus avoiding I/O bottlenecks. Data duplication can also be enabled composably via Mochi for resilience.

Octopus, a cloud-hosted event fabric: is built on Amazon's Managed Streaming for Apache Kafka (MSK). It is implemented as a multi-tenant service via which users can create and manage their own topics, and applications/services can then publish/consume events to/from topics. Deployed as a public REST service, Octopus relies on Globus Auth [79], a standards-compliant identity and access management platform, for authorization. Octopus implements the Globus Action Provider interface, enabling Globus Flows [36] to publish and consume events from arbitrary topics. This allows user-defined flows, for example, supporting real-time analysis of instrument data, to exchange information as they progress and also to adapt their performance in response to failures. Octopus implements a managed *trigger* model that supports automated actions in response to specified events. It allows users to define arbitrary triggers as Python functions using the function-as-a-service (FaaS) paradigm to execute the function and evaluate the action to perform. Octopus uses AWS EventBridge to filter events and AWS Lambda to execute functions.

B. QoS-Aware Data Streams

The producer-consumer pattern is often used in real-time HPC workflows to enable efficient concurrent processing at large scale. This pattern divides tasks between producers, which generate data, and consumers, which process it. Producers and consumers may be geographically distributed (e.g., edge and HPC machines, different HPC machines). By decoupling these roles, the system can achieve greater flexibility and scalability, allowing multiple producers and consumers to operate simultaneously in a pipeline without unnecessary delays. Data streams serve as a powerful abstraction to connect the producers and consumers by offering seamless and efficient data transfers between them. In HPC workflows, producers and consumers typically execute long-running computations on large datasets. In this context, resilience techniques that enable data integrity and continuity such as those provided by low-level protocols (e.g., retransmission in the case TCP/IP) and event the event fabric (e.g., replication of topics) are often insufficient. This happens for two reasons: (1) interruptions may be more severe (e.g., links going down or failed processes that need to restart from a checkpoint and catch up with the rest of the workflow); (2) even under correct operation, links may be congested (e.g., WAN links shared by multiple users), or compute resources may be oversubscribed, leading to poor quality-of-service whose negative impact is as damaging as failures (e.g., slow consumers that can process only a subset of the streamed data).

Thus, there is a need to provide a comprehensive high-level resilient data streaming abstraction that handles such anomalies gracefully. In these sense, we propose three important capabilities: (1) **automated buffering and replay of data items**, which scaveges the free memory of the compute nodes hosting the producers and consumers (e.g., GPU HBM, host memory, local storage) in order to retain a history of the items accesses (produced or consumed) by the stream during normal runtime. Then, in case of restart due to failures, the stream can replay the items even if they during the previous run they were removed from the event fabric topics; (2) **flexible flow control** that matches the production rate with the consumption rate of the data items, either by elastically allocating resources to the producers and consumers (e.g., spawning more processes) or introducing additional intermediate stages to enable aggregations (e.g., group multiple overlapping diffraction patterns) or reorderings (e.g., re-arrange diffraction patterns in a mini-batch based on the overlapping pattern) to reduce the amount of data processed in successive stages; (3) **user-defined data reconstruction**, which enables users to specify a custom technique that is transparently employed to replace missing data items due to failures or stalls (e.g., interpolation, generators based on AI models, etc.).

Thus, resilient data streams enable HPC workflows to fine-tune the trade-off between performance, quality of the results, loss due to failures and poor QoS for producer-consumer patterns, while hiding the complexity of handling anomalies.

C. Resilient Data Views

Applications often require consolidated views of datasets from different streams, each with its own structure and consistency constraints. For instance, in a ptychography workflow (as described in §III), diffraction patterns used in image reconstruction can be paired with the resulting images to train generative AI models. These models, when trained appropriately, can provide faster reconstruction alternatives. However, a naive approach to building such training pipelines is complex and inefficient: it must continuously monitor incoming diffraction patterns, wait for the conventional reconstruction to produce a ground truth image, and then combine both inputs to form new training samples. This process involves numerous potential failure points that increase complexity, and it often leads to redundant data storage since training samples replicate data that may be needed by other tasks.

To address these challenges of complexity and redundancy, we propose the resilient data view abstraction, which dynamically assembles zero-copy, consistent collections of data. The core idea is to allow users to define data sources, specify relationships between items, and impose constraints (e.g., every K items in one source must pair with one item in another source). The resilient data view then operates as a meta-stream, presenting new items (through pointers, not copies) only when all constraints are met. By relying on resilient data streams (discussed in §IV-B) as the original sources, this approach effectively masks failures from users since recovery can be managed by the underlying resilient streams. Notably, if multiple data views share the same sources, repairing a single resilient stream can heal all dependent views.

Resilience is handled at multiple levels. First, data view pointers are kept up-to-date even as original items are dis-

carded (e.g., due to data expiration). This is achieved either by automatically discarding stale data view items (under the assumption that the liveness of all original data items is a precondition for the liveness of an item exposed by the data view) or by temporarily creating extra copies until all dependencies are resolved. Second, since the repair protocols of the underlying streams may be costly, they can be supplemented with checkpointing and versioning techniques. These methods capture consistent snapshots of data views, enabling independent recovery without relying on the original streams. This layered resilience ensures that data views remain robust and consistent even in dynamic, failure-prone environments.

D. Elastically Resilient Compute and Data

Part of mitigating failures and anomalies is automated elastic resource (re)allocation and task reconfiguration/migration with minimal input from the application. In this regard, we propose two novel solutions for addressing these challenges, namely, resilient compute pools and resilient dynamic data storage/replication algorithms.

Resilient compute pools: aim to address challenges related to failures occurring on compute infrastructure. This solution will allow for the construction of resource pools that work to maintain, in a resilient fashion, sets of resources with specified properties: e.g., minimum total size, a certain geographical distribution, minimum number of nodes per site. Geographically distributed tasks require a minimum resource level at each site, with resource deficits due to increased demand or reduced supply potentially compromising overall progress or leading to subtle errors such as bias due to reduced contributions from straggling training tasks. To address this issue, we have created a prototype in Parsl [80], a parallel programming library for Python, and will build upon this prototype to deliver similar capabilities in Globus Compute [81], a federated FaaS platform built on Parsl that allows users to provision compute pools on different resources (HPC, Cloud, edge) and for users to then dispatch computational tasks to remote compute pools. We have extended Parsl to send task and resource information to Octopus, and also to consume event streams when making scheduling and task retry decisions [82]. In addition to maintaining a minimum level of resource availablility, two other important aspects merit further discussion: (1) *compute elasticity*, which is essential in enabling tasks that suffered a failure or anomaly that required a restart from a checkpoint to scale-up if needed in order to recompute the lost computation faster, thus being able to catch up with the rest of the tasks; (2) *live migration of tasks*, which can be triggered by failure detection and prediction in order to proactively and seamlessly move tasks from one site to another with minimal service interruption. Both present interesting HPC-oriented opportunities for innovation (e.g., live task migration that incrementally moves dirty data structures between sites until the increment is small enough to switch over seamlessly, similar to VM live migration).

Resilient dynamic data storage: addresses the issue of keeping persistent data secure and available in a heterogeneous, multi-site configuration. We envision novel algorithms that consider factors such as I/O bandwidth saturation, failure rates, and latency constraints. In this case, classic replication and erasure coding techniques used for single HPC centers are insufficient to address more complex trade-offs arising in a multi-site configuration: due to changing conditions and competition for I/O bandwidth outside of the control of applications (e.g., shared parallel file systems), it may be necessary to dynamically switch between replication and erasure coding at potentially fine granularity (e.g., small chunks of data) to meet both resilience and high availability requirements.

E. Anomaly Detection and Prediction

Ideally, it should be possible to predict and respond to various forms of congestion, such as in the network, shared filesystem, or compute system, before user workflows are impacted. Consider the experiment data collection system in Figure 2, a simplified depiction of our motivating ptychography use case (§III).

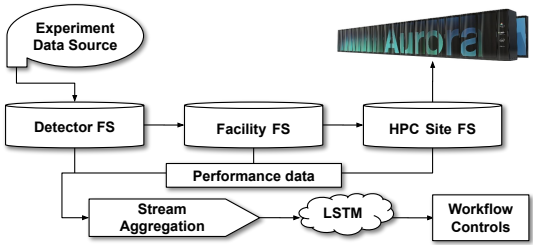

Fig. 2. Multiple-hop data transfers between edge and HPC system are monitored and streamed to an LSTM, which learns anomaly patterns and makes predictions.

Different workflows will have different performance data streams and metrics, and will also want to be able to select from emerging prediction models that may be best for their use cases. We aim here make it easy for users to train and apply arbitrary models to event streams: e.g., to infer metrics on hidden system components, make short-term predictions on individual metrics, and even generate alerts about imminent system congestion dynamics and fault-triggering conditions. To this end, we are developing system interfaces that allow the use of a simple declarative notation (e.g., as in some complex event processing systems [83], [84]) to specify the stream(s) to be monitored and the model method(s) to be applied, such as those described in §II.

In the Diaspora architecture, we can integrate such models to simple anomaly prediction problems that arise in the context of our target applications: for example, by applying ML methods to event streams to predict network degradation or reductions in available compute resources. We can also link multiple such models (corresponding, for example, to different network connections, compute resources, application components, externally observable activities) to form virtual infrastructure twins [85]: i.e., predictive networks that approximate some aspect(s) of the real infrastructure and its applications. Such networks may be able to learn usage patterns with respect to a

particular application, estimate relationships among queues, or identify sources of congestion, anomalous behavior, or points of failure. If we can thus obtain a useful representation of an entire infrastructure from just a partial view of its state, this will represent a significant advance in distributed systems.

V. Diaspora: A Comprehensive Resilience Framework

We integrate the event fabric and the resilience abstractions into *Diaspora*, a comprehensive resilience framework for real-time, distributed HPC workflows, as illustrated in Figure 3.

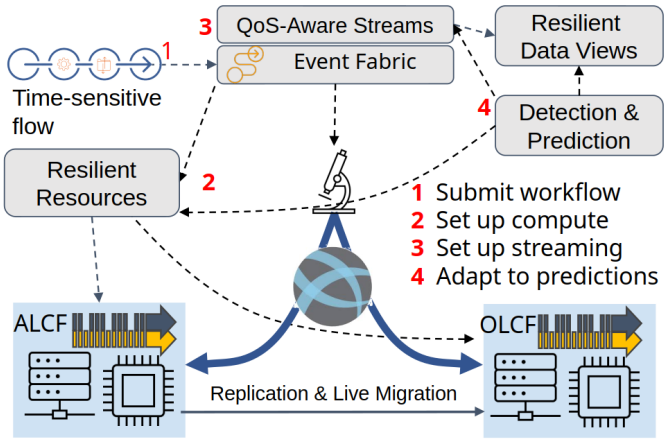

Fig. 3. Diaspora integrates the event fabric and the resilience abstractions.

Specifically, the software stack consisting of the event fabric, resilient compute and data, QoS-aware data streams, resilient data views and the detection/prediction module is deployed on both the edge infrastructure/scientific instruments, as well as the HPC data centers.

Users interact with Diaspora by following these steps: (1) submit the application workflows to a Diaspora-enabled setup; (2) configure the resilient compute and data service with policies for elastic allocation of resources, checkpointing strategy (e.g., checkpoint interval) and restart strategy (e.g., resources needed to allow computations to catch up), data availability (replication, erasure coding); (3) configure the data streaming policies by specifying a buffering strategy (e.g., sliding window), flow-control QoS (e.g., expected data rate between producers and consumers) and alternative strategies to reconstruct lost or slow data items in case of failures and anomalies (e.g., data reduction, interpolation, generative AI models); (4) configure the detection and prediction module with example triggers for alternative strategies that are used as a starting point to learn and adapt based on anomaly severity and predicted duration.

The QoS-aware data streams use the event fabric as a low-level transport mechanism for the data items. In turn, the resilient data views use several QoS-aware data streams as the data sources used in the consolidation. Furthermore, the resilience services use the event fabric for state monitoring, control messages and information exchanges in addition to the actual streamed data items. For example, the resilient compute and data service needs to constantly exchange information about resource availability and occupation between the different sites in order to enable load balancing through elasticity, replication and live migration. The detection and prediction module needs to collect and aggregate distributed monitoring information that is processed and disseminated to the rest of the components.

The development and testing are supported by virtual infrastructure twins (VIT) that emulate hosts and networks to create a software environment nearly identical to the actual so that workflow codes can be executed without change [85], [86]. They enable testing without requiring access to the physical system and can enable testing conditions beyond those of physical systems.

VI. Conclusions and Future Work

Modern scientific experiments are coupled systems with diverse data sources, variable communication patterns, and demanding computing requirements. They run as distributed HPC workflows that combine edge infrastructure (e.g., scientific instruments) with HPC data centers. In this context, real-time processing is paramount, facilitating better performance and scalability, improved resource utilization, and the ability to enable time-sensitive decision making and experimental steering. Resilience of such real-time distributed workflows is a key challenge, as it needs to extend traditional checkpoint-restart HPC approaches with support for fault tolerance of loosely coupled wokflows and support for enforcing QoS constraints. To this end, Diaspora provides a resilient framework that relies on a set of composable resilient abstractions, all of which are bound together by a streaming model that emphasizes an even-driven design. These abstractions hide away the complexity of resilience from application developers, which improves their productivity without sacrificing flexibility in choosing complex trade-offs between performance, quality of service and results, and resilience requirements.

As a next step, we plan to implement the proposed abstractions and integrate them together in a framework, which we will demonstrate with the Ptychography application and another related application that follows a similar pattern, namely Laue diffraction [7]. Other applications will be considered as follows. Sensor networks for smart cities [87] will involve complex distributed computing patterns, including calculations on edge devices and HPC resources, therefore benefiting from resilient compute resources and QoS-aware data streams. Multi-messenger astronomy [19] attempts to utilize diverse telescope and particle detector-based data sources, involving distributed instruments and data streams. This application can take advantage of resilient data views. Fusion energy experiments [88] are coupled with advanced computing resources for simulation, data management, and AI-based investigations. In this case, all components are vital in orchestrating such a complex workflow.

ACKNOWLEDGMENTS

This material is based upon work supported by the U.S. Department of Energy (DOE), Office of Science, Office of Advanced Scientific Computing Research, under Contracts DE-AC02-06CH11357 and DE-AC02-05CH11231.

REFERENCES

[1] O. Aberle, C. Adorisio, A. Adraktas, M. Ady, J. Albertone, L. Alberty, M. Alcaide Leon, A. Alekou, D. Alesini, B. Almeida Ferreira *et al.*, "High-Luminosity Large Hadron Collider (HL-LHC): Technical design report," CERN, Tech. Rep. CERN-2020-010, 2020.

[2] G. Barrand, I. Belyaev, P. Binko, M. Cattaneo, R. Chytracek, G. Corti, M. Frank, G. Gracia, J. Harvey, E. Van Herwijnen *et al.*, "GAUDI—a software architecture and framework for building HEP data processing applications," *Computer physics communications*, vol. 140, no. 1-2, pp. 45–55, 2001.

[3] N. Collier, J. M. Wozniak, A. Stevens, Y. Babuji, M. Binois, A. Fadikar, A. Würth, K. Chard, and J. Ozik, "Developing distributed high-performance computing capabilities of an open science platform for robust epidemic analysis," in *2023 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*. IEEE, 2023, pp. 868–877.

[4] P. Thibault, M. Dierolf, A. Menzel, O. Bunk, C. David, and F. Pfeiffer, "High-resolution scanning x-ray diffraction microscopy," *Science*, vol. 321, no. 5887, pp. 379–382, 2008.

[5] M. Dierolf, A. Menzel, P. Thibault, P. Schneider, C. M. Kewish, R. Wepf, O. Bunk, and F. Pfeiffer, "Ptychographic x-ray computed tomography at the nanoscale," *Nature*, vol. 467, no. 7314, pp. 436–439, 2010.

[6] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster, "TomoGAN: Low-dose synchrotron x-ray tomography with generative adversarial networks," *JOSA A*, vol. 37, no. 3, pp. 422–434, 2020.

[7] M. Prince, D. Gürsoy, D. Sheyfer, R. Chard, B. Côté, H. Parraga, B. Frosik, J. Tischler, and N. Schwarz, "Demonstrating cross-facility data processing at scale with Laue microdiffraction," in *Proceedings of the SC '23 Workshops of The International Conference on High Performance Computing, Network, Storage, and Analysis*, ser. SC-W '23. New York, NY, USA: Association for Computing Machinery, 2023, p. 2133–2139. [Online]. Available: https://doi.org/10.1145/3624062.3624613

[8] C. Wang, U. Steiner, and A. Sepe, "Synchrotron big data science," *Small*, vol. 14, no. 46, p. 1802291, 2018.

[9] I. Bird, "Computing for the Large Hadron Collider," *Annual Review of Nuclear and Particle Science*, vol. 61, no. 1, pp. 99–118, 2011.

[10] E. A. Huerta, A. Khan, X. Huang, M. Tian, M. Levental, R. Chard, W. Wei, M. Heflin, D. S. Katz, V. Kindratenko, D. Mu, B. Blaiszik, and I. Foster, "Accelerated, scalable and reproducible AI-driven gravitational wave detection," *Nature Astronomy*, vol. 5, pp. 1062–1068, Jul. 2021.

[11] N. Ravi, P. Chaturvedi, E. A. Huerta, Z. Liu, R. Chard, A. Scourtas, K. J. Schmidt, K. Chard, B. Blaiszik, and I. Foster, "FAIR principles for AI models, with a practical application for accelerated high energy diffraction microscopy," *Scientific Data*, vol. 9, no. 1, p. 657, November 2022. [Online]. Available: https://doi.org/10.1038/s41597-022-01712-9

[12] S. Steiner, J. Wolf, S. Glatzel, A. Andreou, J. M. Granda, G. Keenan, T. Hinkley, G. Aragon-Camarasa, P. J. Kitson, D. Angelone *et al.*, "Organic synthesis in a modular robotic system driven by a chemical programming language," *Science*, vol. 363, no. 6423, p. eaav2211, 2019.

[13] B. Burger, P. M. Maffettone, V. V. Gusev, C. M. Aitchison, Y. Bai, X. Wang, X. Li, B. M. Alston, B. Li, R. Clowes *et al.*, "A mobile robotic chemist," *Nature*, vol. 583, no. 7815, pp. 237–241, 2020.

[14] M. M. Flores-Leonar, L. M. Mejía-Mendoza, A. Aguilar-Granda, B. Sanchez-Lengeling, H. Tribukait, C. Amador-Bedolla, and A. Aspuru-Guzik, "Materials acceleration platforms: On the way to autonomous experimentation," *Current Opinion in Green and Sustainable Chemistry*, vol. 25, p. 100370, 2020.

[15] A. Vriza, H. Chan, and J. Xu, "Self-driving laboratory for polymer electronics," *Chemistry of Materials*, vol. 35, no. 8, pp. 3046–3056, 2023.

[16] P. W. Nega, Z. Li, V. Ghosh, J. Thapa, S. Sun, N. T. P. Hartono, M. A. N. Nellikkal, A. J. Norquist, T. Buonassisi, E. M. Chan *et al.*, "Using automated serendipity to discover how trace water promotes and inhibits lead halide perovskite crystal formation," *Applied Physics Letters*, vol. 119, no. 4, 2021.

[17] P. A. Meyer, S. Socias, J. Key, E. Ransey, E. C. Tjon, A. Buschiazzo, M. Lei, C. Botka, J. Withrow, D. Neau *et al.*, "Data publication with the Structural Biology Data Grid supports live analysis," *Nature Communications*, vol. 7, no. 1, p. 10882, 2016.

[18] R. Vescovi, R. Chard, N. D. Saint, B. Blaiszik, J. Pruyne, T. Bicer, A. Lavens, Z. Liu, M. E. Papka, S. Narayanan *et al.*, "Linking scientific instruments and computation: Patterns, technologies, and experiences," *Patterns*, vol. 3, no. 10, 2022.

[19] M. Branchesi, "Multi-messenger astronomy: gravitational waves, neutrinos, photons, and cosmic rays," in *Journal of Physics: conference series*, vol. 718, no. 2. IOP Publishing, 2016, p. 022004.

[20] E. A. Huerta, G. Allen, I. Andreoni, J. M. Antelis, E. Bachelet, G. B. Berriman, F. B. Bianco, R. Biswas, M. Carrasco Kind, K. Chard *et al.*, "Enabling real-time multi-messenger astrophysics discoveries with deep learning," *Nature Reviews Physics*, vol. 1, no. 10, pp. 600–608, 2019.

[21] P. Mészáros, D. B. Fox, C. Hanna, and K. Murase, "Multi-messenger astrophysics," *Nature Reviews Physics*, vol. 1, no. 10, pp. 585–599, 2019.

[22] E. N. M. Elnozahy, L. Alvisi, Y.-M. Wang, and D. B. Johnson, "A survey of rollback-recovery protocols in message-passing systems," *ACM Comput. Surv.*, vol. 34, no. 3, p. 375–408, sep 2002.

[23] P. Dickens and J. Logan, "Towards a high performance implementation of MPI-IO on the Lustre file system," in *OTM '08: Proceedings of the OTM 2008 Confederated International Conferences, CoopIS, DOA, GADA, IS, and ODBASE 2008. Part I on On the Move to Meaningful Internet Systems*, Monterrey, Mexico, 2008, pp. 870–885.

[24] B. Nicolae, A. Moody, E. Gonsiorowski, K. Mohror, and F. Cappello, "VeloC: Towards high performance adaptive asynchronous checkpointing at large scale," in *IPDPS'19: The 2019 IEEE International Parallel and Distributed Processing Symposium*, Rio de Janeiro, Brazil, 2019, pp. 911–920. [Online]. Available: https://hal.inria.fr/hal-02184203

[25] T. Bicer, W. Jiang, and G. Agrawal, "Supporting fault tolerance in a data-intensive computing middleware," in *2010 IEEE International Symposium on Parallel & Distributed Processing (IPDPS)*. IEEE, 2010, pp. 1–12.

[26] N. Liu, J. Cope, P. Carns, C. Carothers, R. Ross, G. Grider, A. Crume, and C. Maltzahn, "On the role of burst buffers in leadership-class storage systems," in *IEEE Conference on Massive Data Storage*, Pacific Grove, CA, Apr. 2012.

[27] B. Nicolae, "Scalable multi-versioning ordered key-value stores with persistent memory support," in *IPDPS 2022: The 36th IEEE International Parallel and Distributed Processing Symposium*, Lyon, France, 2022, pp. 93–103. [Online]. Available: https://hal.inria.fr/hal-03598396

[28] A. Maurya, M. Rafique, T. Tonellot, H. AlSalem, F. Cappello, and B. Nicolae, "GPU-enabled asynchronous multi-level checkpoint caching and prefetching," in *HPDC'23: The 32nd International Symposium on High-Performance Parallel and Distributed Computing*, Orlando, USA, 2023, pp. 73–85. [Online]. Available: https://hal.inria.fr/hal-04119928

[29] B. Nicolae, A. Moody, G. Kosinovsky, K. Mohror, and F. Cappello, "VELOC: VEry Low Overhead Checkpointing in the age of exascale," in *SuperCheck'21: The First International Symposium on Checkpointing for Supercomputing*, Virtual Event, 2021. [Online]. Available: https://arxiv.org/pdf/2103.02131.pdf

[30] M. J. Gossman, B. Nicolae, and J. C. Calhoun, "Scalable i/o aggregation for asynchronous multi-level checkpointing," *Future Generation Computer Systems*, vol. 160, pp. 420–432, 2024.

[31] B. Nicolae, T. Hobson, O. Yildiz, T. Peterka, and D. Morozov, "Towards low-overhead resilience for data parallel deep learning," in *CCGrid'22: The 22th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing*, Messina, Italy, 2022, pp. 336–345. [Online]. Available: https://hal.inria.fr/hal-03631882

[32] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: A computation management agent for multi-institutional grids," *Cluster Computing*, vol. 5, pp. 237–246, 2002.

[33] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: The Condor experience," *Concurrency and Computation: Practice and Experience*, vol. 17, no. 2-4, pp. 323–356, 2005.

[34] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *9th USENIX Conference on Networked Systems Design and Implementation*, San Jose, USA, 2012, pp. 2–2.

[35] O. Marcu, A. Costan, B. Nicolae, and G. Antoniu, "Virtual log-structured storage for high-performance streaming," in *CLUSTER'21: The 2021 IEEE International Conference on Cluster Computing*, Portland, USA, 2021, pp. 135–145. [Online]. Available: https://hal.inria.fr/hal-03300796

[36] R. Chard, J. Pruyne, K. McKee, J. Bryan, B. Raumann, R. Ananthakrishnan, K. Chard, and I. T. Foster, "Globus automation services: Research process automation across the space–time continuum," *Future Generation Computer Systems*, vol. 142, pp. 393–409, 2023.

[37] Y. Zhao, M. Wilde, I. Foster, J. Voeckler, J. Dobson, E. Gilbert, T. Jordan, and E. Quigg, "Virtual data grid middleware services for data-intensive science," *Concurrency and Computation: Practice and Experience*, vol. 18, no. 6, pp. 595–608, 2006.

[38] P. Huang, C. Guo, J. R. Lorch, L. Zhou, and Y. Dang, "Capturing and enhancing in situ system observability for failure detection," in *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*, 2018, pp. 1–16.

[39] S. Karumuri, F. Solleza, S. Zdonik, and N. Tatbul, "Towards observability data management at scale," *ACM Sigmod Record*, vol. 49, no. 4, pp. 18–23, 2021.

[40] B. Li, X. Peng, Q. Xiang, H. Wang, T. Xie, J. Sun, and X. Liu, "Enjoy your observability: an industrial survey of microservice tracing and analysis," *Empirical Software Engineering*, vol. 27, pp. 1–28, 2022.

[41] Z. Ouyang, B. Sun, W. Tang, T. Han, and K. Zhang, "Attention based Bi-LSTM for power line partial discharge fault detection," in *4th International Conference on Electronics and Electrical Engineering Technology*, ser. EEET 2021. New York, NY, USA: Association for Computing Machinery, 2022, p. 28–33. [Online]. Available: https://doi.org/10.1145/3508297.3508302

[42] W. Chen, J. Zhou, Z. Yuan, and J. Chen, "An LSTM-based online fault diagnosis method for capacitor voltage transformer," in *2021 International Conference on Mechanical, Aerospace and Automotive Engineering*, ser. CMAAE 2021. New York, NY, USA: Association for Computing Machinery, 2022, p. 347–351. [Online]. Available: https://doi.org/10.1145/3518781.3519179

[43] Y. Wang, X. Dong, L. Wang, W. Chen, and X. Zhang, "Optimizing small-sample disk fault detection based on LSTM-GAN model," *ACM Trans. Archit. Code Optim.*, vol. 19, no. 1, jan 2022. [Online]. Available: https://doi.org/10.1145/3500917

[44] A. Berhich, F.-Z. Belouadha, and M. I. Kabbaj, "LSTM-based models for earthquake prediction," in *3rd International Conference on Networking, Information Systems and Security*, ser. NISS2020. New York, NY, USA: Association for Computing Machinery, 2020. [Online]. Available: https://doi.org/10.1145/3386723.3387865

[45] I. U. Akgun, A. S. Aydin, A. Burford, M. McNeill, M. Arkhangelskiy, and E. Zadok, "Improving storage systems using machine learning," *ACM Trans. Storage*, vol. 19, no. 1, jan 2023. [Online]. Available: https://doi.org/10.1145/3568429

[46] J. Chen, S. S. Banerjee, Z. T. Kalbarczyk, and R. K. Iyer, "Machine learning for load balancing in the Linux kernel," in *11th ACM SIGOPS Asia-Pacific Workshop on Systems*, ser. APSys '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 67–74. [Online]. Available: https://doi.org/10.1145/3409963.3410492

[47] J. R. Ky, B. Mathieu, A. Lahmadi, and R. Boutaba, "Assessing unsupervised machine learning solutions for anomaly detection in cloud gaming sessions," in *18th International Conference on Network and Service Management*, ser. CNSM '22. Laxenburg, AUT: International Federation for Information Processing, 2023.

[48] E. Brewer, "CAP twelve years later: How the 'rules' have changed," *Computer*, vol. 45, no. 2, pp. 23–29, 2012.

[49] S. Gilbert and N. Lynch, "Perspectives on the CAP theorem," *Computer*, vol. 45, no. 2, pp. 30–36, 2012.

[50] J. Kreps *et al.*, "Kafka: A distributed messaging system for log processing," in *Proceedings of the NetDB*, vol. 11, no. 2011. Athens, Greece, 2011, pp. 1–7.

[51] M. Ott, W. Shin, N. Bourassa, T. Wilde, S. Ceballos, M. Romanus, and N. Bates, "Global experiences with HPC operational data measurement, collection and analysis," in *2020 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 2020, pp. 499–508.

[52] J. Thaler, W. Shin, S. Roberts, J. H. Rogers, and T. Rosedahl, "Hybrid approach to HPC cluster telemetry and hardware log analytics," in *2020 IEEE High Performance Extreme Computing Conference (HPEC)*. IEEE, 2020, pp. 1–7.

[53] L. Ramakrishnan, P. T. Zbiegel, S. Campbell, R. Bradshaw, R. S. Canon, S. Coghlan, I. Sakrejda, N. Desai, T. Declerck, and A. Liu, "Magellan: experiences from a science cloud," in *Proceedings of the 2nd international workshop on Scientific cloud computing*, 2011, pp. 49–58.

[54] J.-S. Vöckler, G. Juve, E. Deelman, M. Rynge, and B. Berriman, "Experiences using cloud computing for a scientific workflow application," in *Proceedings of the 2nd international workshop on Scientific cloud computing*, 2011, pp. 15–24.

[55] C. L. Gentemann, C. Holdgraf, R. Abernathey, D. Crichton, J. Colliander, E. J. Kearns, Y. Panda, and R. P. Signell, "Science storms the cloud," *AGU Advances*, vol. 2, no. 2, p. e2020AV000354, 2021.

[56] E. Gaussier, D. Glesser, V. Reis, and D. Trystram, "Improving backfilling by using machine learning to predict running times," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, 2015, pp. 1–10.

[57] W. Smith, V. Taylor, and I. Foster, "Using run-time predictions to estimate queue wait times and improve scheduler performance," in *Job Scheduling Strategies for Parallel Processing: IPPS/SPDP'99Workshop, JSSPP'99 San Juan, Puerto Rico, April 16, 1999 Proceedings 5*. Springer, 1999, pp. 202–219.

[58] G. P. Rodrigo, P.-O. Östberg, E. Elmroth, K. Antypas, R. Gerber, and L. Ramakrishnan, "Towards understanding HPC users and systems: a NERSC case study," *Journal of Parallel and Distributed Computing*, vol. 111, pp. 206–221, 2018.

[59] W. Smith, I. Foster, and V. Taylor, "Predicting application run times with historical information," *Journal of Parallel and Distributed Computing*, vol. 64, no. 9, pp. 1007–1016, 2004.

[60] A. Matsunaga and J. A. Fortes, "On the use of machine learning to predict the time and resources consumed by applications," in *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*. IEEE, 2010, pp. 495–504.

[61] C. Guok, D. Robertson, M. Thompson, J. Lee, B. Tierney, and W. Johnston, "Intra and interdomain circuit provisioning using the OSCARS reservation system," in *2006 3rd International Conference on Broadband Communications, Networks and Systems*. IEEE, 2006, pp. 1–8.

[62] X. Yu, V. Nikitin, D. J. Ching, S. Aslan, D. Gürsoy, and T. Biçer, "Scalable and accurate multi-GPU-based image reconstruction of large-scale ptychography data," *Scientific reports*, vol. 12, no. 1, p. 5334, 2022.

[63] M. Hidayetoğlu, T. Bicer, S. G. De Gonzalo, B. Ren, V. De Andrade, D. Gursoy, R. Kettimuthu, I. T. Foster, and W. H. Wen-mei, "Petascale XCT: 3D image reconstruction with hierarchical communications on multi-GPU nodes," in *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 2020, pp. 1–13.

[64] J. Liu, B. Nicolae, D. Li, J. M. Wozniak, T. Bicer, Z. Liu, and I. Foster, "Large scale caching and streaming of training data for online deep learning," in *Proceedings of the 12th Workshop on AI and Scientific Computing at Scale using Flexible Computing Infrastructures*, 2022, pp. 19–26.

[65] Z. Liu, T. Bicer, R. Kettimuthu, D. Gursoy, F. De Carlo, and I. Foster, "TomoGAN: Low-dose x-ray tomography with generative adversarial networks," in *arXiv preprint arXiv:1902.07582*, 2019.

[66] A. V. Babu, T. Zhou, S. Kandel, T. Bicer, Z. Liu, W. Judge, D. J. Ching, Y. Jiang, S. Veseli, S. Henke *et al.*, "Deep learning at the edge enables real-time streaming ptychographic imaging," *Nature Communications*, vol. 14, no. 1, p. 7059, 2023.

[67] T. Bouvier, B. Nicolae, A. Costan, T. Bicer, I. Foster, and G. Antoniu, "Efficient distributed continual learning for steering experiments in real-time," *Future Generation Computer Systems*, 2024.

[68] C. Benmore, T. Bicer, M. K. Chan, Z. Di, D. a. Gürsoy, I. Hwang, N. Kuklev, D. Lin, Z. Liu, I. Lobach *et al.*, "Advancing ai/ml at the advanced photon source," *Synchrotron Radiation News*, vol. 35, no. 4, pp. 28–35, 2022.

[69] T. Bicer, X. Yu, D. J. Ching, R. Chard, M. J. Cherukara, B. Nicolae, R. Kettimuthu, and I. T. Foster, "High-performance ptychographic reconstruction with federated facilities," in *Smoky Mountains Computational Sciences and Engineering Conference*. Springer International Publishing Cham, 2021, pp. 173–189.

[70] M. Hidayetoğlu, T. Biçer, S. G. de Gonzalo, B. Ren, D. Gürsoy, R. Kettimuthu, I. T. Foster, and W.-m. W. Hwu, "Memxct: Memory-centric x-ray ct reconstruction with massive parallelization," in *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*. ACM, 2019, p. 85.

[71] X. Yu, T. Bicer, R. Kettimuthu, and I. Foster, "Topology-aware optimizations for multi-GPU ptychographic image reconstruction," in *Proceedings of the ACM International Conference on Supercomputing*, 2021, pp. 354–366.

[72] B. Ma, V. Nikitin, D. Li, and T. Bicer, "Accelerated laminographic image reconstruction using GPUs," *Electronic Imaging*, vol. 36, pp. 1–6, 2024.

[73] T. Bicer, D. Gürsoy, V. D. Andrade, R. Kettimuthu, W. Scullin, F. D. Carlo, and I. T. Foster, "Trace: a high-throughput tomographic reconstruction engine for large-scale datasets," *Advanced structural and chemical imaging*, vol. 3, pp. 1–10, 2017.

[74] M. J. Cherukara, T. Zhou, Y. Nashed, P. Enfedaque, A. Hexemer, R. J. Harder, and M. V. Holt, "AI-enabled high-resolution scanning coherent diffraction imaging," *Applied Physics Letters*, vol. 117, no. 4, 2020.

[75] M. J. Cherukara, Y. S. Nashed, and R. J. Harder, "Real-time coherent diffraction inversion using deep generative networks," *Scientific reports*, vol. 8, no. 1, p. 16520, 2018.

[76] A. V. Babu, T. Bicer, S. Kandel, T. Zhou, D. J. Ching, S. Henke, S. Veseli, R. Chard, A. Miceli, and M. J. Cherukara, "AI-assisted automated workflow for real-time x-ray ptychography data analysis via federated resources," *Electronic Imaging*, vol. 35, no. 11, pp. 1–6, 2023.

[77] A. V Babu, Z. Liu, T. Bicer, and S. Veseli, "Real-time edge inference for ptychographic nanoscale xray imaging," Argonne National Lab.(ANL), Argonne, IL (United States), Tech. Rep., 2021.

[78] R. B. Ross, G. Amvrosiadis, P. H. Carns, C. D. Cranor, and M. Dorier, "Mochi: Composing data services for high-performance computing environments," *Journal of Computer Science and Technology*, vol. 35, pp. 121 – 144, 2020.

[79] S. Tuecke, R. Ananthakrishnan, K. Chard, M. Lidman, B. McCollam, S. Rosen, and I. Foster, "Globus Auth: A research identity and access management platform," in *2016 IEEE 12th International Conference on e-Science (e-Science)*, 2016, pp. 203–212.

[80] Y. Babuji, A. Woodard, Z. Li, D. S. Katz, B. Clifford, R. Kumar, L. Lacinski, R. Chard, J. M. Wozniak, I. Foster, M. Wilde, and K. Chard, "Parsl: Pervasive parallel programming in Python," in *Proceedings of the 28th International Symposium on High-Performance Parallel and Distributed Computing*, ser. HPDC '19. New York, NY, USA: Association for Computing Machinery, 2019, p. 25–36. [Online]. Available: https://doi.org/10.1145/3307681.3325400

[81] R. Chard, Y. Babuji, Z. Li, T. Skluzacek, A. Woodard, B. Blaiszik, I. Foster, and K. Chard, "funcX: A federated function serving fabric for science," in *Proceedings of the 29th International symposium on high-performance parallel and distributed computing*, 2020, pp. 65–76.

[82] H. Pan, R. Chard, S. Zhou, A. Kamatar, R. Vescovi, V. Hayot-Sasson, A. Bauer, M. Gonthier, K. Chard, and I. Foster, "Octopus: Experiences with a hybrid event-driven architecture for distributed scientific computing," 2024. [Online]. Available: https://arxiv.org/abs/2407.11432

[83] D. Luckham, *The Power of Events: An Introduction to Complex Event Processing in Distributed Enterprise Systems*. Addison-Wesley Reading, 2002.

[84] G. Cugola and A. Margara, "Processing flows of information: From data stream to complex event processing," *ACM Computing Surveys*, vol. 44, no. 3, pp. 1–62, 2012.

[85] A. Al-Najjar and N. S. V. Rao, "Virtual Infrastructure Twin for computing-instrument ecosystems: Software and measurements," *IEEE Access*, pp. 1–1, 2023.

[86] N. S. V. Rao, A. Al-Najjar, , H. Zandi, R. Sankaran, S. E. Hicks, K. Roccapriori, and D. Mukherjee, "Virtual infrastructure twins: Software testing platforms for computing-instrument ecosystems," in *Proceedings of Smoky Moutains Computational Sciences and Engineering Conference*, D. Kothe, A. Geist, S. Pophale, H. Liu, and S. Parete-Koon, Eds. Springer, 2022.

[87] Y. Kim, S. Park, S. Shahkarami, R. Sankaran, N. Ferrier, and P. Beckman, "Goal-driven scheduling model in edge computing for smart city applications," *Journal of Parallel and Distributed Computing*, vol. 167, pp. 97–108, 2022.

[88] S. Smith, E. Belli, O. Meneghini, R. Budiardja, D. Schissel, J. Candy, T. Neiser, and A. Eubanks, "A vision for coupling operation of US fusion facilities with HPC systems and the implications for workflows and data management," in *Accelerating Science and Engineering Discoveries Through Integrated Research Infrastructure for Experiment, Big Data, Modeling and Simulation: 22nd Smoky Mountains Computational Sciences and Engineering Conference*. Springer, 2023, pp. 87–100.