Building Resilient Self-Driving Laboratories with the INTERSECT Federated Ecosystem

Christian Engelmann, Andrew Ayres, Stephen DeWitt, Michael J. Brim, and Brett Eiffert
Oak Ridge National Laboratory, Oak Ridge, TN, USA
engelmannc@ornl.gov, ayresaf@ornl.gov, dewittsj@ornl.gov, brimmj@ornl.gov, and eiffertbc@ornl.gov

***Abstract*—Failure resilience in federated ecosystems for instrument science presents a critical challenge. Failures disrupt experiments and make them potentially useless, wasting valuable resources and creating setbacks. Oak Ridge National Laboratory's Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) offers a federated ecosystem for instrument science, enabling autonomous experiments, self-driving laboratories, smart manufacturing, and artificial intelligence (AI)-driven design, discovery, and evaluation. This paper documents the recent advances in creating a resilient INTERSECT ecosystem. The proposed solution includes a resilient architecture with resilience design patterns, a resilient system of systems (SoS) architecture, and a resilient microservices architecture; and a resilient software development kit with reliable service communication and asynchronous and synchronous failure detection and notification. The resilience capabilities are demonstrated for an autonomous additive manufacturing process with a real-time feedback loop.**

***Index Terms*—failure resilience, self-driving laboratories, federated systems, instrument-science workflows**

I. Introduction

Modern scientific research uses modeling, simulation, data analytics, and artificial intelligence (AI) to predict and understand experiment results. The automation of such processes reduces human-in-the-loop needs to the point where future laboratories will be "self-driving" [1]. The human scientist defines the hypothesis to be tested and the "robot scientist" designs the experiments to be performed and then performs/steers the experiments and analyzes their results. This requires federating the various involved instruments and computing/data resources and orchestrating corresponding workflows.

Resilience (the ability keep applications, services, and workflows running to a correct solution in a timely and cost-efficient manner despite frequent faults, errors, and failures of various types) in federated ecosystems for instrument science presents a critical challenge. Failures disrupt experiments and make them potentially useless, wasting valuable instrument, network, and computing allocations, and creating setbacks for scientists. Oak Ridge National Laboratory's (ORNL's) Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) [2]–[7] offers a federated ecosystem for instrument science, enabling autonomous experiments, self-driving laboratories, smart manufacturing, and AI-driven design, discovery, and evaluation. INTERSECT experiments require failure resilience, such as to reliably steer an autonomous additive manufacturing (AAM) process using real-time data streamed to a simulation in the feedback loop.

This paper documents recent advances in creating a resilient INTERSECT ecosystem and in demonstrating its capabilities with the INTERSECT AAM cross-facility experiment between the ORNL Manufacturing Demonstration Facility (MDF) and the Oak Ridge Leadership Computing Facility (OLCF) Advanced Computing Ecosystem (ACE) testbed. It makes the following novel contributions:

- A resilient INTERSECT federated ecosystem architecture that provides an implementation blueprint using resilience design patterns, a resilient system of systems (SoS) architecture, and a resilient microservices architecture.
- A resilient INTERSECT software development kit (SDK) that ensures reliable service communication and asynchronous and synchronous failure detection and notification.
- A demonstration of the INTERSECT AAM cross-facility experiment that offers fail-stop and fail-over resilience capabilities of a real-time feedback loop using the resilient INTERSECT ecosystem architecture and SDK.

The paper is structured as follows. Section II outlines the current state of instrument-science ecosystems. Section III describes the resilience challenge in federated instrument-science ecosystems, such as INTERSECT. Section IV details the technical approach, including the resilient INTERSECT architecture and SDK. Section V shows the demonstration of resilience capabilities for the INTERSECT AAM cross-facility experiment. Section VI concludes the paper.

II. Current State of the Field

The National Energy Research Scientific Computing Center Superfacility framework [8] offers access to common supercomputer functions using a representational state transfer based application programming interface [9], enabling use cases, such as connecting the Linac Coherent Light Source via ESnet to the Cori supercomputer for photosynthesis re-

This manuscript has been authored by UT-Battelle, LLC under Contract No. DE-AC05-00OR22725 with the U.S. Department of Energy. The United States Government retains and the publisher, by accepting the article for publication, acknowledges that the United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes. The Department of Energy will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

SC26 Workshops, November 15-20, 2026, Chicago, Illinois, USA
979-8-3195-1221-5/26/$31.00 ©2026 IEEE

search [10]. It currently has no support for failure resilience. The Integrated Research Infrastructure (IRI) Architecture Blueprint Activity [11] organized by the U.S. Department of Energy (DoE) defined the patterns for a cross-facility application programming interface (API). The resulting IRI API [12] that is now part of the DoE Genesis Mission platform under the American Science Cloud leverages and extends the Superfacility API, but without resilience features.

Nexus is Argonne National Laboratory's (ANL's) distributed ecosystem for supporting instrument science workflows. It is based on data transfer and workflow tools developed by ANL and the University of Chicago, such as Globus Data Transfer, Data Sharing, Compute, and Research Automation [13]. They permit automated analysis of instrument data, such as by connecting ANL's Advanced Photon Source with the Theta super-computer [14]. Nexus has some failure resilience in its tools, such as to data loss and corruption in Globus Data Transfer and to task failure in Globus Research Automation (with user-defined responses). However, there is no holistic resilience approach across these tools and there is no consistent design methodology that allows to pick and choose the right solution for the resilience problem at hand.

Workflow tools, such as Pegasus, are used in experiment and data science workflows and they do provide similar task-based failure resilience [15]. There is some existing work in failure resilience for the design of SoS solutions, but it mostly focuses on classic reliability modeling of dependent systems [16], [17]. There is also a wealth of work in resilient distributed systems. Prior efforts on resilience design patterns for high-performance computing (HPC) [18]–[21] is leveraged in this work. The INTERSECT federated ecosystem relies on microservices, a technology used in production environments, such as by Netflix [22], [23]. This work leverages existing work in microservice failure monitoring, failure tolerance, failure isolation, redundancy and replication, scalability and elasticity, and decentralized communication [24]. While there are ad-hoc resilience features for individual workflows and workflow tools, and production microservice resilience capabilities, there is no holistic resilient federated ecosystem solution for instrument-science workflows.

III. Problem Statement

Failures can take different forms, such as outages or performance/functionality degradation. They have also different root causes and frequencies. Failures of essential hardware components, such as a sensor in an instrument, a hard drive in an instrument controller, or a general-purpose computing graphics processing unit in a compute resource, are rather rare but do exist. Due to the shared nature of the distributed federated ecosystem, failures of essential shared hardware or software components, such as those caused by oversubscription of a shared network or compute resource, are more likely. For example, data delivery may get delayed due to high cross-facility network usage, or computation in the feedback loop may be slower due to shared file system or high-speed network contention. There is also the challenge of scalability, since resource contention is more likely with more workflows utilizing the shared federated ecosystem. While the federated ecosystem can adapt by adding more resources, such adaptation does not happen instantly if resources are not available or it requires preventative failure mitigation if they are. Failures also have distinctive impact scopes, depending on location and containment. The following failure impact scopes may exist:

- A resource exposed by a microservice, e.g., a supercomputer file system or an instrument sensor.
- A microservice exposing resources, e.g., a supercomputer data storage or instrument sensor data microservice.
- A service grouping microservices, e.g., a supercomputer storage management or data transfer service.
- An infrastructure system with its subsystems, services, and microservices, e.g., a supercomputer or instrument.

Depending on failure type, location, impact, and mitigation, failure propagation can be limited by proper containment. For example, a failed microservice may be just restarted using a prior checkpoint of its state. Some failure scenarios are more complex and involve more elaborate mitigation strategies. For example, the failure of a supercomputer parallel file system will not only make the corresponding data storage microservice, storage management service, and data management subsystem fail, but it will also impact the compute microservices and services that are part of the supercomputer infrastructure system. Any computation on the supercomputer's infrastructure system would fail and could significantly impact ongoing experiments. Mitigation of failures may include the following actions involving impacted resources, microservices, services, infrastructure systems, or workflows:

- Fail-stop, i.e., an orderly stop of operation;
- Restructuring, reinitializing, rejuvenating, or restarting;
- Rollback or rollforward;
- Failing over to a cold, warm, or hot standby;
- Employing full redundancy; and
- Utilizing different versions or implementations.

This is not an exhaustive list of failure mitigation options. Each solution provides a certain level of resilience, i.e., availability, reliability, and performance under failure. Each also incurs its own cost, such as performance under failure-free conditions and the need for additional resources, and offers a specific degree of containment. Failure mitigation is either performed completely transparently to systems, services, or microservices outside the containment boundary, or impacts the workflows and other systems, services, or microservices using them. A breach of failure containment requires further mitigation that creates a new containment boundary. Due to the involved complexities of failure detection, notification, containment, and mitigation, resilience needs to be an integral part of the federated ecosystem through co-design, such that the burden for providing resilience is on the system by design and not on the operator or user as an afterthought.

IV. Technical Approach

The technical approach of the proposed solution is twofold: (1) a resilient INTERSECT architecture with resilience design patterns, a resilient SoS architecture, and a resilient microservices architecture; and (2) a resilient INTERSECT SDK with reliable service communication and asynchronous and synchronous failure detection and notification.

A. Resilient INTERSECT Architecture

The resilient INTERSECT architecture consists of resilience design patterns, a resilient SoS architecture, and a resilient microservices architecture. It offers a consistent design methodology for a resilient federated instrument-science ecosystem that allows scientists to pick and choose the right solution for the resilience problem at hand and deploy it with ease. The following provides a brief overview of the existing INTERSECT architecture and details the new contributions.

1) Interconnected Science Ecosystem Architecture: The INTERSECT architecture [2], [4] is prior work and summarized here. It uses the U.S. Department of Defense (DoD) Architecture Framework as a guideline [25], which documents an architecture through different viewpoints: (i) operational scenarios, (ii) composition, interconnectivity and context, (iii) services and their capabilities, (iv) policies, standards and guidance, and (v) capability. The INTERSECT architecture approach separates these views into three parts:

- At a coarse-grained abstraction, "self-driving" laboratories are illustrated as science use case design patterns [3], [4] that identify and abstract the involved components and their interactions in terms of control, work and data flow.
- At a medium-grain level, a SoS architecture [4] decomposes the federated hardware/software ecosystem into individual systems and their components. Similar to the DoD Architecture Framework, it offers different architectural viewpoints: logical, operational, user, data, physical, and standards.
- At fine granularity, a microservices architecture [4], [26] uses loosely coupled microservices and uniform interfaces. It defines microservice interaction patterns, a classification of infrastructure and experiment microservices, and microservice orchestration and deployment.

2) INTERSECT Resilience Design Patterns: The resilience design patterns for the INTERSECT federated ecosystem extend prior work on resilience design patterns for HPC [18]–[21]. The existing concept, which only targets HPC system services and parallel HPC applications, is applied to INTERSECT resources, microservices, services, and infrastructure systems and to instrument- and computational-science workflows. The INTERSECT resilience design patterns identify and evaluate the repeatedly occurring resilience problems in the federated ecosystem and coordinate solutions using a proven software engineering approach. They permit resilience to become an integral part of the ecosystem through co-design, such that the burden for providing resilience is on the system by design and not on the operator or user as an afterthought. They offer a consistent design methodology that allows to pick and choose the right solution for the resilience problem at hand.

The prior work provides a classification of 31 resilience design patterns (Fig. 1), each with its own formal description, control flow chart, state diagram, and mathematical models for performance, reliability, and availability. It also offers a modeling tool to understand performance, reliability and, availability with different pattern parameters and combinations [21]. While it focused on system services and computation in parallel computing systems, the work presented in this paper targets workflows in distributed systems. However, the core concepts and therefore the patterns are similar.

INTERSECT follows an architectural approach of a federated SoS architecture, where systems have services and services consist of microservices. This distributed service-level approach is quite analogous to the system services in an HPC system. Solutions, such as rollback recovery or redundancy, are the same and can be reused. The aspect of workflows between a distributed set of services is new, but comparable solutions exist, such as fail-over to a redundant data link between services or to a redundant service. This work formalizes these resilience approaches for the federated instrument-science ecosystem.

Out of 31 existing patterns, 12 structural patterns (which provide concrete descriptions of applicable solutions rather than the high-level strategies of architectural and strategy patterns) have been adopted in the INTERSECT SoS architecture's logical view (rectangular selection in Fig. 1). This includes their formal description, control flow chart, etc.. They represent the resilience capabilities supported in the federated ecosystem, such as system monitoring, active/standby, and n-modular redundancy. Each pattern has also been extended with a workflow diagram to enable resilient orchestration, e.g., active/standby (Fig. 2) and n-modular redundancy (Fig. 3).

A specific solution may require the composition of patterns, such as to provide different containment and mitigation for different failures. For example, the active/standby pattern may be used for a fail-over capability from an active to a standby system in case the active system fails. The rollback pattern may be additionally used for restart capability for a service running on the active system in case the service fails but the active system does not. The patterns provide concrete descriptions of individual resilience solutions that are applied to systems, subsystems, and services in the INTERSECT SoS architecture, and correspondingly to the microservices in the INTERSECT microservices architecture that make up services.

3) INTERSECT Resilient System of Systems Architecture: The resilient SoS architecture employs the resilience design patterns for services and infrastructure systems and the resilient workflow patterns in the orchestration system. The different views of the SoS architecture are adapted to support resilience. The SoS architecture's logical view decomposes the overall ecosystem into infrastructure and logical systems, subsystems, and services. It offers a classification of fault, error and failure types and scopes, and a catalog of corresponding notification, mitigation, and containment capabilities using the

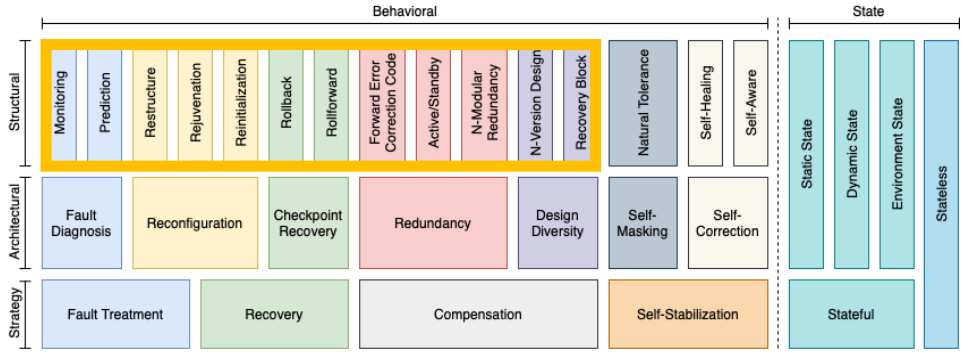


Fig. 1. The HPC resilience design patterns [19] leveraged in the INTERSECT resilience design patterns subset (rectangular selection).

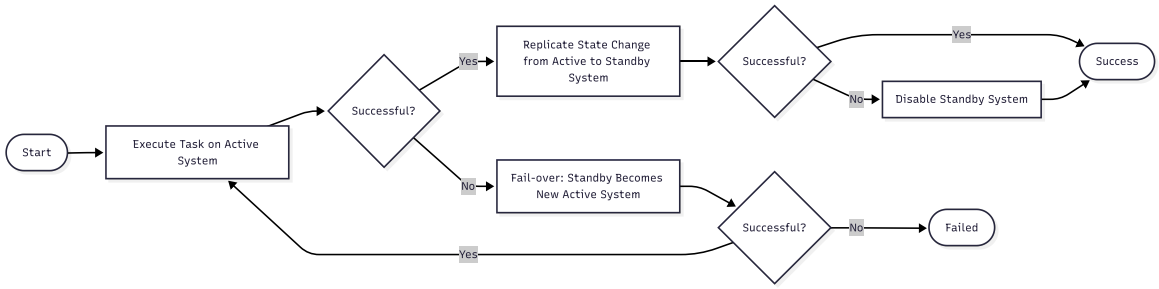


Fig. 2. The active/standby workflow pattern can execute a standby task, which may be located on another resource, when the active task fails.

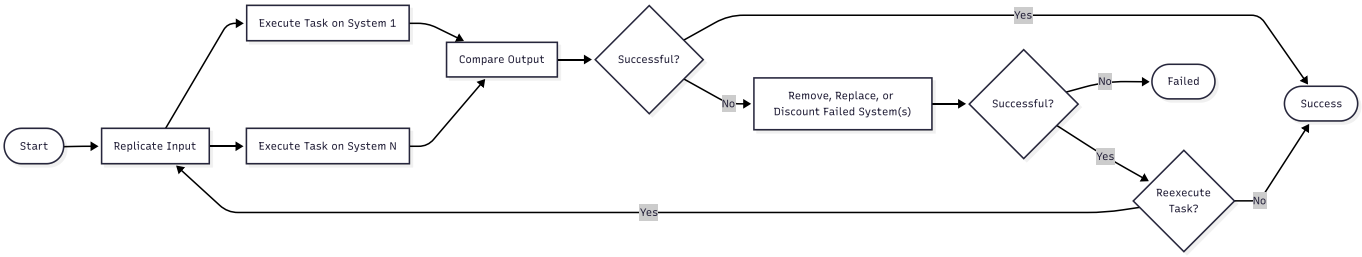


Fig. 3. The n-modular redundancy workflow pattern executes redundant tasks, which may be located on different resources, and compares the outputs.

resilience design patterns. It also defines the orchestration system, which employs resilient workflow patterns of the resilience design patterns for mitigation of task failures. The resilient microservices architecture, detailed in the following, further decomposes services into microservices. The decomposition approach of the INTERSECT architecture offers the unique capability to apply the resilience design patterns at three different scopes: systems (subsystems), services, and microservices. Each resilience design pattern in the logical view has been extended with scope context and application descriptions. This is especially relevant when composing patterns for complete solutions that span scopes.

4) INTERSECT Resilient Microservices Architecture: The INTERSECT microservices architecture provides a catalog of infrastructure and experiment-specific microservices that may be useful within an interconnected science ecosystem. All microservices are defined to facilitate composition within federated SoS architectures, where each system corresponds to one or more microservices coordinating through well-defined microservice interaction patterns. Infrastructure microservices represent common service functionality and capabilities, such as data management, computing, system management, and workflow orchestration that are likely to be generally useful across many science ecosystems without the need for customization. Experiment-specific microservices represent services whose implementation may require detailed application knowledge, such as experiment planning or steering services that require knowledge of experiment-specific control parameters and their associated constraints.

The resilient INTERSECT microservices architecture leverages the resilient INTERSECT SoS architecture by using the logical view's classification of fault, error and failure types and resilience design patterns in the microservice scope. It entirely relies on the corresponding descriptions of the SoS architecture to avoid duplication and inconsistencies. The resilience design patterns are essentially leveraged as universally applicable templates applied to the microservice scope.

The resilient INTERSECT microservices architecture also defines a basic retry pattern [27] for handling both transient microservice interaction errors and external operation failures. In this pattern, an action is retried after experiencing an error or failure in anticipation of eventual success. Often, the Retry

pattern involves the use of two parameters: a retry count and a retry delay. The retry count is the maximum number of failed attempts before the failure condition is treated as permanent. The retry delay is the interval between successive action attempts. Some implementations of this pattern may utilize a progressive delay (e.g., exponential backoff) to increase the delay for each observed failure.

B. Resilient INTERSECT Software Development Kit

The resilient INTERSECT SDK has three components: (1) a resilient message broker that ensures reliable communication despite broker service failures, (2) a watchdog service that asynchronously detects service failures using a heartbeat and notifies interested services, and (3) a communication timeout that synchronously detects service failures and notifies a service waiting for a response.

1) INTERSECT Resilient Message Broker: The INTERSECT federated ecosystem utilizes RabbitMQ as a message broker to facilitate communication between INTERSECT services that may reside on scientific instruments, edge computing devices, and HPC systems. Its service availability and capability to route messages despite individual message broker service failures is imperative for any resilience capability in the INTERSECT ecosystem. The resilient SDK employs a clustered RabbitMQ setup [28] (Fig. 4), which groups multiple message broker services across different computing systems and shares their distributed state among them. A failure of a message broker service triggers a fail-over to another service within the group. This ensures service availability and the routing of messages despite failures.

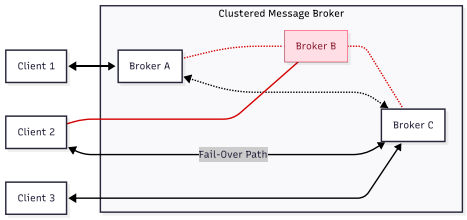


Fig. 4. The INTERSECT SDK's clustered message broker ensures reliable message delivery despite broker service failures. In this example, clustered RabbitMQ Broker C takes over for failing Broker B.

2) INTERSECT Watchdog Service: Services in the INTERSECT ecosystem communicate by messages that are transmitted via the broker service. There is no direct connection between services that would use a TCP heartbeat to detect a failure. Services also communicate with each other only when needed, so a failure may only be detected synchronously when it does not respond. To enable reasonably fast responses to failures, the resilient SDK leverages a configurable watchdog service that asynchronously detects failures using a heartbeat mechanism (Fig. 5). The watchdog sends heartbeat messages via the broker to individual services in a configurable interval and waits for the response with configurable timeouts. When a response from a service does not arrive, then it is considered failed and late responses are discarded. A detected failure triggers an immediate notification message to a configurable destination service that then handles it. The watchdog solution ensures that failures are detected within a known time frame, even if a service is not communicating with other services.

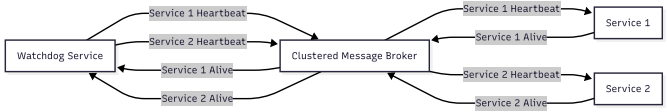


Fig. 5. The INTERSECT SDK's watchdog service sends out individual heartbeat messages in regular intervals to specific detect service outages. Messages are routed though the clustered RabbitMQ message broker.

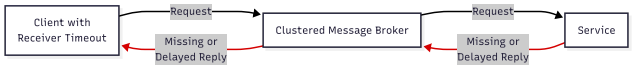


Fig. 6. The INTERSECT SDK's communication timeout mechanism waits for a reply from a service within a configurable time window to detect an outage. Messages are routed though the clustered RabbitMQ message broker.

3) INTERSECT Communication Timeout Mechanism: Services may communicate irregularly with each other, sometimes frequently and sometimes rarely. Such bursty communication is a challenge for the watchdog solution in terms of heartbeat interval frequency. Too short intervals interfere with regular communication. Too long intervals lead to delays in failure detection. The resilient SDK provides the capability to individually configure communication timeouts for synchronous detection (Fig. 6). A service waits for a response from another service within the configured timeout. When it does not arrive, then the other service is considered failed and late responses are discarded. The service waiting for the response is promptly notified. This solution permits the use of the watchdog service with longer heartbeat intervals for asynchronous detection and the utilization of shorter communication timeouts for synchronous detection, enabling a balanced approach.

V. Resilient INTERSECT Autonomous Additive Manufacturing Experiment

The goal of this work is to demonstrate failure resilience capabilities for a cross-facility autonomous additive manufacturing workflow with a real-time feedback loop that survives certain types of hardware and software failures, such as a service outage. The federated INTERSECT AAM capability (Fig. 7) is being developed to perform real-time control of the thermal history and residual stress for metal 3D metal printed components. In the 3D printer, a laser is depositing metal, layer by layer, by heating up metal powder that is blown over the laser and melting it. Printing a layer takes about 2 minutes and the deposited molten material must not heat up or cool down the underlying material too much to avoid the build up of stresses that could bend or crack it or make it otherwise structurally weak. The experiment live streams infra-red camera images and thermocouple measurement data between science facilities from the 3D printer at ORNL's MDF to a thermo-mechanical simulation [29], [30] running

on the OLCF ACE testbed's Defiant-2 computing system [31] 9 miles (or 14.5 km) away. The simulation performs faster than real-time estimates about the stress in the 3D printed object. The simulation's output is used as feedback to modulate the real time aspects of the printing process, such as the laser's power and toolpath. This permits building parts that are ready and safe to use immediately. Once the printing process is finished, the residual stress in the part and what it can and cannot do structurally is known. A detailed description of the INTERSECT AAM approach will be discussed in a separate, upcoming publication. A virtual version of the AAM workflow, where the 3D printer is replaced with a thermal simulation and synthetic infra-red images are created from surface temperatures, has been used demonstrate multi-facility agentic workflows [32] and the PROV-AGENT provenance model [33]. In this work we also use the virtual INTERSECT AAM workflow as our demonstration case, albeit without the agentic features used in Refs. [32], [33].

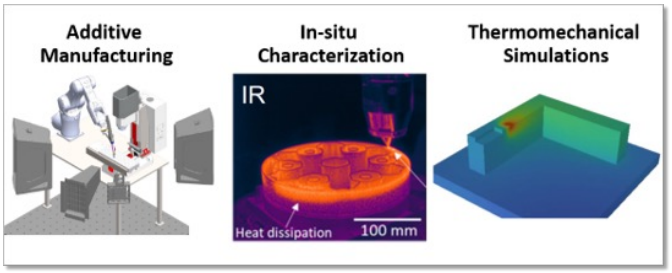


Fig. 7. The AAM experiment performs 3D metal printing with in-situ observations and thermo-mechanical simulations as live feedback to modulate the printing process and ensure parts are safe to use in critical applications.

A. Service Failure Resilience

Resilience capabilities are required, as a failure of a component in the live feedback loop (Fig. 8), such as the thermo-mechanical simulation or the control/data path to the OLCF, could severely degrade the quality of the printed product or even stop the entire printing process. The INTERSECT SDK's resilient message broker is employed to ensure a reliable communication path. The INTERSECT SDK's watchdog service is utilized to detect service failures using a heartbeat mechanism. The INTERSECT SDK's communication timeout mechanism is leveraged to detect service failures as well. A basic resilience capability is provided using the watchdog service and communication timeout mechanism to trigger a fail-stop with an orderly shutdown of the printing process.

A more advanced resilience feature is offered using the SoS architecture's active/standby resilience design pattern (Fig. 9). The simulation is executed redundantly on a different computing system. Upon the detection of a failure of the active simulation service on the ACE testbed, a fail-over is performed to a standby simulation service. A proxy service facilitates a seamless fail-over with no modification of the original simulation service. The proxy has the same service interface and acts as a virtual simulation service, hiding simulation service redundancy and the fail-over mechanism. It essentially

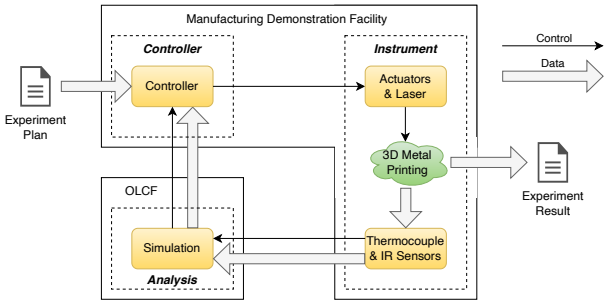


Fig. 8. The INTERSECT AAM experiment's workflow without resilience. A failure in the feedback loop could degrade product quality or stop the printing.

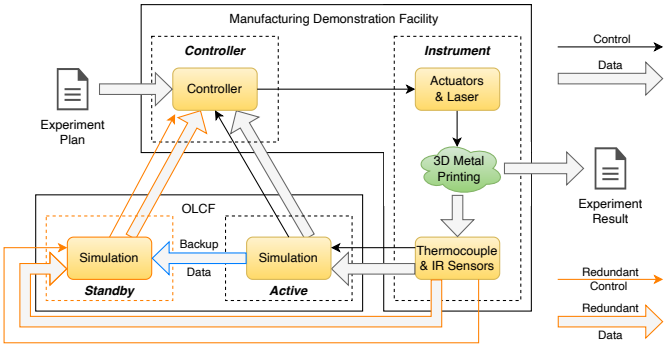


Fig. 9. Example of an INTERSECT AAM experiment's workflow with resilience capabilities using the SoS architecture's active/standby resilience design pattern. A failure in the feedback loop causes a fail-over to the standby.

creates a "sphere of replication". Replicated services inside the sphere are resilient, while the proxy itself is still a single point of failure at the sphere boundary. In future work, the proxy service can be made resilient through redundancy as well [34], essentially moving the sphere boundary to clients.

There are several variations of this resilience solution. The standby simulation may be an exact replica of the active simulation, receiving the same input and producing the same output. It also may be a low-fidelity simulation using less computing resources than the active simulation, receiving the same input but producing less accurate output. Other resilience design patterns, such as rollforward or n-modular redundancy, can be implemented using the proxy approach as well. For rollforward, checkpoints of the active simulation would be used in combination with the faster than real-time simulation capability to catch up. In case of n-modular redundancy, two or more exact replicas of the simulation would run and their output would be compared to detect failures and errors.

B. Resilient Real-time Workflow Demonstration

The basic resilience capabilities (resilient message broker, watchdog, and communication timeout) and the advanced active/standby resilience feature using exact replicas were deployed at ORNL as a demonstration of a federated INTERSECT AAM experiment that is resilient to failures in the feedback loop and has a fail-stop mechanism for other failures. The demonstration uses a virtual 3D printer service that emulates the real 3D printer and replays the data of a

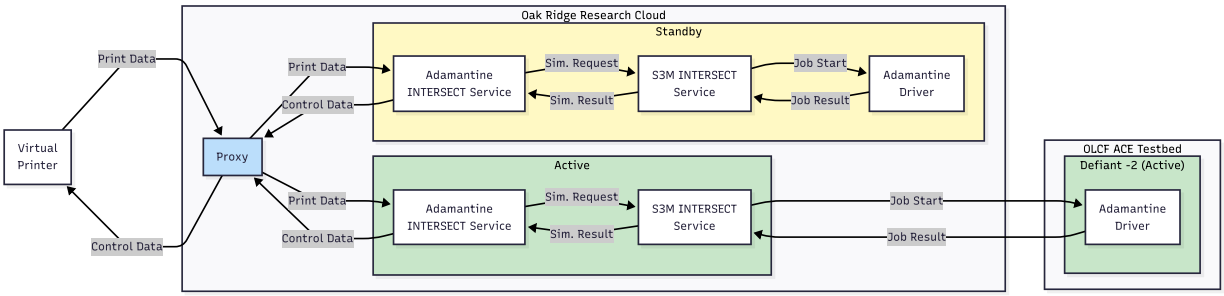


Fig. 10. The demonstration setup for the resilient federated INTERSECT autonomous additive manufacturing experiment at ORNL.

past experiment. Various failures were injected to test the resilience of the deployed solution. The implementation and demonstration details are as follows (Fig. 10).

The clustered RabbitMQ message broker was deployed on the Oak Ridge Research Cloud (ORC). The watchdog was integrated with the proxy service to simplify deployment and demonstrate composability. It was also deployed on the ORC, along with an Adamantine INTERSECT and an Secure Scientific Service Mesh (S3M) INTERSECT service for protocol translations to access the Adamantine Driver (simulation) on the Defiant-2 system in the ACE testbed. The redundant setup with the Adamantine Driver was deployed on the ORC as well.

The clustered RabbitMQ message broker provides reliable communication. For asynchronous detection, the watchdog's heartbeat interval was set to 30 seconds with 3 missed intervals indicating a failure. For synchronous detection, the proxy's timeout for a compute request to the Adamantine simulation was set to 5 seconds. An additional 60 second timeout was set for the Adamantine simulation results.

The Adamantine INTERSECT, S3M INTERSECT and Adamantine Driver services were individually stopped before and after the arrival of print data at the proxy to test the resilience features during different workflow stages [35]. All injected failures were detected, and the fail-over to the standby was performed. The real-time feedback loop was not interrupted. A proxy failure triggered a fail-stop of the printer.

VI. Conclusion

This paper reports on recent work in a resilient INTERSECT federated ecosystem and in demonstrating resilience capabilities for the INTERSECT AAM cross-facility experiment between the ORNL MDF and the OLCF ACE testbed. The resilient INTERSECT ecosystem consists of an architecture with resilience design patterns, a resilient SoS architecture, and a resilient microservices architecture. It also consists of a SDK with reliable service communication and asynchronous and synchronous failure detection and notification. A demonstration shows resilience under various failures using basic resilience capabilities with a resilient message broker, watchdog service, and communication timeout mechanism, and utilizing advanced active/standby resilience using exact replicas of the simulation in the live feedback loop.

Future work will target the wider adoption of the presented concepts and solutions by the instrument-science community, specifically the DoE Genesis Mission, leveraging the INTERSECT federated ecosystem. It will also focus on resilience for agentic workflows and federated learning.

Acknowledgment

Research sponsored by the Laboratory Directed Research and Development Program's INTERSECT Initiative of ORNL. This research used resources of the OLCF and the Compute and Data Environment for Science (CADES) at ORNL, which are supported by the Advanced Scientific Computing Research programs in the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

References

[1] R. F. da Silva et al., "Shaping the future of self-driving autonomous laboratories workshop," Oak Ridge National Laboratory, Tech. Rep. ORNL/TM-2024/3714, Jan. 2, 2025. DOI: 10.5281/zenodo.14430232.

[2] C. Engelmann et al., "The INTERSECT open federated architecture for the laboratory of the future," in *Communications in Computer and Information Science: 18th Smoky Mountains Computational Sciences and Engineering Conference*, vol. 1690, Aug. 24–25, 2022, pp. 173–190. DOI: 10.1007/978-3-031-23606-8_11.

[3] C. Engelmann and S. Somnath, "Science use case design patterns for autonomous experiments," in *28th European Conference on Pattern Languages of Programs*, Jul. 5–9, 2023, pp. 1–14. DOI: 10.1145/3628034.3628060.

[4] C. Engelmann et al., *INTERSECT architecture documentation*. [Online]. Available: https://intersect-architecture.readthedocs.io.

[5] A. M. Thakur et al., "Towards a software development framework for interconnected science ecosystems," in *Communications in Computer and Information Science: 18th Smoky Mountains Computational Sciences and Engineering Conference*, vol. 1690, Aug. 24–25, 2022, pp. 206–224. DOI: 10.1007/978-3-031-23606-8_13.

[6] J. McGaha et al., "Enabling interconnected science workflows through an adapter approach," in *US Research Software Engineers Association Conference*, Oct. 16–25, 2023. DOI: 10.5281/zenodo.10426187.

[7] Oak Ridge National Laboratory, *INTERSECT software development kit*. [Online]. Available: https://github.com/INTERSECT-SDK.

[8] National Energy Research Scientific Computing Center, *Superfacility project*. [Online]. Available: https://www.nersc.gov/research-and-development/superfacility.

[9] National Energy Research Scientific Computing Center, *Superfacility API*. [Online]. Available: https://api.nersc.gov.

[10] K. Troutman, *Superfacility framework advances photosynthesis research*, May 2019. [Online]. Available: https://biosciences.lbl.gov/2019/05/07/superfacility-framework-advances-photosynthesis-research/.

[11] W. L. Miller et al., "Integrated research infrastructure architecture blueprint activity (final report 2023)," US Department of Energy, Office of Science, Tech. Rep. DOI: 10.2172/1984466.

[12] US Department of Energy, *Integrated Research Infrastructure API*. [Online]. Available: https://github.com/doe-iri.

[13] *Globus*. [Online]. Available: https://docs.globus.org.

[14] N. Heinonen, *Argonne researchers use Theta for real-time analysis of COVID-19 proteins*, Jul. 2020. [Online]. Available: https://www.alcf.anl.gov/news/argonne-researchers-use-theta-real-time-analysis-covid-19-proteins.

[15] *Pegasus*. [Online]. Available: https://pegasus.isi.edu.

[16] A. Chatterjee, C. Helbig, R. Malak, and A. Layton, "A comparison of graph-theoretic approaches for resilient system of systems design," *Journal of Computing and Information Science in Engineering*, vol. 23, no. 3, p. 030 906, Apr. 2023. DOI: 10.1115/1.4062231.

[17] D. Mitchell et al., "Symbiotic system of systems design for safe and resilient autonomous robotics in offshore wind farms," *IEEE Access*, vol. 9, pp. 141 421–141 452, 2021. DOI: 10.1109/ACCESS.2021.3117727.

[18] S. Hukerikar and C. Engelmann, "Resilience design patterns: A structured approach to resilience at extreme scale," *Journal of Supercomputing Frontiers and Innovations*, vol. 4, no. 3, pp. 4–42, Oct. 2017. DOI: 10.14529/jsfi170301.

[19] C. Engelmann, R. Ashraf, S. Hukerikar, M. Kumar, and P. Sao, "Resilience design patterns: A structured approach to resilience at extreme scale (version 2.0)," Oak Ridge National Laboratory, Tech. Rep., Dec. 16, 2022. DOI: 10.2172/1922296.

[20] R. Ashraf, S. Hukerikar, and C. Engelmann, "Pattern-based modeling of multiresilience solutions for high-performance computing," in *9th ACM/SPEC International Conference on Performance Engineering*, Apr. 9–13, 2018, pp. 80–87. DOI: 10.1145/3184407.3184421.

[21] M. Kumar and C. Engelmann, "RDPM: An extensible tool for resilience design patterns modeling," in *Lecture Notes in Computer Science: 14th Workshop on Resiliency in High Performance Computing in Clusters, Clouds, and Grids*, vol. 13098, Aug. 30, 2021, pp. 283–297. DOI: 10.1007/978-3-031-06156-1_23.

[22] *Netflix Open Source Software*. [Online]. Available: https://netflix.github.io.

[23] *Spring Cloud Netflix*, Jun. 2026. [Online]. Available: https://spring.io/projects/spring-cloud-netflix.

[24] M. Larsson, *Microservices with Spring Boot and Spring Cloud: Build resilient and scalable microservices using Spring Cloud, Istio, and Kubernetes*. Packt Publishing, Aug. 2025, ISBN: 978-1805801276.

[25] U.S. Department of Defense, *The DoDAF architecture framework version 2.02*, Aug. 2010. [Online]. Available: https://dodcio.defense.gov/Library/DoD-Architecture-Framework.

[26] M. J. Brim, L. Drane, M. McDonnell, C. Engelmann, and A. M. Thakur, "A microservices architecture toolkit for interconnected science ecosystems," in *19th Workshop on Workflows in Support of Large-Scale Science*, Nov. 18, 2024, pp. 2072–2079. DOI: 10.1109/SCW63240.2024.00259.

[27] Microsoft, *Retry pattern - Azure Architecture Center | Microsoft Learn*. [Online]. Available: https://learn.microsoft.com/en-us/azure/architecture/patterns/retry.

[28] *RabbitMQ Clustering Guide*. [Online]. Available: https://www.rabbitmq.com/docs/clustering.

[29] B. Turcksin and S. DeWitt, "Adamantine 1.0: A thermomechanical simulator for additive manufacturing," *Journal of Open Source Software*, vol. 9, no. 102, 2024. DOI: 10.21105/joss.07017.

[30] *Adamantine, a thermomechanical code for additive manufacturing*. [Online]. Available: https://github.com/adamantine-sim/adamantine.

[31] Oak Ridge Leadership Computing Facility, *Advanced Computing Ecosystem Testbed – Defiant-2*. [Online]. Available: https://docs.olcf.ornl.gov/ace_testbed/defiant_quick_start_guide.html.

[32] D. Rosendo et al., "AI agents for enabling autonomous experiments at ORNL's HPC and manufacturing user facilities," in *7th Annual Workshop on Extreme-Scale Experiment-in-the-Loop Computing*, 2025, pp. 2354–2361. DOI: 10.1145/3731599.3767592.

[33] R. Souza et al., "PROV-AGENT: Unified provenance for tracking ai agent interactions in agentic workflows," in *IEEE International Conference on eScience*, 2025, pp. 467–473. DOI: 10.1109/eScience65000.2025.00093.

[34] C. Engelmann, S. L. Scott, C. Leangsuksun, and X. He, "Transparent symmetric active/active replication for service-level high availability," in *7th International Workshop on Global and Peer-to-Peer Computing*, May 14–17, 2007, pp. 755–760. DOI: 10.1109/CCGRID.2007.116.

[35] C. Engelmann, A. Ayres, S. DeWitt, M. Brim, and B. Eiffert, *Resilient federated intersect autonomous additive manufacturing experiment demonstration*, Jul. 2026. DOI: 10.5281/zenodo.21708367.

ARTIFACT DESCRIPTION (AD) APPENDIX

AD I. OVERVIEW OF CONTRIBUTIONS AND ARTIFACTS

A. Paper's Main Contributions

This paper documents recent advances in creating a resilient federated Self-driven Experiments for Science / Interconnected Science Ecosystem (INTERSECT) and in demonstrating its capabilities with the INTERSECT autonomous additive manufacturing (AAM) cross-facility experiment between the Oak Ridge National Laboratory (ORNL) Manufacturing Demonstration Facility (MDF) and the Oak Ridge Leadership Computing Facility (OLCF) Advanced Computing Ecosystem (ACE) testbed. It makes the following novel contributions:

- C_1 A resilient INTERSECT federated ecosystem architecture that provides an implementation blueprint using resilience design patterns, a resilient system of systems (SoS) architecture, and a resilient microservices architecture.
- C_2 A resilient INTERSECT software development kit (SDK) with reliable service communication and asynchronous and synchronous failure detection and notification.
- C_3 A demonstration of the INTERSECT AAM cross-facility experiment that offers fail-stop and fail-over resilience capabilities of a real-time feedback loop using the resilient INTERSECT ecosystem architecture and SDK.

B. Artifacts

- A_1 https://intersect-architecture.readthedocs.io
- A_2 https://github.com/INTERSECT-SDK/python-sdk
- A_3 https://github.com/INTERSECT-SDK/microservice-proxy
- A_4 DOI 10.5281/zenodo.21708367

TABLE I
ARTIFACTS AND RELATED TO CONTRIBUTIONS AND PAPER ELEMENTS

Artifact ID	Contributions Supported	Related Paper Elements
A_1	C_1	Sections IV-A and V, and Figures 7-8
A_2	C_2	Section IV-B and Figures 4-6
A_3	C_3	Section V-B and Figure 10
A_4	C_3	Section V-B and Figure 10

AD II. ARTIFACT IDENTIFICATION

A. Artifact A_1

The resilient INTERSECT federated ecosystem architecture (artifact A_1) contains the contribution C_1, specifically the resilience design patterns, resilient SoS architecture, and resilient microservices architecture. It is a publicly available Web site maintained through GitHub and written in reStructuredText. Its purpose is to communicate the INTERSECT federated ecosystem architecture and provide implementation guidelines to the resilient INTERSECT SDK (contribution C_2 and artifact A_2) and to applications using the INTERSECT federated ecosystem, such as the resilient INTERSECT AAM cross-facility experiment proxy service (contribution C_3 and artifact A_3). The architecture offers a consistent design methodology for a resilient federated instrument-science ecosystem that allows scientists to pick and choose the right solution for the resilience problem at hand and deploy it with ease. A digital object identifier (DOI) is not available for this artifact. It is a Web site containing the contribution.

B. Artifact A_2

The resilient INTERSECT SDK (artifact A_2) is a downloadable package of pre-built tools, libraries, and instructions to create a resilient INTERSECT federated ecosystem for instrument-science workflows. It contains the contribution C_2, specifically: (1) a resilient message broker that ensures reliable communication despite broker service failures, (2) a watchdog service that asynchronously detects service failures using a heartbeat and notifies interested services, and (3) a communication timeout that synchronously detects service failures and notifies a service waiting for a response. The SDK is publicly available through GitHub and written in Python. It follows the design guidelines of the resilient INTERSECT federated ecosystem architecture (contribution C_1 and artifact A_1). It is also used by the resilient INTERSECT AAM cross-facility experiment proxy service (contribution C_3 and artifact A_3). This artifact is part of a versioned code repository. A DOI is not available, as a specific code release with the artifact is not available for DOI generation at the time of writing this paper.

C. Artifact A_3

The proxy service (artifact A_3) is part of the demonstration of the resilient INTERSECT AAM cross-facility experiment (contribution C_3). Specifically, it transparently provides the active/standby fail-over capabilities combined with the watchdog service that asynchronously detects service failures (contribution C_2 and artifact A_2). Since it is part of the resilient INTERSECT SDK, it is publicly available through GitHub and written in Python. It follows the design guidelines of the resilient INTERSECT federated ecosystem architecture (contribution C_1 and artifact A_1). Specifically, it implements the active/standby resilience design pattern of the SoS architecture's logical view. This artifact is part of a versioned code repository. A DOI is not available, as a specific code release with the artifact is not available for DOI generation at the time of writing this paper.

D. Artifact A_4

The demonstration of the INTERSECT AAM cross-facility experiment (contribution C_3) offers fail-stop and fail-over resilience capabilities of a real-time feedback loop using the resilient INTERSECT ecosystem architecture and SDK. These resilience capabilities have been tested by injecting various failures. A video of a part the testing campaign (artifact A_4) demonstrates the following three scenarios:

- The active path accepts the simulation request but returns no results. After the configurable 60-second per-request timeout elapses, the proxy re-sends the same request to the standby path, which takes over seamlessly. Notably, in this specific scenario the active is not marked down — a timeout can be transient, so it stays in rotation.
- The active path stops answering the watchdog's heartbeats entirely. After three consecutive missed configurable 30-second heartbeats the proxy declares it unhealthy and removes it from rotation — this, not a one-off request failure, is what takes it out of service.
- The active path starts answering the watchdog's heartbeats again; on that recovery the proxy automatically fails back to the higher-priority active path, and control updates resume flowing from it.

Throughout, the workflow keeps running — control updates continue reaching the printer across every scenario.