# The E4S 25.06 Data and Visualization Ecosystem: A Comparative Analysis for Scientific Software Users

## Executive Summary

The E4S 25.06 release represents a mature, comprehensive, and cross-platform software stack designed to support the next generation of High-Performance Computing (HPC) and Artificial Intelligence (AI) workloads.1 This ecosystem, a key legacy of the Exascale Computing Project (ECP), curates over 130 HPC and AI packages, providing the largest collection of open-source, GPU-enabled libraries for scientific applications.1 The 25.06 release specifically highlights support for all major GPU architectures, including NVIDIA (CUDA 12.8 and the new Blackwell architecture), AMD (ROCm 6.3), and Intel (oneAPI 2025.1).2

This report analyzes the data and visualization tools within this stack, focusing on user experience and strategic selection. The primary challenge facing scientific software users is the escalating volume of data generated by exascale simulations. This data deluge is rendering traditional *post-hoc* (post-simulation) analysis, which relies on writing massive files to disk, increasingly unviable.5

Consequently, the ecosystem is shifting toward two fundamental pillars: GPU acceleration for all compute-bound tasks and *in-situ* processing, where data is analyzed and visualized in-memory as it is generated.6 This report provides a comparative analysis of the key choices users and developers must make within the E4S 25.06 stack:

1. **I/O Libraries:** The foundational choice between the archival portability of HDF5, the extreme-scale performance and flexibility of ADIOS2, and the domain-specific standard of PnetCDF.
2. **Visualization Platforms:** The ecosystem choice between the two flagship applications, ParaView and VisIt, and their differing design philosophies.
3. **In-Situ Frameworks:** The instrumentation choice between tool-specific libraries (Catalyst, Libsim), a lightweight, GPU-native framework (Ascent), and the strategic abstraction layer that unifies them (SENSEI).
4. **GPU Portability:** The enabling technology, VTK-m, that provides a single, portable solution for GPU-accelerated visualization algorithms across NVIDIA, AMD, and Intel hardware.

This analysis provides scientific users and simulation developers with the context to select the right toolchain for their specific problem class, problem size, and target HPC platform.

## The Foundation: Parallel I/O and Data Management Libraries

The starting point for any analysis or visualization workflow is data input/output (I/O). The ECP has heavily invested in this area, as I/O performance is a primary bottleneck for exascale applications.6 The user's choice of I/O library dictates performance, file portability, and compatibility with the wider ecosystem.

### HDF5 (Hierarchical Data Format 5)

HDF5 is widely described as the "de-facto standard for scientific computing" 9 and the "most popular high-level I/O library" at supercomputing facilities.8

* **User Experience and Problem Class:** The core value of HDF5 is its creation of portable, self-describing files. It provides a hierarchical data model, allowing users to organize complex and heterogeneous data (e.g., meshes, parameters, and results) into a structured system of "groups" and "datasets" within a single file.9 Users select HDF5 when data organization, long-term archival, and broad community compatibility are the primary concerns.13
* **Programming Language:** A key driver for its adoption is its broad language support. The core library is written in C, with official wrappers for C++, Fortran, and Java.12 In the E4S 25.06 AI/ML stack, the h5py Python package provides a critical and easy-to-use interface, making HDF5 a staple in Python-based analysis workflows.11
* **Problem Size and Performance:** HDF5 supports parallel I/O for large-scale simulations by building on top of the MPI-IO standard.9 However, user experiences and benchmarks show that HDF5 performance can be a significant bottleneck without careful "tuning" of parameters like chunking and alignment.15 At extreme scale, its N-to-1 (all processes write to one file) approach can suffer from file system locking contention.18 The ECP's "ExaHDF5" project is actively developing new features to address these performance challenges and adapt HDF5 to modern, tiered storage systems like burst buffers.8
* **GPU Support:** HDF5 is not inherently GPU-accelerated. However, recent ECP-funded work has developed a Virtual File Driver (VFD) that leverages NVIDIA's GPUDirect Storage (GDS). This allows for high-speed data movement directly from GPU-resident memory to the file system, with experiments showing speedups of over 2x compared to traditional CPU-based memory copies.20

### ADIOS2 (The Adaptable Input/Output System 2)

ADIOS2 is an ECP-funded library designed from the ground up to provide extreme-scale I/O performance and *adaptability*.21

* **User Experience and Problem Class:** The central philosophy of ADIOS2 is the "separation of concerns".25 A developer writing simulation code makes simple, declarative API calls (e.g., "put this variable").26 The actual I/O *method* (the "engine") is specified at runtime in a configuration file, requiring no code recompilation.27
* **The "Engine" Concept:** This is the most powerful feature for users.
  + **BP4 / BP5 Engine:** This is the default, proprietary "Binary Pack" format. It consistently wins I/O benchmarks against HDF5 and PnetCDF.27 It achieves this by using an N-to-N or N-to-M approach where each process writes its own "sub-file," avoiding file-locking contention.18
  + **HDF5 Engine:** ADIOS2 can be configured to use HDF5 as its backend, writing a single, fully compliant parallel HDF5 file.25
  + **Staging Engines (SST, SSC, DataMan):** These engines bypass disk entirely. They are designed to *stream* data in-memory or over the network to another application, forming the backbone of *in-situ* and *in-transit* workflows.22
* **Programming Language:** The primary API is a modern C++11 interface. Robust wrappers are provided for C, Fortran, Python, and MATLAB.22
* **GPU Support:** ADIOS2 is explicitly "GPU-aware".33 It can be built with support for CUDA, HIP, and SYCL. Its API can automatically detect if a data buffer passed by the user resides on the host (CPU) or the device (GPU) and orchestrate data movement, minimizing costly memory copies.34 This is a critical feature for modern GPU-centric simulations.

### PnetCDF (Parallel-netCDF)

PnetCDF fills a specific, vital role within the HPC ecosystem, particularly for scientific domains that have standardized on the classic NetCDF file format.35

* **User Experience and Problem Class:** PnetCDF provides a high-performance, parallel I/O interface for the *classic* NetCDF file formats (CDF-1, CDF-2, and the large-file CDF-5).35 The NetCDF data model, which is simpler than HDF5's, is the dominant standard in fields like climatology, meteorology (e.g., the WRF application), and oceanography.24 For users in these communities, PnetCDF is the preferred, high-performance solution.38
* **The HDF5 vs. PnetCDF Distinction:** This is a common point of confusion. The *NetCDF-4* library *uses HDF5* as its underlying storage layer.40 *PnetCDF* is a separate, independent library that provides parallel access to the *classic* (pre-NetCDF-4) formats.40
* **Programming Language and Performance:** The PnetCDF API was designed to be a "minimal" change from the serial NetCDF library, easing migration for legacy codes.36 It provides C, Fortran (77/90), and C++ interfaces.43 It uses MPI-IO underneath to provide collective and non-blocking I/O operations, offering "dramatic performance gains" over serial I/O.36

### API vs. Format: A New I/O Strategy

The sophisticated "engine" architecture of ADIOS2 is reframing the I/O decision for application developers. The choice is no longer a permanent, mutually exclusive commitment to a single file format.

A simulation's "producer" (the simulation) and its "consumer" (the analysis tool) have different requirements. The producer needs maximum write performance to avoid stalling the simulation.23 The consumer needs a portable, self-describing file (like HDF5) compatible with a wide array of tools.14

Historically, these needs were in conflict. Writing a parallel HDF5 file is often slower than writing to a simpler format.27 ADIOS2's HDF5 engine resolves this conflict.25 A simulation developer can code *once* against the modern ADIOS2 API (in C++, Fortran, or Python).31 At runtime, a user can provide an XML file 22 to select the HDF5 engine, producing a fully compliant HDF5 file. For a performance run, they can switch to the BP5 engine. For an *in-situ* debugging run, they can switch to the SST streaming engine. All of this is possible *without recompiling the simulation code*.

This indicates a strategic shift: the decision is moving from the *file format* to the *API*. ADIOS2 is positioning itself as the superior, adaptable *API* for HPC I/O, treating the final output format (BP5, HDF5, or a stream) as a runtime "deployment" detail.

**Table 1: I/O Library Feature and Performance Matrix**

| **Feature** | **HDF5** | **ADIOS2** | **PnetCDF** |
| --- | --- | --- | --- |
| **Data Model** | Hierarchical groups & datasets | Flexible key/value variables | Multi-dim arrays (classic format) |
| **Primary API** | C, C++, Fortran, Java, Python (h5py) 12 | C++11, Fortran, C, Python, MATLAB 22 | C, Fortran 77/90, C++ 43 |
| **Core Philosophy** | Long-term archival, data organization, portability 12 | Performance, adaptability, *in-situ* streaming 23 | Parallel I/O for classic NetCDF standard 35 |
| **Performance Profile** | Highly tunable; can be I/O bound at scale 17 | Highest performance via BP5 engine; scales well 18 | High-performance for its data model 36 |
| **GPU-Aware** | Emerging (GDS VFD) 20 | **Yes** (CUDA, HIP, SYCL buffer-aware) 33 | No |
| **Best-Fit Use Case** | Archival data, complex heterogeneous data, community standard | Exascale simulations, *in-situ* workflows, performance-critical I/O | Climate, weather, and ocean modeling (e.g., WRF) |

## The Core Platforms: Post-Hoc Analysis and Visualization

The E4S 25.06 stack includes two premier, open-source visualization applications: ParaView and VisIt.2 Both are mature, feature-rich platforms that serve as the primary interfaces for scientists to conduct *post-hoc* analysis (i.e., on data files) and can also be used as backends for *in-situ* analysis.

### ParaView

* **User Experience and Ecosystem:** ParaView is an open-source, multi-platform tool developed as a collaboration between Kitware, Los Alamos National Laboratory, and Sandia National Laboratories.48 It is heavily used and supported at Sandia.51 The application is built on top of the Visualization Toolkit (VTK).48 It has a large and active user community, with official support forums 54 and extensive tutorial materials.56
* **Target Problem Class:** ParaView is explicitly designed for the analysis of *extremely large datasets*.53 It is frequently cited in the context of Computational Fluid Dynamics (CFD), particularly for post-processing OpenFOAM results.59 Users interact with a "pipeline" browser 61, applying a series of filters (e.g., Slicing, Contouring, Streamlines) to large mesh data.57 User discussions confirm its use for complex meshes of 40-60 million points and billion-cell simulations.57
* **Programming Language:** The core of ParaView is C++.53 However, its primary interface for automation and advanced analysis is a rich Python scripting API.48 Users can script the entire visualization pipeline, from loading data to applying filters and saving images, using the paraview.simple module.66 This API is powerful enough that recent research is exploring the use of Large Language Models (LLMs) to generate this Python code automatically.67
* **GPU Availability:** ParaView leverages GPUs for two distinct tasks. First, it uses GPUs for *rendering* via standard OpenGL or advanced ray-tracing backends like NVIDIA OptiX (for NVIDIA GPUs) and Intel OSPRay (a CPU-based ray tracer).64 Second, it can use GPUs for *computation* (i.e., running filters) through the VTK-m backend, which is discussed in Section 5.63

### VisIt

* **User Experience and Ecosystem:** VisIt is also an open-source, interactive, and scalable visualization tool, originating from Lawrence Livermore National Laboratory (LLNL).72 It is designed to handle data "ranging in scale from small... to... leadership-class computing facility simulation campaigns".73 A hallmark feature is its ability to read over 130 different scientific file formats, making it a "Swiss Army knife" for data analysis.74
* **Target Problem Class:** VisIt is built for visualizing large-scale scientific data.74 It supports a wide range of plots, including Mesh, Pseudocolor, Contour, and Volume rendering.76 User discussions show its application to challenging problems like high-resolution volume rendering and time-series analysis.79
* **Programming Language:** Like ParaView, VisIt's core is C++.76 It also offers a comprehensive Python interface, which it refers to as the Command Line Interface (CLI).77 This Python interface can be used interactively with the GUI or in a "batch" mode (e.g., visit -nowin -cli -s script.py) for automated processing.81 VisIt also provides a C, C++, and Fortran API for *in-situ* integration, known as "Libsim".76
* **GPU Availability:** VisIt uses the GPU for rendering. Its capability for GPU-accelerated *computation* is also provided by the VTK-m backend, which users can activate in the application's preferences.87

### Comparative Analysis: ParaView vs. VisIt

Scientific computing users typically standardize on one of these two platforms.72 While both are built on VTK 90 and have similar goals, their user experiences and design philosophies differ in ways that guide user selection.

A key philosophical difference is revealed in user forum discussions.91 A user noted that an HDF5 file volume-rendered automatically in VisIt but appeared as a blank screen in ParaView. An expert explained that VisIt "does some data transformations implicitly" to produce a "correct" image, while ParaView "errs on side of letting user transform the data" explicitly. In that case, ParaView required the user to manually add a CellDataToPointData filter to see the result. This reveals a fundamental trade-off:

* **VisIt** is often preferred for a "quick look" or by users who trust the tool to *implicitly* make smart decisions to get them to a useful visualization quickly.
* **ParaView** is often preferred by users who want *explicit*, granular control over every step of the visualization pipeline, which is crucial for debugging data or creating publication-quality images.

Institutional inertia and R&D investment also play a significant role. Sandia and Kitware are primary drivers of ParaView 51, while LLNL is the home of VisIt.73 This R&D drives the future of the tools. For example, recent publications show active research at LLNL and Kitware to build autonomous LLM agents (e.g., "ParaView-MCP") on top of ParaView's Python API.67 This suggests a strong push to lower ParaView's "steep learning curve" 68 via natural language interfaces, which may become a key differentiator in the future.

**Table 2: Post-Hoc Visualization Platform Comparison**

| **Feature** | **ParaView** | **VisIt** |
| --- | --- | --- |
| **Primary Origins** | Sandia Nat'l Labs, Kitware, Los Alamos Nat'l Lab 48 | Lawrence Livermore Nat'l Lab 72 |
| **Scripting Interface** | Python (paraview.simple) 65 | Python (CLI/VisIt module) 77 |
| **Data Model Philosophy** | **Explicit:** User must build the full pipeline (e.g., manually add CellToPoint filter) 91 | **Implicit:** Attempts to "do the right thing" for the user automatically 91 |
| **Key Strengths** | Rich filter pipeline, strong CFD community, deep Python integration, Catalyst *in-situ* 60 | Reads >130 file formats, robust remote client-server, Libsim *in-situ* 51 |
| **Common Problem Class** | CFD/OpenFOAM, large mesh analysis, climate 57 | Astrophysics, climate, diverse file format analysis, large mesh analysis 74 |
| **GPU Computation** | Via VTKmFilters Plugin 71 | Via "Use VTK-m" Preferences toggle 87 |

## The Exascale Imperative: In-Situ Visualization Frameworks

The fundamental challenge of exascale computing is that simulation data is generated far faster than it can be written to disk.5 The "I/O bottleneck" 5 has forced a paradigm shift from *post-hoc* analysis to *in-situ* analysis, where visualization and data reduction are performed in memory, "whilst the simulation is running".5 The E4S 25.06 stack provides three key libraries for this.

### ParaView Catalyst

Catalyst is ParaView's *in-situ* analysis and visualization library.94 It allows a simulation code to be instrumented to execute ParaView visualization pipelines "in step with the simulation".7 Users can generate "canned" image outputs, save reduced datasets (e.g., slices or isosurfaces), or connect a live ParaView GUI for real-time visualization and computational steering.7 The analysis pipelines themselves are often defined using Python scripts that can be generated directly from the ParaView GUI, lowering the barrier for complex analysis.95

### VisIt Libsim

Libsim is the *in-situ* library for VisIt.94 Its design philosophy is to make the instrumented simulation "act in many ways like a VisIt compute engine".85 This is a powerful model because it exposes the *full feature list of VisIt* to the user for *in-situ* data.97 Developers instrument their C, C++, or Fortran simulations by linking libsim and providing a set of data access "callback functions" (e.g., GetMetaData, GetMesh) that Libsim uses to "pull" data from the simulation's memory as needed.76

### Ascent

Ascent is a newer, ECP-funded *in-situ* library designed to be a "flyweight" solution.99 It is built from the ground up for minimal dependencies (e.g., no OpenGL is required), a low memory footprint, and native execution on many-core (GPU) architectures.99 It has been demonstrated scaling across over 16,000 GPUs.99 For simulation developers, Ascent is designed for "ease of use".99 It provides C++, C, Python, and Fortran APIs 100, but the visualization *actions* are defined in simple JSON or YAML files rather than complex Python scripts 99, representing a much lower barrier to entry.

### A Spectrum of In-Situ Trade-offs

These three frameworks are not just competitors; they represent a spectrum of trade-offs for the simulation developer.

1. **VisIt Libsim** is at the "heavyweight, full-featured" end. It embeds the *entire* VisIt engine, granting immense power 97 at the cost of a higher resource footprint and a more complex C/C++ callback instrumentation.85 This is for users who *need* a specific, complex VisIt plot *in-situ*.
2. **ParaView Catalyst** is in the "pipeline-driven" middle. It executes a pre-defined ParaView *pipeline*.94 This is more constrained than "all of VisIt" but is highly flexible, especially since those pipelines can be prototyped in the ParaView GUI and exported to Python.95
3. **Ascent** is at the "flyweight, GPU-native" end. It is not a full-featured visualization tool. It is designed to do a few key tasks (make pictures, transform data, save extracts) 99 *extremely fast* on GPUs with minimal overhead.99

The choice depends on the developer's needs: Libsim for full-featured analysis, Catalyst for complex data reduction pipelines, and Ascent for high-speed, low-overhead, GPU-native rendering.

**Table 3: In-Situ Framework Comparison**

| **Feature** | **ParaView Catalyst** | **VisIt Libsim** | **Ascent** |
| --- | --- | --- | --- |
| **Core Philosophy** | Execute ParaView pipelines *in-situ* 94 | Embed the *full* VisIt engine in the simulation 85 | Flyweight, GPU-native, minimal-dependency execution 99 |
| **Resource Footprint** | Medium-to-High | High | Low ("flyweight") 99 |
| **Primary Interface** | C++/Python sim API, Python pipeline script 95 | C/C++/Fortran data callbacks 76 | C++/C/Python/Fortran sim API, YAML/JSON action file 99 |
| **Ecosystem** | ParaView | VisIt | Standalone, VTK-m/Devil Ray backends 99 |
| **Best-Fit Use Case** | Complex data reduction pipelines, *in-situ* steering with ParaView | Full-featured analysis, users needing specific VisIt plots | Fast, low-overhead, GPU-native rendering and data transforms |

## The Unification Layer: SENSEI and "Write Once, Run Anywhere"

The *in-situ* landscape, with its competing, powerful frameworks, presents a new problem: fragmentation. A simulation developer who invests the significant effort to instrument their code for Catalyst is then locked out of Libsim, and vice-versa.105

SENSEI, an ECP-funded project, was created to solve this problem.107 Its design motto is "Write once, run anywhere".109

* **For the Simulation Developer:** SENSEI provides a "generic in situ interface".107 The developer instruments their code *only once* using the SENSEI API, mapping their simulation data to SENSEI's abstract data model.96 This single instrumentation effort "future-proofs" the code.108
* **For the End-User:** A SENSEI-instrumented simulation can use *any* supported backend. The end-user selects their desired tool at *runtime* via a simple XML configuration file.96

SENSEI provides "analysis adaptors" that "bridge" the SENSEI data model to the specific APIs of all major *in-situ* backends, including:

* ParaView Catalyst 112
* VisIt Libsim 112
* Ascent 114
* ADIOS (for *in-transit* staging) 112
* VTK-m (for direct, lightweight computation) 112
* Python scripts (for custom analysis) 112

SENSEI is a strategic abstraction layer. It was developed by a collaboration of researchers from Kitware, ORNL, LBNL, and others 108 to unify the DOE's *in-situ* ecosystem. For any large, multi-institution simulation project (like those using AMReX 112), instrumenting with SENSEI is the clear strategic choice. It de-risks development and provides the entire user community with the flexibility to use their preferred analysis tool, all from a single, common instrumentation.

## The Portability Engine: VTK-m and Cross-Platform GPU Acceleration

A central challenge for the E4S stack is the diverse and incompatible GPU hardware landscape of modern HPC. Leadership systems are built with NVIDIA (CUDA), AMD (HIP), or Intel (SYCL) accelerators.117 The E4S 25.06 release ships support for all three.2 The "secret weapon" that makes cross-platform *visualization* on these GPUs possible is **VTK-m**.

* **The Problem:** Visualization algorithms (e.g., isosurfacing, slicing) are computationally expensive. On a GPU-based machine, this computation *must* run on the GPU to be performant.71 Rewriting every visualization filter in ParaView and VisIt for CUDA, HIP, *and* SYCL would be unmaintainable.
* **The Solution: VTK-m** (Visualization Toolkit for Massively Threaded Architectures) is an ECP-funded acceleration library, not a standalone tool.120 It is designed to be integrated into applications like ParaView and VisIt.88
* **The "Write Once, Run Anywhere" Model:** VTK-m provides a C++ template-based framework that allows developers to write a visualization algorithm *once*.71 VTK-m's "device adapter" layer then compiles this single algorithm to run on numerous backends:
  + **NVIDIA GPUs** using the **CUDA** backend 71
  + **AMD GPUs** using the **HIP** backend 71
  + **Intel GPUs** using the **SYCL / oneAPI** backend 71
  + Multicore CPUs using **OpenMP** and **TBB** 123
* **The User Experience:** The integration of VTK-m is designed to be seamless.
  + **In ParaView:** VTK-m accelerated filters are provided via a plugin named VTKmFilters.71 The user must load this plugin (Tools -> Manage Plugins...) to access the GPU-accelerated versions of filters.
  + **In VisIt:** The experience is even simpler. The user navigates to "Preferences" and "Turns on VTK-m".87 After this, when a plot with a VTK-m implementation (like Contour or Pseudocolor) is used, VisIt will automatically engage the VTK-m (and thus GPU) backend.

VTK-m is the abstraction layer that decouples the visualization algorithm from the specific GPU hardware. A developer can write a new filter in VTK-m, and it *automatically* becomes a GPU-accelerated filter in *both* ParaView and VisIt on *all three* major GPU platforms. This is a massive force multiplier for the entire HPC visualization ecosystem and the core of the E4S 25.06 cross-platform visualization strategy.

**Table 4: GPU Portability Support Matrix (via VTK-m)**

| **Tool** | **NVIDIA (CUDA)** | **AMD (HIP)** | **Intel (SYCL)** | **Enabling Mechanism** |
| --- | --- | --- | --- | --- |
| **ParaView (Filters)** | **Yes** 71 | **Yes** 71 | **Yes** 71 | Load VTKmFilters plugin 71 |
| **VisIt (Plots)** | **Yes** 71 | **Yes** 71 | **Yes** 71 | Enable "Use VTK-m" in Preferences 87 |
| **Ascent (Actions)** | **Yes** (via VTK-m/Devil Ray) 104 | **Yes** (via Devil Ray) 104 | (Not listed) | N/A (Core design) 99 |

## Synthesis: Scenario-Based Recommendations

This final section synthesizes the analysis into actionable recommendations for common scientific user scenarios.

### 6.1 Scenario: The CFD Lab (e.g., OpenFOAM User)

* **Recommendation:** Use **ParaView** for *post-hoc* visualization.
* **Rationale:** The link between ParaView and the OpenFOAM community is exceptionally strong, with a large ecosystem of tutorials and native readers.59 ParaView's explicit pipeline philosophy 91 is well-suited for the complex filter chains (e.g., streamlines, vorticity) common in CFD. For *in-situ*, **Catalyst** is the natural choice.
* I/O: If I/O performance is a bottleneck, instrumenting the solver to use ADIOS2 with the HDF5 engine 25 provides a performant path that still produces standard HDF5 files for  
  collaboration.
* **GPU:** Use ParaView's **NVIDIA OptiX** backend for high-fidelity ray-traced images 64 and load the **VTKmFilters plugin** 71 to accelerate filter computations on the GPU.

### 6.2 Scenario: The Climate/Weather Researcher (e.g., WRF User)

* **Recommendation:** Use **PnetCDF** for I/O and **VisIt** for visualization.
* **Rationale:** This toolchain is the established standard for the climate, weather, and ocean modeling communities.24 **PnetCDF** is the ECP-supported library for high-performance parallel I/O on the *classic* NetCDF file formats.35 **VisIt** is an excellent choice due to its "implicit" data transformation philosophy 91 and its historical strength in reading and interpreting the data formats from this domain.74
* **GPU:** Enable **VTK-m in VisIt's preferences** 87 to leverage GPU acceleration for common plots like contours and pseudocolor.

### 6.3 Scenario: The Exascale Simulation Developer (e.g., New ECP Code)

* **Recommendation:** Instrument the simulation with **SENSEI** and use **ADIOS2** for I/O.
* **Rationale:** This is the most strategic, flexible, and future-proof combination.
  1. **SENSEI** 109 de-risks the *in-situ* instrumentation effort. The code is instrumented *once* and is instantly compatible with ParaView/Catalyst, VisIt/Libsim, and Ascent.112 This gives the simulation's *users* their choice of tool and frees the development team from "ecosystem lock-in".105
  2. **ADIOS2** 23 provides the highest-performance I/O for simulation checkpoints using its BP5 engine.27 More importantly, its staging engines (SST) 22 are the ideal *transport layer* for SENSEI, enabling *in-transit* analysis on dedicated nodes.

### 6.4 Scenario: The User on Frontier (OLCF - AMD GPUs)

* **Recommendation:** Use the E4S 25.06 module for **ParaView** or **VisIt**.
* Rationale: The E4S 25.06 stack includes AMD's ROCm 6.3.2 The visualization tools are compiled with the VTK-m HIP backend 71, providing native GPU acceleration for visualization algorithms on Frontier's AMD GPUs. This is not an emulated or  
  slow-path; it is the intended, high-performance solution. For in-situ work, Ascent is an excellent, GPU-native choice.99

### 6.5 Scenario: The User on Aurora (ALCF - Intel GPUs)

* **Recommendation:** Use the E4S 25.06 module for **ParaView** or **VisIt**.
* Rationale: The E4S 25.06 stack includes Intel's oneAPI 2025.1 toolkit.2 This  
  enables the VTK-m SYCL backend 71, allowing the same portable visualization filters to run on Aurora's Intel GPUs. This cross-platform portability is a key achievement of the ECP and a core strength of the E4S stack.

#### Works cited

1. NEWS RELEASE - E4S, accessed November 6, 2025, <https://e4s.io/news/NEWS_RELEASE_E4S_25.06.pdf>
2. E4S: The Extreme-scale Scientific Software Stack Release 25.06, accessed November 6, 2025, <https://oaciss.uoregon.edu/e4s/talks/E4S_25.06.pdf>
3. E4S - Home, accessed November 6, 2025, <https://e4s.io/>
4. Releases · E4S-Project/e4s - GitHub, accessed November 6, 2025, <https://github.com/E4S-Project/e4s/releases>
5. In-situ Visualization: State-of-the-art and Some Use Cases, accessed November 6, 2025, <https://prace-ri.eu/wp-content/uploads/In-situ_Visualization_State-of-the-art_and_Some_Use_Cases-2.pdf>
6. Data Management and Visualization Approaches for the U.S. Exascale Computing Project, accessed November 6, 2025, <https://oaciss.uoregon.edu/E4S-Forum19/talks/Ahrens-E4S.pdf>
7. ParaView - Advanced - Advanced Topics - Catalyst: In-Situ Visualization, accessed November 6, 2025, <https://cvw.cac.cornell.edu/paraview-adv/advanced-topics/catalyst>
8. ExaIO - Exascale Computing Project, accessed November 6, 2025, <https://www.exascaleproject.org/research-project/exaio/>
9. Efficient Parallel I/O with HDF5 and Proactive Data Containers (PDC), accessed November 6, 2025, <https://www.olcf.ornl.gov/wp-content/uploads/2019/08/20190625_Suren_OLCF-talk-s.pdf>
10. ECP Software Technology Summaries - Exascale Computing Project, accessed November 6, 2025, <https://www.exascaleproject.org/wp-content/uploads/2024/08/ECP-ST-Summaries_2024-08-05.pdf>
11. Opinions on NetCDF vs HDF5 for storing scientific data? - Stack Overflow, accessed November 6, 2025, <https://stackoverflow.com/questions/1075074/opinions-on-netcdf-vs-hdf5-for-storing-scientific-data>
12. Parallel I/O - ADMIN Magazine, accessed November 6, 2025, <https://www.admin-magazine.com/HPC/Articles/Improved-Performance-with-Parallel-I-O/(offset)/4>
13. HDF5 Data Management: Tools for Faster, Better Science | Stanford Research Computing, accessed November 6, 2025, <https://srcc.stanford.edu/events/hpc-tools/hdf5-data-management-tools-faster-better-science>
14. Academic Research - The HDF Group - ensuring long-term access and usability of HDF data and supporting users of HDF technologies, accessed November 6, 2025, <https://www.hdfgroup.org/solutions/hdf5/academic-research/>
15. Achieving High Performance I/O with HDF5 - The HDF Group, accessed November 6, 2025, <https://www.hdfgroup.org/wp-content/uploads/2020/02/20200206_ECPTutorial-final.pdf>
16. HDF5 Research Data Management Toolbox — Documentation, accessed November 6, 2025, <https://h5rdmtoolbox.readthedocs.io/>
17. Battle of the Defaults: Extracting Performance Characteristics of HDF5 under Production Load - SDM, accessed November 6, 2025, <https://sdm.lbl.gov/~sbyna/research/papers/2021/202105-CCGrid2021_hdf5.pdf>
18. High Performance Parallel I/O and In-Situ Analysis in the WRF Model with ADIOS2 - arXiv, accessed November 6, 2025, <https://arxiv.org/pdf/2201.08228>
19. DataLib Innovates with Some of the Most Successful Storage and I/O Software in the DOE Complex - Exascale Computing Project, accessed November 6, 2025, <https://www.exascaleproject.org/highlight/datalib-innovates-with-some-of-the-most-successful-storage-and-i-o-software-in-the-doe-complex/>
20. GPU Direct I/O with HDF5, accessed November 6, 2025, <https://pdsw.org/pdsw20/papers/ws_pdsw_S2_P3_Ravi.pdf>
21. Research Computing Systems - Lehigh Confluence - Atlassian, accessed November 6, 2025, <https://lehigh.atlassian.net/wiki/spaces/hpc/pages/45252895/ADIOS2>
22. ADIOS 2: The Adaptable Input/Output System version 2 — ADIOS2 2.10.2 documentation, accessed November 6, 2025, <https://adios2.readthedocs.io/>
23. Adaptable IO System delivers data for high-performance computing science at ORNL, accessed November 6, 2025, <https://www.ornl.gov/news/adaptable-io-system-delivers-data-high-performance-computing-science-ornl>
24. Parallel I/O - NHR@KIT User Documentation, accessed November 6, 2025, <https://www.nhr.kit.edu/userdocs/horeka/parallel_IO/>
25. Parallel I/O Libraries ADIOS - Basic Design of ADIOS - Cornell Virtual Workshop, accessed November 6, 2025, <https://cvw.cac.cornell.edu/parallel-io-libraries/adios/design>
26. ADIOS: Storage and in situ I/O - Exascale Computing Project, accessed November 6, 2025, <https://www.exascaleproject.org/wp-content/uploads/2021/02/ADIOS_tutorial_ECP_AHM_Apr2021_full.pdf>
27. ADIOS2: Performance benefits over MPIIO and HDF5? - Supercomputing Asia, accessed November 6, 2025, <https://www.sc-asia.org/2023/wp-content/uploads/2023/02/post107.pdf>
28. Performance Analysis of Filesystem I/O using HDF5 and ADIOS on a Cray XC30, accessed November 6, 2025, <https://cug.org/proceedings/cug2014_proceedings/includes/files/pap121.pdf>
29. ADIOS 2.9: A Framework to Enable HPC Tools for Extreme Scale I/O, In Situ Visualization, and Performance Analysis - My OLCF, accessed November 6, 2025, <https://users.nccs.gov/~pnorbert/ADIOS_tutorial_SC23.pdf>
30. Streaming Data in HPC Workflows Using ADIOS - arXiv, accessed November 6, 2025, <https://arxiv.org/html/2410.00178v1>
31. FAQ — ADIOS2 2.10.0-rc1 documentation, accessed November 6, 2025, <https://adios2.readthedocs.io/en/latest/faq/faq.html>
32. Install from Source — ADIOS2 2.10.0-rc1 documentation, accessed November 6, 2025, <https://adios2.readthedocs.io/en/latest/setting_up/setting_up.html>
33. ADIOS 2: A Framework for Extreme Scale I/O and In Situ Processing, accessed November 6, 2025, <https://www.olcf.ornl.gov/wp-content/uploads/ADIOS_OLCF_July2024.pdf>
34. GPU-aware I/O — ADIOS2 2.10.0-rc1 documentation, accessed November 6, 2025, <https://adios2.readthedocs.io/en/latest/advanced/gpu_aware.html>
35. PnetCDF - Virtual Institute for I/O, accessed November 6, 2025, <https://www.vi4io.org/tools/middleware/pnetcdf>
36. Parallel netCDF: A High-Performance Scientific I/O Interface - Northwestern University, accessed November 6, 2025, <https://users.ece.northwestern.edu/~choudhar/Publications/LiLia03A.pdf>
37. PnetCDF (Parallel netCDF) - cucis - Northwestern University, accessed November 6, 2025, <http://cucis.ece.northwestern.edu/projects/PnetCDF/>
38. ACCELERATING WRF I/O PERFORMANCE WITH ADIOS2 AND NETWORK-BASED STREAMING - arXiv, accessed November 6, 2025, <https://arxiv.org/pdf/2304.06603>
39. Higher-level I/O libraries - Argonne National Laboratory, accessed November 6, 2025, <https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2023/08/ATPESC-2023-Track-7-Talk-8-latham-io-libs.pdf>
40. Parallel I/O with NetCDF-4, accessed November 6, 2025, <https://docs.unidata.ucar.edu/netcdf-c/4.9.3/parallel_io.html>
41. PnetCDF, accessed November 6, 2025, <https://parallel-netcdf.github.io/>
42. Interoperability with HDF5 - NetCDF, accessed November 6, 2025, <https://docs.unidata.ucar.edu/netcdf-c/4.9.3/interoperability_hdf5.html>
43. PnetCDF Quick Tutorial, accessed November 6, 2025, <https://parallel-netcdf.github.io/wiki/QuickTutorial.html>
44. PnetCDF User Guide, accessed November 6, 2025, <https://parallel-netcdf.github.io/wiki/Documentation.html>
45. Linking PnetCDF, NetCDF-C and NetCDF-Fortran libraries for an earth system model - Stack Overflow, accessed November 6, 2025, <https://stackoverflow.com/questions/77457782/linking-pnetcdf-netcdf-c-and-netcdf-fortran-libraries-for-an-earth-system-model>
46. [cs/0306048] Parallel netCDF: A Scientific High-Performance I/O Interface - arXiv, accessed November 6, 2025, <https://arxiv.org/abs/cs/0306048>
47. Parallel-NetCDF - ATPESC, accessed November 6, 2025, <https://extremecomputingtraining.anl.gov/wp-content/uploads/sites/96/2022/11/ATPESC-2022-Track-3-Talk-8-Latham-io-libs.pdf>
48. ParaView in a nutshell - Wolf Dynamics, accessed November 6, 2025, <https://www.wolfdynamics.com/wiki/paraview.pdf>
49. ParaView - Wikipedia, accessed November 6, 2025, <https://en.wikipedia.org/wiki/ParaView>
50. ParaView - Open-source, multi-platform data analysis and visualization application, accessed November 6, 2025, <https://www.paraview.org/>
51. [Paraview] [EXTERNAL] Re: Comparison of Visit and ParaView development, accessed November 6, 2025, <https://www.paraview.org/pipermail/paraview/2016-June/037222.html>
52. ParaView – Center for Computing Research (CCR) - Sandia National Laboratories, accessed November 6, 2025, <https://www.sandia.gov/ccr/software/paraview/>
53. 1. Introduction to ParaView, accessed November 6, 2025, <https://docs.paraview.org/en/latest/UsersGuide/introduction.html>
54. ParaView, accessed November 6, 2025, <https://discourse.paraview.org/>
55. Welcome to the ParaView discussion forum, accessed November 6, 2025, <https://discourse.paraview.org/t/welcome-to-the-paraview-discussion-forum/32>
56. Community Recorded Workshop: Introduction to Scientific Visualization with ParaView, accessed November 6, 2025, <https://discourse.paraview.org/t/community-recorded-workshop-introduction-to-scientific-visualization-with-paraview/6663>
57. 4. Visualizing Large Models - ParaView Documentation, accessed November 6, 2025, <https://docs.paraview.org/en/latest/Tutorials/SelfDirectedTutorial/visualizingLargeModels.html>
58. Compare ParaView vs. Visdom in 2025 - Slashdot, accessed November 6, 2025, <https://slashdot.org/software/comparison/ParaView-vs-Visdom/>
59. [General] Paraview Visualization - Forums - CFD Online, accessed November 6, 2025, <https://www.cfd-online.com/Forums/paraview/216505-paraview-visualization.html>
60. Paraview or CFD-Post: Which one is better to use? Especially for the Vortex - Reddit, accessed November 6, 2025, <https://www.reddit.com/r/CFD/comments/1l6ujmz/paraview_or_cfdpost_which_one_is_better_to_use/>
61. 2. Basic Usage - ParaView Documentation, accessed November 6, 2025, <https://docs.paraview.org/en/latest/Tutorials/SelfDirectedTutorial/basicUsage.html>
62. Large Scale Visualization with ParaView 3, accessed November 6, 2025, <https://perswww.kuleuven.be/~u0016541/MHD_sheets_pdf/SC07_tut107_ParaView_Handouts.pdf>
63. Improving ParaView Performance for the Visualization of Large .vtu Models, accessed November 6, 2025, <https://discourse.paraview.org/t/improving-paraview-performance-for-the-visualization-of-large-vtu-models/13485>
64. Hardware Recommendations - ParaView Support, accessed November 6, 2025, <https://discourse.paraview.org/t/hardware-recommendations/12209>
65. ParaView/Python 5.13.0 documentation, accessed November 6, 2025, <https://www.paraview.org/paraview-docs/v5.13.0/python/>
66. simple Package — ParaView/Python 6.0.0 documentation, accessed November 6, 2025, <https://www.paraview.org/paraview-docs/latest/python/paraview.simple.html>
67. ChatVis: Assisting and Evaluating Large Language Models for Generating Scientific Visualizations - arXiv, accessed November 6, 2025, <https://arxiv.org/html/2507.23096>
68. ParaView-MCP: An Autonomous Visualization Agent with Direct Tool Use - arXiv, accessed November 6, 2025, <https://arxiv.org/html/2505.07064v1>
69. Paraview compatibility with different graphic cards, accessed November 6, 2025, <https://discourse.paraview.org/t/paraview-compatibility-with-different-graphic-cards/8272>
70. Leverage the power of your GPU in ParaView Webinar - YouTube, accessed November 6, 2025, <https://www.youtube.com/watch?v=IXXDxM62tUU>
71. VTK-m Accelerated Filters in VTK and ParaView - Kitware, Inc., accessed November 6, 2025, <https://www.kitware.com/vtk-m-accelerated-filters-in-vtk-and-paraview/>
72. The Origins of the ParaView and VisIt Scientific Visualization Tools - IEEE Computer Society, accessed November 6, 2025, <https://www.computer.org/csdl/magazine/cg/2025/05/11184314/2apqonluk92>
73. VisIt - | HPC @ LLNL - Lawrence Livermore National Laboratory, accessed November 6, 2025, <https://hpc.llnl.gov/software/visualization-software/visit>
74. VisIt Home - GitHub Pages, accessed November 6, 2025, <https://visit-dav.github.io/visit-website/index.html>
75. Key Scientific Visualization Tools to Know for Applications of Scientific Computing - Fiveable, accessed November 6, 2025, <https://fiveable.me/lists/key-scientific-visualization-tools>
76. VisIt LibSim Tutorial, accessed November 6, 2025, <https://portal.nersc.gov/project/visit/cyrush/2018_02_ecp/2018_02_VisIt_ECP_LibSim_Tutorial.pdf>
77. VisIt Python Interface Manual - Princeton University, accessed November 6, 2025, <http://www.princeton.edu/~efeibush/viscourse/VisItPythonManual.pdf>
78. 1. VisIt Basics — VisIt User Manual 3.2.2 documentation, accessed November 6, 2025, <https://visit-sphinx-github-user-manual.readthedocs.io/en/3.4rc/tutorials/VisIt_Basics.html>
79. Discussions - visit-dav visit - GitHub, accessed November 6, 2025, <https://github.com/visit-dav/visit/discussions>
80. Visit visualization error with Silo time series · Issue #4845 - GitHub, accessed November 6, 2025, <https://github.com/visit-dav/visit/issues/4845>
81. 3. Scripting — VisIt User Manual 3.2.2 documentation, accessed November 6, 2025, <https://visit-sphinx-github-user-manual.readthedocs.io/en/3.4rc/tutorials/Scripting.html>
82. 3. Scripting — VisIt User Manual 3.1 documentation, accessed November 6, 2025, <https://visit-sphinx-github-user-manual.readthedocs.io/en/v3.2.0/tutorials/Scripting.html>
83. Introduction to VisIt — VisIt User Manual 3.1 documentation, accessed November 6, 2025, <https://visit-sphinx-github-user-manual.readthedocs.io/en/v3.2.2/cli_manual/intro.html>
84. Using CLI - VisItusers.org, accessed November 6, 2025, <https://www.visitusers.org/index.php?title=Using_CLI>
85. In-situ Visualization: State-of-the-art and Some Use Cases - ResearchGate, accessed November 6, 2025, <https://www.researchgate.net/publication/256294218_In-situ_Visualization_State-of-the-art_and_Some_Use_Cases>
86. VisIt-tutorial-in-situ - VisItusers.org, accessed November 6, 2025, <https://www.visitusers.org/index.php?title=VisIt-tutorial-in-situ>
87. Lessons from Building a Visualization Toolkit for Massively Threaded Architectures | NVIDIA, accessed November 6, 2025, <https://developer.download.nvidia.com/video/gputechconf/gtc/2019/presentation/s9458-vtk-m-lessons-from-building-a-visualization-toolkit-for-massively-threaded-architectures-v2.pdf>
88. ECP Brings Much Needed Visualization Software to Exascale and GPU-Accelerated Systems, accessed November 6, 2025, <https://www.exascaleproject.org/highlight/ecp-brings-much-needed-visualization-software-to-exascale-and-gpu-accelerated-systems/>
89. CFD visualization workflow: Visit vs Paraview vs Tecplot and others, accessed November 6, 2025, <https://scicomp.stackexchange.com/questions/23135/cfd-visualization-workflow-visit-vs-paraview-vs-tecplot-and-others>
90. ChatVis: Automating Scientific Visualization with a Large Language Model - arXiv, accessed November 6, 2025, <https://arxiv.org/html/2410.11863>
91. Why do ParaView and VisIt treat this data file somewhat differently?, accessed November 6, 2025, <https://discourse.paraview.org/t/why-do-paraview-and-visit-treat-this-data-file-somewhat-differently/11581>
92. ParaView-MCP levels the playing field for complex scientific visualization, accessed November 6, 2025, <https://data-science.llnl.gov/paraview-mcp-levels-playing-field-complex-scientific-visualization>
93. HPC Visualization and In Situ Visualization at KAUST, accessed November 6, 2025, <https://www.hpc.kaust.edu.sa/sites/default/files/2023-06/5.HPC-Vis.pdf>
94. Software Design - sensei, accessed November 6, 2025, <https://sensei-insitu.org/learn-more/software.html>
95. In Situ Analysis with Catalyst: Getting Started - YouTube, accessed November 6, 2025, <https://www.youtube.com/watch?v=WFJ_CVhEMIQ>
96. SENSEI: Cross-Platform View of In Situ Analytics - Argonne National Laboratory, accessed November 6, 2025, <https://www.alcf.anl.gov/sites/default/files/2019-08/SENSEI_tutorial.pdf>
97. In Situ Visualization Techniques for High Performance Computing - Computer Science - University of Oregon, accessed November 6, 2025, <https://www.cs.uoregon.edu/Reports/AREA-201703-Kress.pdf>
98. In Situ Visualization using VisIt, accessed November 6, 2025, <http://vis.cs.ucdavis.edu/Ultravis11/slides/SC11_InSituUsingVisIt.pdf>
99. Ascent — Ascent 0.9.5 documentation, accessed November 6, 2025, <https://ascent.readthedocs.io/>
100. Ascent: Flyweight In Situ Visualization and Analysis for HPC Simulations, accessed November 6, 2025, <https://www.olcf.ornl.gov/wp-content/uploads/Ascent_Tutorial_Sep2023.pdf>
101. Ascent: A Flyweight In Situ Library for Exascale Simulations | Request PDF - ResearchGate, accessed November 6, 2025, <https://www.researchgate.net/publication/360356298_Ascent_A_Flyweight_In_Situ_Library_for_Exascale_Simulations>
102. Flyweight In Situ Visualization and Analysis for HPC Simulations - Ascent, accessed November 6, 2025, <https://www.ascent-dav.org/tutorial/2025_07_22_ascent_tutorial.pdf>
103. Insitu Visualization using Ascent and ParaView - Kitware, Inc., accessed November 6, 2025, <https://www.kitware.com/insitu-visualization-using-ascent-and-paraview/>
104. Building Ascent — Ascent 0.9.5 documentation - Read the Docs, accessed November 6, 2025, <https://ascent.readthedocs.io/en/stable/BuildingAscent.html>
105. The SENSEI Generic In Situ Interface: Tool and Processing Portability at Scale [Book Chapter] - OSTI, accessed November 6, 2025, <https://www.osti.gov/biblio/2448487>
106. The SENSEI Generic In Situ Interface: Tool and Processing Portability at Scale, accessed November 6, 2025, <https://www.researchgate.net/publication/360361220_The_SENSEI_Generic_In_Situ_Interface_Tool_and_Processing_Portability_at_Scale>
107. The SENSEI Generic In Situ Interface | Request PDF - ResearchGate, accessed November 6, 2025, <https://www.researchgate.net/publication/313451605_The_SENSEI_Generic_In_Situ_Interface>
108. The SENSEI Generic In Situ Interface - OSTI.GOV, accessed November 6, 2025, <https://www.osti.gov/servlets/purl/1350978>
109. The SENSEI User Guide — SENSEI 4.1.0-130-gb92de09 documentation, accessed November 6, 2025, <https://sensei-insitu.readthedocs.io/>
110. SENSEI Documentation, accessed November 6, 2025, <https://sensei-insitu.readthedocs.io/_/downloads/en/rtd_user_guide/pdf/>
111. The SENSEI User Guide — SENSEI 4.1.0 documentation, accessed November 6, 2025, <https://sensei-insitu.readthedocs.io/en/stable/>
112. In situ Visualization with SENSEI — WarpX 25.10 documentation - Read the Docs, accessed November 6, 2025, <https://warpx.readthedocs.io/en/latest/dataanalysis/sensei.html>
113. System Overview & Architecture — SENSEI 4.1.0 documentation, accessed November 6, 2025, <https://sensei-insitu.readthedocs.io/en/stable/system_overview.html>
114. SENSEI ∙ Scalable in situ analysis and visualization - GitHub, accessed November 6, 2025, <https://github.com/SENSEI-insitu/SENSEI>
115. sensei - Spack Packages, accessed November 6, 2025, <https://packages.spack.io/package.html?name=sensei>
116. Amrvis — amrex 25.11-dev documentation, accessed November 6, 2025, <https://amrex-codes.github.io/amrex/docs_html/Visualization.html>
117. An Evaluative Comparison of Performance Portability across GPU Programming Models, accessed November 6, 2025, <https://arxiv.org/html/2402.08950v1>
118. Portable kernel-based models — GPU programming: why, when and how? documentation, accessed November 6, 2025, <https://enccs.github.io/gpu-programming/8-portable-kernel-models/>
119. Implementing Multi-GPU Scientific Computing Miniapps Across Performance Portable Frameworks - arXiv, accessed November 6, 2025, <https://arxiv.org/html/2511.02655v1>
120. VTK-m | A toolkit of scientific visualization algorithms for emerging processor architectures, accessed November 6, 2025, <https://m.vtk.org/>
121. The VTK-m User's Guide - INFO - Oak Ridge National Laboratory, accessed November 6, 2025, <https://info.ornl.gov/sites/publications/Files/Pub171416.pdf>
122. VTK-m - Exascale Computing Project, accessed November 6, 2025, <https://www.exascaleproject.org/research-project/vtk-m/>
123. Interest in GPU image processing pipeline - Development - VTK Discourse, accessed November 6, 2025, <https://discourse.vtk.org/t/interest-in-gpu-image-processing-pipeline/535>
124. VTK-m install - Support, accessed November 6, 2025, <https://discourse.vtk.org/t/vtk-m-install/13577>
125. Build ParaView with VTK-m CUDA enabled, accessed November 6, 2025, <https://discourse.paraview.org/t/build-paraview-with-vtk-m-cuda-enabled/4404>
126. Do we want to support GPU backend in the SMPTools? - Development - VTK Discourse, accessed November 6, 2025, <https://discourse.vtk.org/t/do-we-want-to-support-gpu-backend-in-the-smptools/5802>