# A Model-Agnostic Orchestration Engine for Multi-Stage High-Performance Scientific Workflows

Sajan Neupane
*Scientific Computing and Imaging (SCI) Institute*
*The University of Utah*
Salt Lake City, UT, USA
sajan.neupane@utah.edu

Jess Tate
*Scientific Computing and Imaging (SCI) Institute*
*The University of Utah*
Salt Lake City, UT, USA
0000-0002-2934-1453

Saleem Alharir
*Scientific Computing and Imaging (SCI) Institute*
*The University of Utah*
Salt Lake City, UT, USA
0009-0008-2315-4914

Lorien Wheeler
*NASA Ames Research Center*
Moffett Field, CA, USA
0000-0002-3298-3721

Jessie Dotson
*NASA Ames Research Center*
Moffett Field, CA, USA
0000-0003-4206-5649

Donovan Mathias
*NASA Ames Research Center*
Moffett Field, CA, USA
0000-0003-4750-3003

Manish Parashar
*Scientific Computing and Imaging (SCI) Institute*
*The University of Utah*
Salt Lake City, UT, USA
0000-0003-0983-7408

***Abstract*—Scientific workflows increasingly require the integration of independently developed models and heterogeneous data sources. We present a model-agnostic orchestration engine that represents both data sources and simulation models as producers of named variables. Given a requested target, the engine recursively resolves variable dependencies, derives an executable workflow, manages intermediate products through a shared data layer, and dispatches computation through configurable execution backends. We demonstrate the approach using an asteroid-impact wildfire scenario in which a requested fire-spread product automatically resolves the data required to execute WRF-SFIRE on high-performance computing resources. The prototype demonstrates target-driven workflow generation, dependency resolution, provenance tracking, and intermediate-product reuse, providing a foundation for multi-stage cascading-hazard workflows.**

***Index Terms*—scientific workflows, workflow orchestration, model coupling, high-performance computing, dependency resolution, cascading hazards**

## I. Introduction

Many scientific questions span multiple physical processes, each represented by independently developed models with different inputs, data formats, representations, and computational requirements. Composing these models into integrated workflows often requires manual, scenario-specific data preparation and coupling. We present a model-agnostic orchestration engine that represents both data sources and simulation models as producers of named variables. Given a requested target, the engine resolves dependencies, derives an executable workflow, dispatches stages through configurable execution backends, and records intermediate products for reuse and provenance.

## II. The Gap and Objectives

Integrating independently developed scientific models remains challenging because models are built and maintained by different scientific communities and often use incompatible inputs, formats, resolutions, and execution environments. Existing workflow and model-coupling approaches can connect these components, but their relationships and data exchanges typically require explicit workflow configuration.

Intermediate products may also be repeatedly fetched, transformed, or recomputed. Our framework therefore targets three objectives: (1) describe heterogeneous data sources and models through a common contract; (2) automatically derive an executable workflow from a requested target based on variable dependencies; and (3) discover, transform, track, and reuse intermediate products through a shared data layer.

## III. Architecture

Both data sources and simulation models are represented uniformly as *producers*. Each producer declares the variables it requires and produces, together with request constraints such as spatial domain, temporal window, and resolution. The engine is model-agnostic because its dependency resolution and execution logic operate only on this common producer contract and named variables, rather than on model-specific internal implementations.

A terrain source, for example, produces elevation data, while WRF-SFIRE declares required meteorological, terrain, fuel, and ignition variables and the fire products it generates. Given a requested target variable, the dependency resolver identifies a producer capable of generating it and recursively resolves that producer's requirements. Resolution terminates when each requirement is satisfied by an existing product or by a source producer. This process derives the workflow automatically rather than requiring direct model-to-model connections.

Each producer reads from and writes to a shared, variable-addressed data layer. Existing products are reused when they satisfy the requested constraints. Required transformations,

This research was funded in part by the US National Science Foundation via awards 2333609, 2440195, and 2504592.

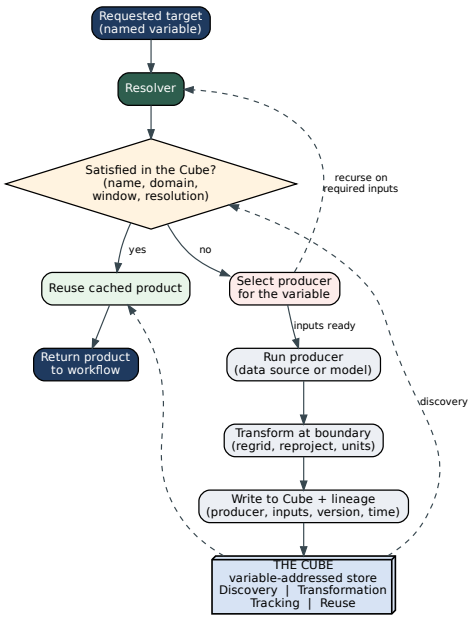



Fig. 1. Dependency resolution against the shared data layer. Existing products are reused when they satisfy the request; otherwise, the resolver recursively selects producers and resolves their requirements.

such as regridding, reprojection, and unit conversion, are applied at producer boundaries. Generated products are cataloged with provenance describing their producer, inputs, version, and execution time.

## IV. Composition and Execution

Dependency resolution produces a directed acyclic graph whose edges represent variable dependencies. Dependent stages execute in order, while producers with satisfied dependencies may run concurrently. Execution is separated from dependency resolution through configurable serial, process, Dask, and SLURM backends, allowing producers to remain independent of machine-specific execution logic.

## V. Application: Impact Cascades

An asteroid impact can trigger a cascade of hazards, including wildfire, smoke, runoff, and downstream exposure (Fig. 2). The corresponding models are developed and maintained by different scientific communities, making their integration challenging.

Our orchestration engine connects these models through the variables they require and produce. For example, a request for smoke concentration can resolve thermal pulse, fuel, wind, and terrain as inputs to WRF-SFIRE, whose fire-arrival and burned-area products can feed a downstream smoke model (Fig. 3). Shared variables such as wind can be reused rather than independently retrieved for each stage. The current prototype executes the workflow through the wildfire stage.

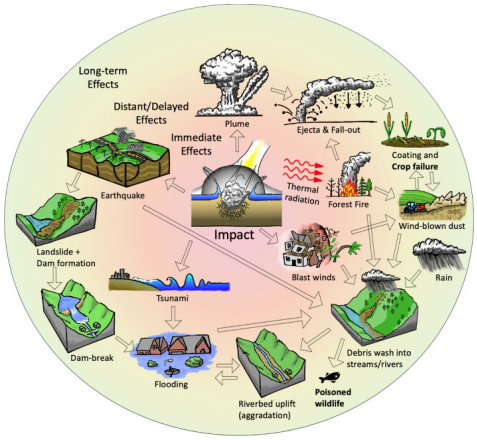



Fig. 2. Asteroid-impact hazard cascade spanning immediate, delayed, and long-term effects.

## VI. Results and Discussion

The current prototype executes the wildfire stage using WRF-SFIRE on the University of Utah Center for High Performance Computing (CHPC). A seven-day scenario near Dallas, Texas, was run on a 300 m fire grid using resolved meteorological, terrain, fuel, and ignition inputs. The simulation produced fire-arrival time, rate of spread, and fireline intensity (Fig. 4).

Starting from the requested target, *rate of fire spread*, the engine resolved the required dependencies, staged the necessary inputs, executed WRF-SFIRE through the SLURM backend, and cataloged the resulting products. The experiment evaluates the orchestration behavior of the framework rather than the scientific performance of WRF-SFIRE. It demonstrates target-driven execution, recursive dependency resolution, provenance-aware product management, and product reuse within the prototype workflow.

## VII. Conclusion

A common producer contract allows heterogeneous data sources and scientific models to expose their requirements and outputs through named variables. From a requested target, the engine derives and executes the required workflow while tracking intermediate products for reuse. The current prototype demonstrates this approach through a seven-day WRF-SFIRE wildfire simulation executed on CHPC.

## VIII. Future Work

Near-term work will extend the framework to scenario ensembles on the NASA Advanced Supercomputing (NAS) facility and continue the cascade beyond wildfire to smoke, runoff, and exposure. A longer-term direction is agentic planning, where an agent reasons over available producers and candidate workflows while the deterministic engine remains responsible for validated dependency resolution and execution.

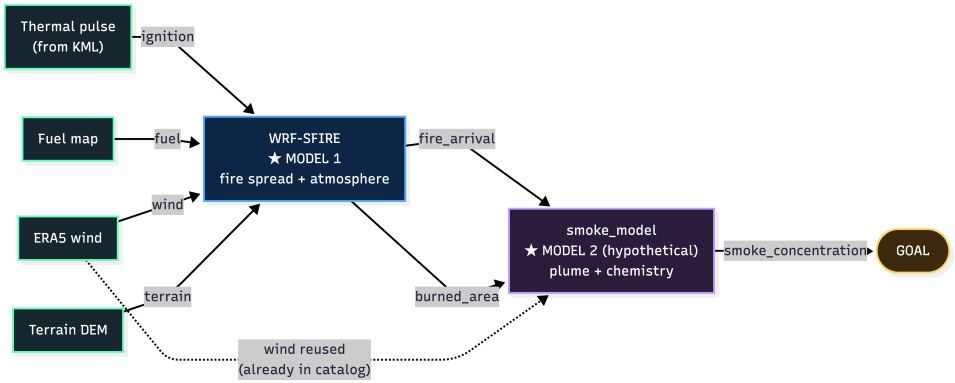



Fig. 3. Example dependency resolution for smoke concentration. WRF-SFIRE consumes thermal pulse, fuel, wind, and terrain, while shared wind data can be reused by downstream stages.

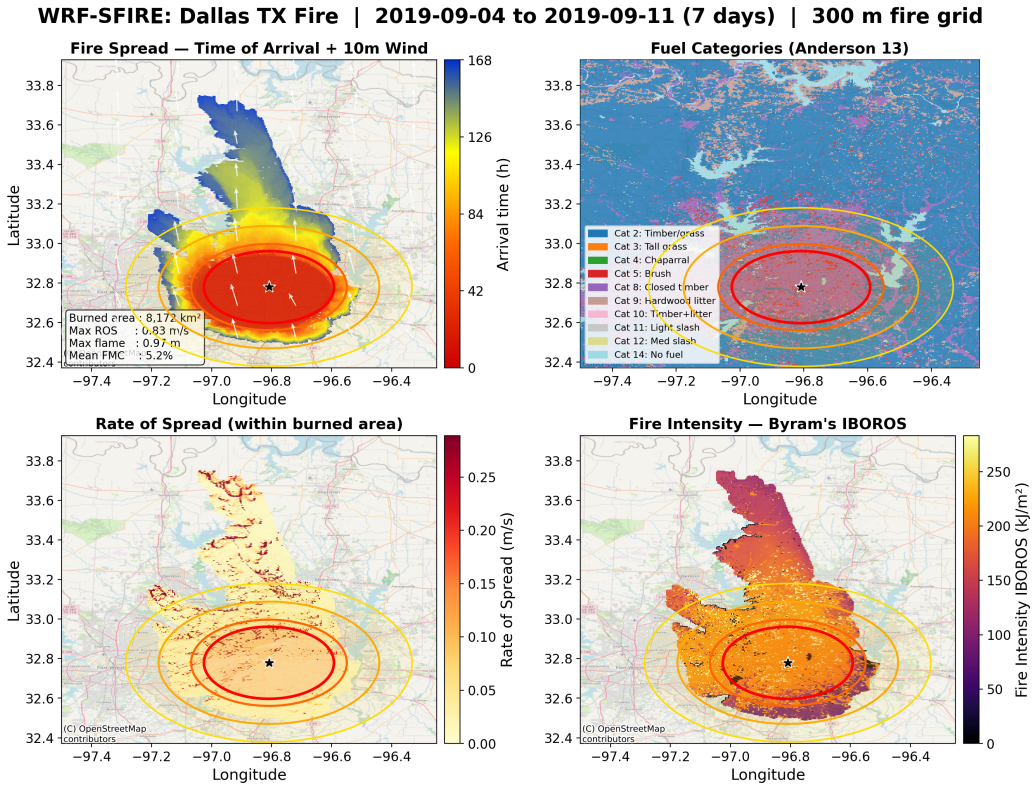



Fig. 4. Seven-day WRF-SFIRE scenario near Dallas, Texas, on a 300 m fire grid, showing resolved fuel categories together with simulated fire-arrival time, rate of spread, and fireline intensity.