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

github.com/sci-ndp/NASA_Project

Sajan Neupane[1] **Jess Tate**[1] **Saleem Alharir**[1] **Jessie Dotson**[2] **Lorien Wheeler**[2] **Donovan Mathias**[2] **Manish Parashar**[1]

[1]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, UT 84112, USA · [2]NASA Ames Research Center, Moffett Field, CA 94035, USA

sajan.neupane@utah.edu · manish.parashar@utah.edu

01 INTRODUCTION

Many scientific questions span several physical processes, each modeled by a code from a different community with its own inputs, formats, grids and runtime. Composing them is manual work, and the result is fragile and specific to one scenario.

This work presents an orchestration engine that is agnostic to the codes it runs. Data sources and models are described uniformly as **producers of named variables**. Given a set of these producers, the engine resolves dependencies, generates the workflow, runs the stages on HPC, and records lineage for everything it writes.

02 THE GAP

- **Isolated models.** Models are developed independently and share no common integration interface, so every pairing needs its own reconciliation of grids, units and time bases.
- **Inefficient data management.** Intermediate products are repeatedly fetched, transformed or recomputed, and nothing records where they came from.
- **Non-automated workflows.** Model relationships and data exchanges are configured by hand, one workflow at a time.

04 PRODCUERS AND THE RESOLVER

A **producer** is anything that can supply a named variable. A weather archive supplies wind because it has it; a fire model supplies burned area because it computes it. Both register the same contract.

> **The engine is model agnostic:** it reasons only over the variables each producer requires and produces, never over its internal implementation.

The **resolver** is given a variable, not a plan. It finds a producer that makes it, then asks the same of everything that producer requires. A branch ends where the store already holds the variable, and what remains is the set of stages and their order.

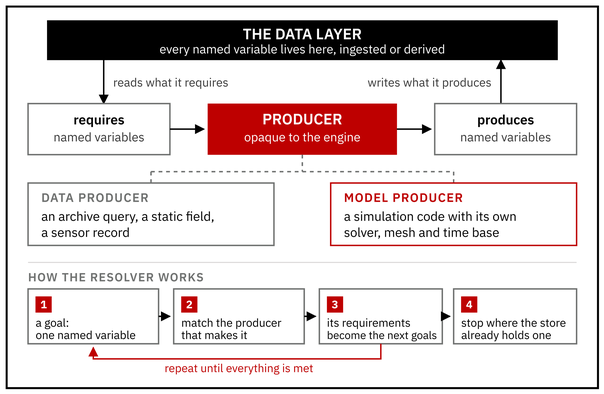


Fig. 1. The Producer Contract and Resolver used to compile the workflow in the orchestration engine. A Producer declares the variables it requires and produces, reads the first from the data layer and writes the second back, so it never addresses another producer. The resolver works over those declarations alone: it matches on variable name, then on domain, window and resolution, and resolves each variable.

03 OBJECTIVES

- Describe any data source or model through a uniform contract.
- Build and execute a workflow based on a requested data target.
- Discover, transform, track and reuse intermediate products automatically.

05 THE DATA LAYER

THE CUBE
Ingested and derived products alike, addressed by name

Producer A	↕	Producer B
writes a product		reads a product

no producer ever addresses another; grids, projections and units are reconciled here

Metadata
Every product is indexed by name, domain, window and resolution; the catalogue is consulted before any producer runs.

Transformation
Regridding, reprojection and unit conversion happen at the boundary, not between producers.

Provenance
Every write records its producer, inputs, version and time.

Data Caching
A satisfying product skips its producer entirely, within a run and across runs.

Fig. 2. The Cube features. Ingested and derived products are stored together, addressed by name, domain, window and resolution, and cataloged by their metadata. Data are transformed to a unified domain grid, and every operation is tracked in the provenance layer.

06 MODEL COMPOSITION WORKFLOW

The chain is derived, not predefined.

Goal the user asks for
↓
Resolver: which Producers, in which order
↓
Model 1
needs Input 1 and Input 2
makes **Output A**
↓
Model 2
needs Input 1 and Output A
makes **Output B**
↓
Model N
needs selected data and Output B
makes **Output N**
↓
Final output

↑ every output is written back to the data layer, so the next model or the next run can reuse it

Fig. 3. Workflow resolution. From the goal the user names, the resolver works backward until every input is stored or produced by another stage.

07 APPLICATION: IMPACT CASCADES

An asteroid impact is not one hazard but a chain of them [1]: thermal radiation ignites fires, fires produce smoke, and burn scars change runoff, with people and property in the path of every stage. Combining predictions of all these phenomena is a challenge, because the domains and communities involved manage their own data repositories, software, standards and formats.

We applied our orchestration engine to a cascading hazard question to demonstrate that such a chain can be assembled and run from a single named goal.

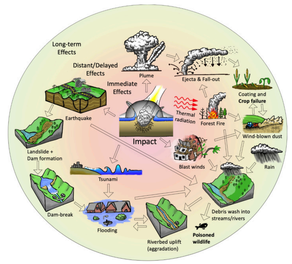


Fig. 4. The cascading hazards that might occur as a result of a midsized (100m -1 km) asteroid impact. Adapted from [1].

7.1 CASCADE UNDER STUDY

1. **Asteroid on an impact trajectory**
   Size, speed and entry angle set how much energy arrives.
2. **Impact and thermal radiation**
   The blast and heat pulse are mapped onto the ground.
3. **Wildfire ignition and spread**
   The heat map seeds ignitions; fire spreads with wind, fuel and terrain. Modeled with WRF-SFIRE [3].
4. **Smoke and air quality**
   The fire's heat and burned area drive smoke transport downwind.
5. **Hydrology**
   Burn scars change soil properties and increase runoff, causing mudslides and flooding

7.2 RESOLVING A REQUEST

Asking for smoke concentration is enough. The engine finds the stages that reach it and the inputs each one needs, without being told the chain.

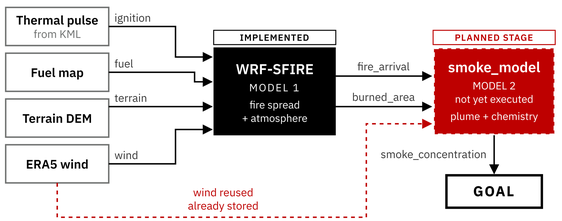


Fig. 5. Derived Workflow when asked for smoke concentration. Thermal pulse, fuel map, ERA5 wind and terrain are known inputs to WRF-SFIRE, whose products, arrival time and burned area, drive the smoke model. Wind is requested twice but fetched once, the dotted path.

implemented and executed · planned downstream stage

08 RESULTS

> **We evaluate the orchestration, not the fire physics.**
>
Requested target	rate of fire spread
Resolved automatically	ignition, fuel, terrain, meteorology
Execution	WRF-SFIRE via SLURM on Utah CHPC, 7 days, 300 m fire grid

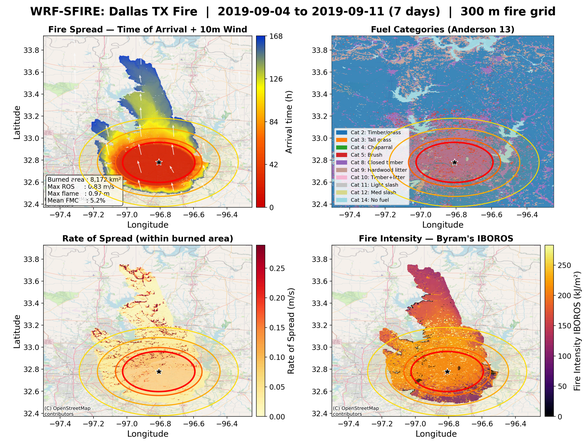


Fig. 6. Products of the resolved run near Dallas, TX: arrival time, Anderson 13 fuel category, rate of spread and Byram fireline intensity from WRF-SFIRE [3].

09 CONCLUSIONS

- One contract is sufficient for both data producers and simulation producers, so a new code is added by declaring what it requires and produces.
- A requested variable can be resolved into an executable workflow with no hard coded model to model connections, reusing stored products rather than recomputing them.
- The WRF-SFIRE run validates target driven resolution and execution on HPC, and the same mechanism extends to the remaining cascade stages.

10 FUTURE WORK

- **Run the full downstream cascade as one combined batch.** Carry the chain past fire into smoke and runoff, and submit every stage of every scenario as one batch, scheduled in a single allocation on NASA NAS.

AGENTIC PLANNING

An agent interprets a goal and proposes the run plan; the engine validates dependencies and executes it. The agent decides:

- which models to chain, and which data to draw on
- what resolution each stage runs at, and how much compute to request

REFERENCES

[1] T. Titus, D. Robertson, J. B. Sankey, L. Mastin, and F. Rengers, "A review of common natural disasters as analogs for asteroid impact effects and cascading hazards," *Nat. Hazards*, 2023, doi: 10.1007/s11069-022-05722-z.

[2] L. Wheeler, J. Dotson, M. Aftosmis, A. Coates, G. Chomette, and D. Mathias, "Risk assessment for asteroid impact threat scenarios," *Acta Astronaut.*, 2024, doi: 10.1016/j.actaastro.2023.12.049.

[3] J. L. Coen, M. Cameron, J. Michalakes, E. G. Patton, P. J. Riggan, and K. M. Yedinak, "WRF-Fire: Coupled weather–wildland fire modeling with the Weather Research and Forecasting model," *J. Appl. Meteorol. Climatol.*, vol. 52, no. 1, pp. 16–38, Jan. 2013, doi: 10.1175/JAMC-D-12-023.1.

ACKNOWLEDGEMENTS

This research was funded in part by the US National Science Foundation via awards 2333609, 2440195, and 2504592. We also thank NASA Ames Research Center for the asteroid problem formulation, supporting materials, and future compute facilities. The authors thank the National Data Platform and the Science Data Exchange team for their collaboration and support.