An AI-Ready Data Harness for Fusion: The Fusion Data Platform

Brian Sammuli

General Atomics

on behalf of the FDP and FEDER teams

ALCF Service Enabled Science Series
July 16, 2026

Work supported by the U.S. Department of Energy under Award No. DE-FC02-04ER54698 and Award No. DE-SC0024426.

What fusion experimental data looks like

Each DIII-D experiment is a "shot": one plasma pulse, ~7 seconds long, with a numeric ID.

- Thousands of signals per shot, from hundreds of diagnostics
- Raw sensor data plus derived physics quantities
- Analysis is often multi-shot: trends across many shots at once

DIII-D, IN NUMBERS

~200k shots over the facility's lifetime

~10k+ signals recorded per shot

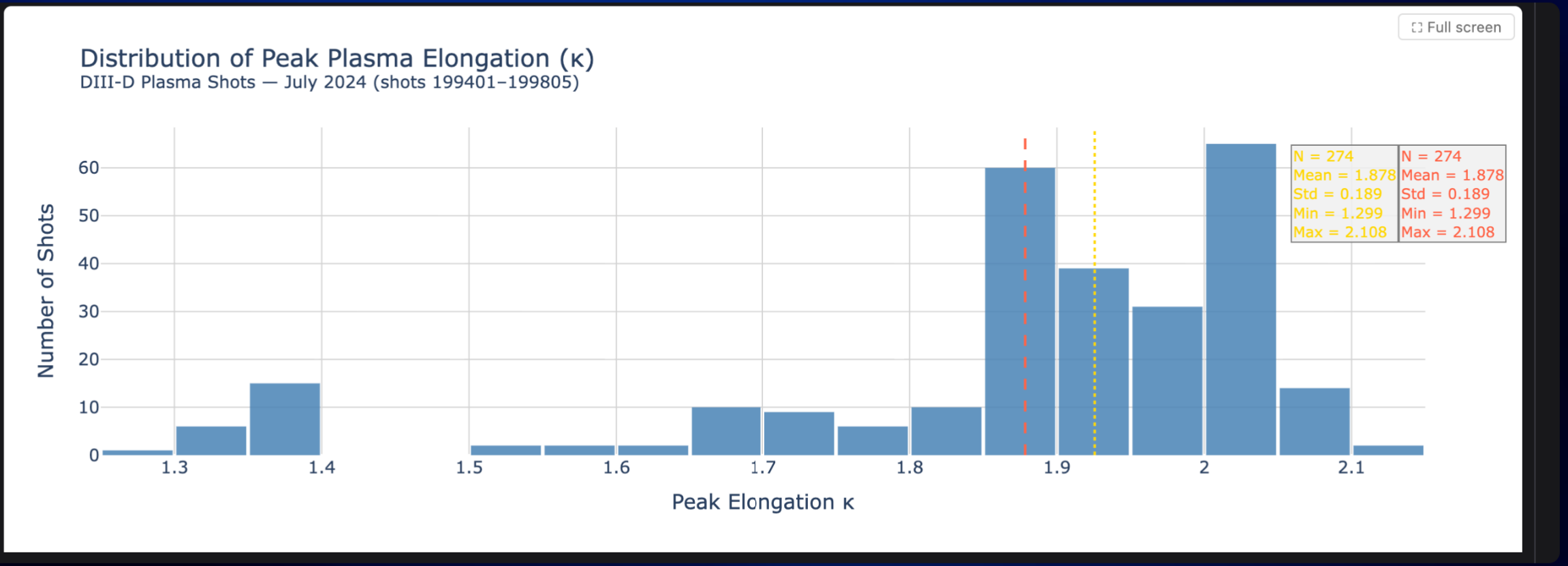
~PB scale of the full data archive

DIII-D is the largest tokamak operating in the US, run by General Atomics for the Department of Energy.

Let's start with an example: a plain-English question, answered end to end using Fusion Data Platform (FDP) tools

… and the agent produces a working result, end to end

- The plan runs to completion: Signal lookup, single-shot test, 274-shot pipeline, plot
- Every LLM call served by the **AmSC endpoint**



… and writes up what it found

- Statistics, observations, and a back-reference to the actual signal path used

Here's the distribution of **peak plasma elongation (κ)** for all 274 DIII-D plasma shots in July 2024 (shots 199401–199805). Here's a summary of what the data shows:

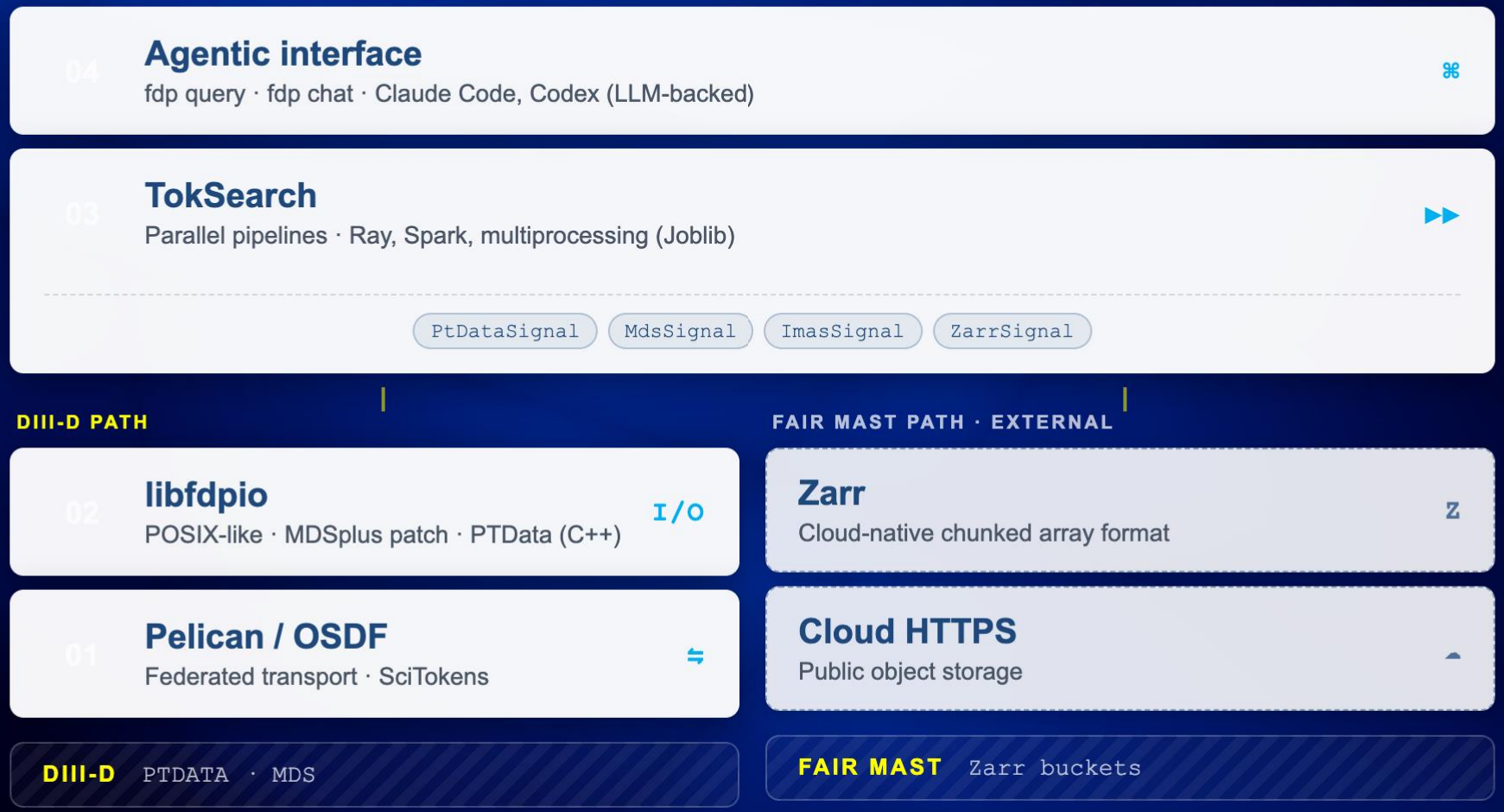
Statistic	Value
N (shots)	274
Mean κ	1.878
Median κ	1.926
Std dev	0.189
Min κ	1.299
Max κ	2.108

Key observations:

- **Most shots cluster around κ ≈ 1.87–2.01** (the interquartile range), reflecting DIII-D's typical highly-shaped, diverted plasmas.
- **The distribution is left-skewed** — a tail of lower-elongation shots pulls the mean (1.878) below the median (1.926), likely corresponding to circular or weakly shaped early/test discharges.
- **Peak κ up to ~2.1** is achieved in the most highly shaped shots, consistent with DIII-D's capability for advanced tokamak configurations.
- The data was fetched via `ImasSignal('equilibrium.time_slice.boundary_separatrix.elongation')` from the EFIT01 equilibrium reconstruction.

The FDP software stack makes this possible

- Common interface for multiple fusion devices
- Available anywhere in the world, with no facility network access needed



FDP data-movement infrastructure is deployed at multiple supercomputing centers

Pelican / OSDF (Open Science Data Federation): a content delivery network for scientific data.

- Standard origin / cache model
- Built on XRootD, proven at scale in high-energy physics (LHC data distribution)
- SciTokens for simple auth
- Caches give data locality automatically — lower latency, near the compute
- Live today: origin at GA; caches at NERSC and SDSC; ALCF and PPPL in progress

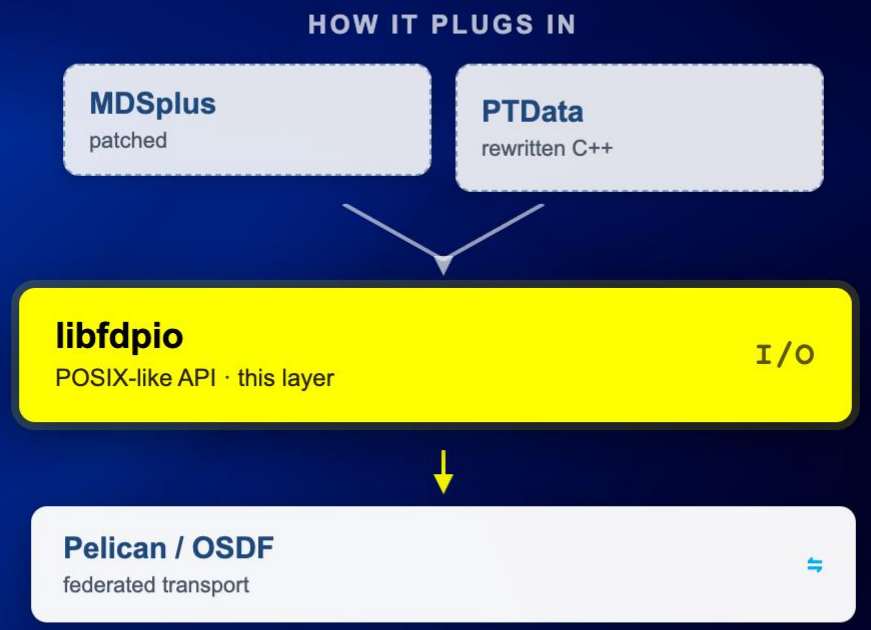
The wider OSDF spans many origins and caches across North America and Europe. FDP plugs into this existing fabric.

BUILT ON

FDP offers multiple straightforward ways to access data

Two complementary access paths, so existing tools integrate with little effort:

- libfdpio — a small, embeddable POSIX-like client library
  - Drops into C / C++ / Fortran applications
  - Already in MDSplus and the PTData client
- fsspec — for Python, provided separately by Pelican
- Transport is HTTP (libfdpio uses it under the hood); the root / xroot protocol also works
- Existing tooling works from anywhere, no rewrites



Fusion clients call `libfdpio`; it does the work of speaking to caches and origins.

A brief primer on TokSearch

- The primary tool for accessing and processing data on FDP
- Massively parallel — a dataframe-style pipeline, keyed by experiment ("shot")
- Backends swap freely: Ray · Spark · multiprocessing

```python
from toksearch import Pipeline
from toksearch_d3d import PtDataSignal

p = Pipeline(shots)
p.fetch_dataset(                                # FETCH - built-in
    'ds',
    {
        'ip': PtDataSignal('ip'),
        'bt': PtDataSignal('bt'),
    }
)
p.align('ds', 'bt')                             # ALIGN - built-in
p.map(lambda r: r['ip'] * r['bt'])              # MAP   - user-defined
p.keep(['ip_bt'])                               # KEEP  - built-in
p.where(lambda r: r['ip'].max() > 1e6)          # WHERE - user-defined
results = p.compute_multiprocessing(num_workers=20)
```

TokSearch data access uses a simple, extensible abstraction for many data sources

TokSearch is easy to point at new data. Each source is a small `Signal` subclass with a uniform interface, whether it's another archive within a tokamak or a different device entirely.

PtDataSignal	PTData client (rewritten) → libfdpio → Pelican / OSDF → **DIII-D**
MdsSignal	MDSplus (patched) → libfdpio → Pelican / OSDF → **DIII-D**
ImasSignal	IMAS Composer → *resolves to PTDATA / MDS signals*
ZarrSignal	Zarr → cloud HTTPS → **FAIR MAST**

Supporting a new data source, within DIII-D or on another device, means writing one more `Signal` subclass. Everything above it (pipeline, agentic interface) is untouched.

A common vocabulary across devices: the IMAS schema

- IMAS — the fusion community's standard data schema
- One shared name for each physics quantity, across every device

```python
from toksearch_d3d import ImasSignal

# same request works on any supported device
beta_n_sig = ImasSignal('equilibrium.time_slice.global_quantities.beta_normal')
```

- The IMAS Composer maps each device's native signals onto these paths
- Write an analysis once, run it across DIII-D, MAST, and others

An MCP server grounds the agent, with skills embedded in the TokSearch package

- Skill docs ship inside the package(s)
- Loaded lazily — **no upfront context bloat**

MULTIPLE WAYS OF USING IT

`fdp query "…"`	Single-shot question from the CLI
`fdp chat`	Multi-turn terminal application
`fdp chat --gui`	Same agent, browser-based GUI
`Claude Code · Codex`	Any standard agentic harness pointed at your repo

How do we know the agent is any good? A golden query suite

A benchmark of real physicist requests, each with an expert-written "gold" pipeline as the answer key.

- 18 plain-English queries
- Each run 10 times, auto-scored per shot
- Model alone vs. with MCP

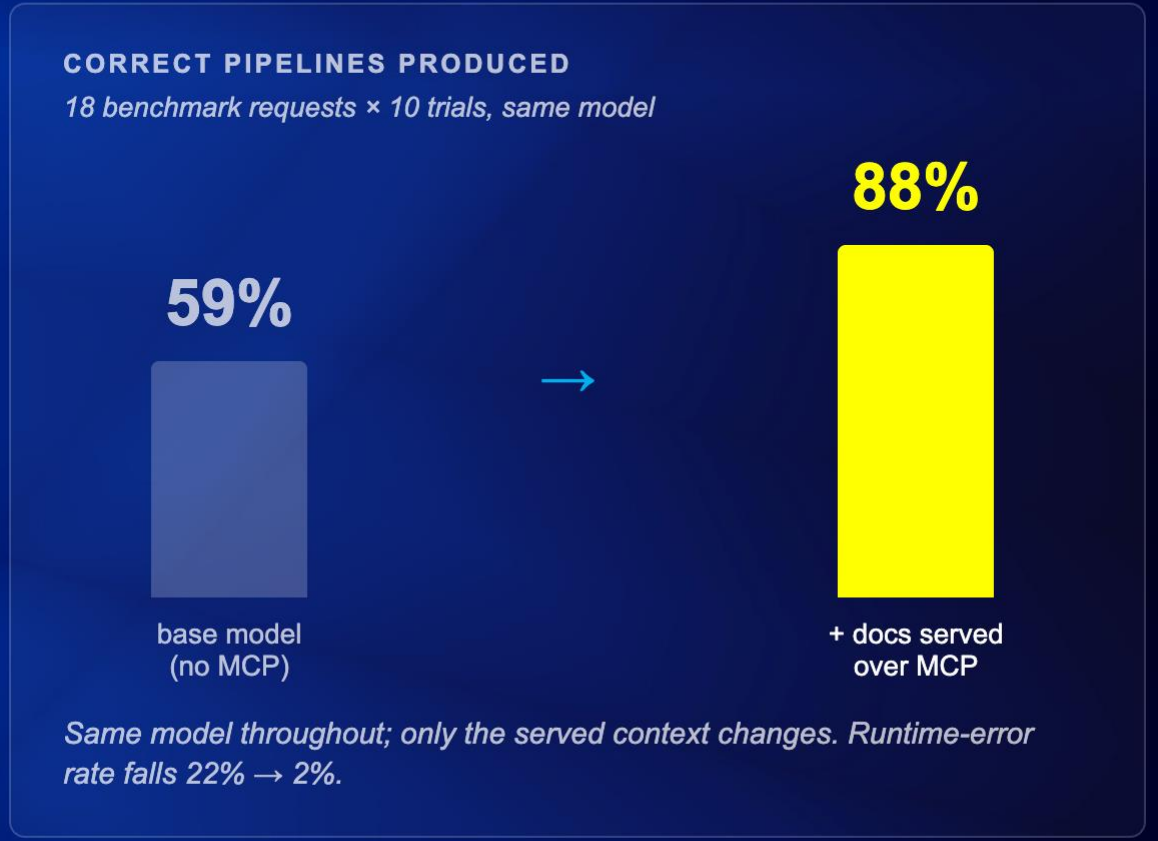
A RUN COUNTS AS CORRECT WHEN

- It executes without error
- Matches the gold result within 1% per shot
- For ≥ 95% of the shots

The base model already knows DIII-D; it just lacks the context to query it

Claude Sonnet gets a majority of queries right with no help at all**; what it lacks is the context to route to the right signal.**

- Strong even without MCP — the base model knows the domain
- The MCP server serves version-synced docs on demand
- Agent looks up real signals, API usage, and available data + metadata before writing the pipeline
- With MCP, most remaining "wrong" results trace to ambiguous or contradictory data, not agent error



Deep dive 1: composing two signals the base model can't route

VERBATIM PROMPT

```
For the shots in EVAL_SHOTS, compute peak normalized beta divided by peak total NBI
power (MW) per shot (only for shots with NBI).
```

0% → **100%**

no MCP | with MCP

WHY IT FAILED WITHOUT MCP

- Both quantities need non-obvious routing: normalized beta from an EFIT node, NBI power summed over the eight beam units as an IMAS object-array
- Without that context the base model couldn't locate *either* signal — all ten runs failed to execute
- With the routing docs in session, the two-signal compose is exact every time — the single largest MCP effect in the benchmark

Deep dive 2: when it runs, but silently wrong

VERBATIM PROMPT

```
For the shots in EVAL_SHOTS, resample plasma current (Ip) and stored energy (Wmhd)
onto a common uniform 10 ms time grid using zero-order-hold (pad / forward-fill)
alignment, then report the Pearson correlation coefficient between them per shot.
```

0% → **100%**

no MCP | with MCP

WHY IT FAILED WITHOUT MCP

- The dangerous failure mode: the code *ran*
- Nine of ten no-MCP runs improvised their own time-alignment and landed ~17% off the correct correlation — a plausible-looking wrong answer
- MCP points the agent at the documented `align` / `fetch_dataset` path, and the result becomes exact

Deep dive 3: when MCP can't fix an ambiguous ask

VERBATIM PROMPT

```
For the shots in EVAL_SHOTS, compute how long (in ms) the plasma current stayed above
1 MA in each shot.
```

30% → **30%**

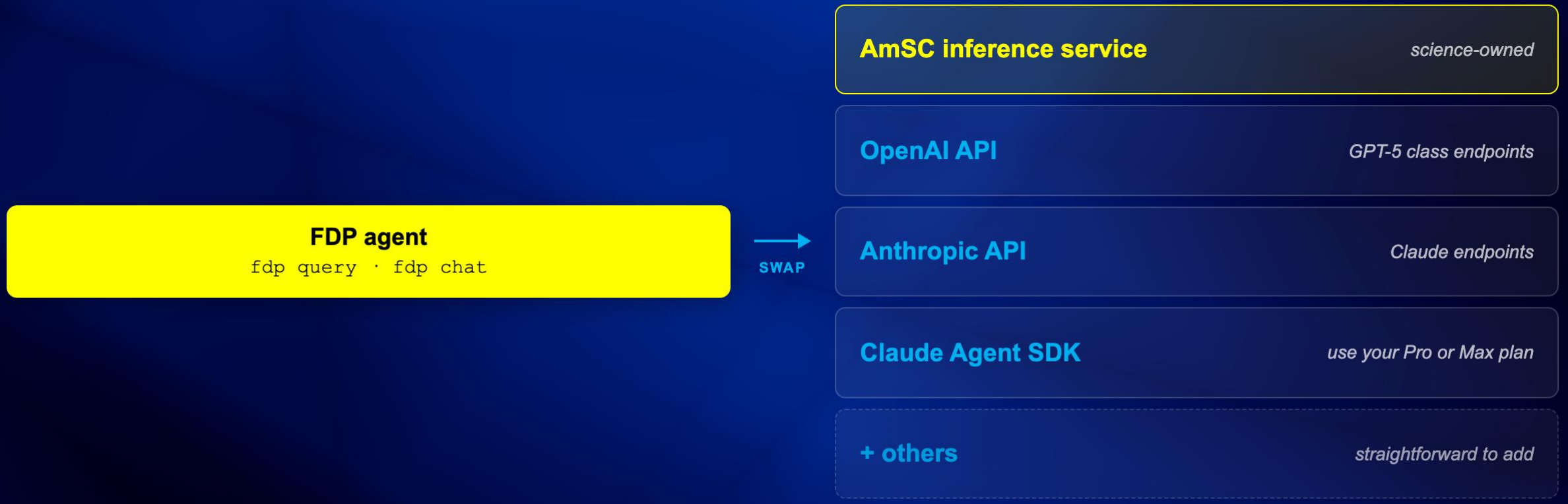
no MCP | with MCP

WHY MCP DOESN'T MOVE THE NEEDLE

- Signal routing is trivial here; the failure isn't tool use
- The *task* is under-specified: dwell-time above a threshold has real ambiguity (contiguous vs. total time, how to treat brief dips), and seven of ten runs return all-zeros
- MCP grounds signal access, not task intent; the remaining gap is a spec problem, and we report it directly

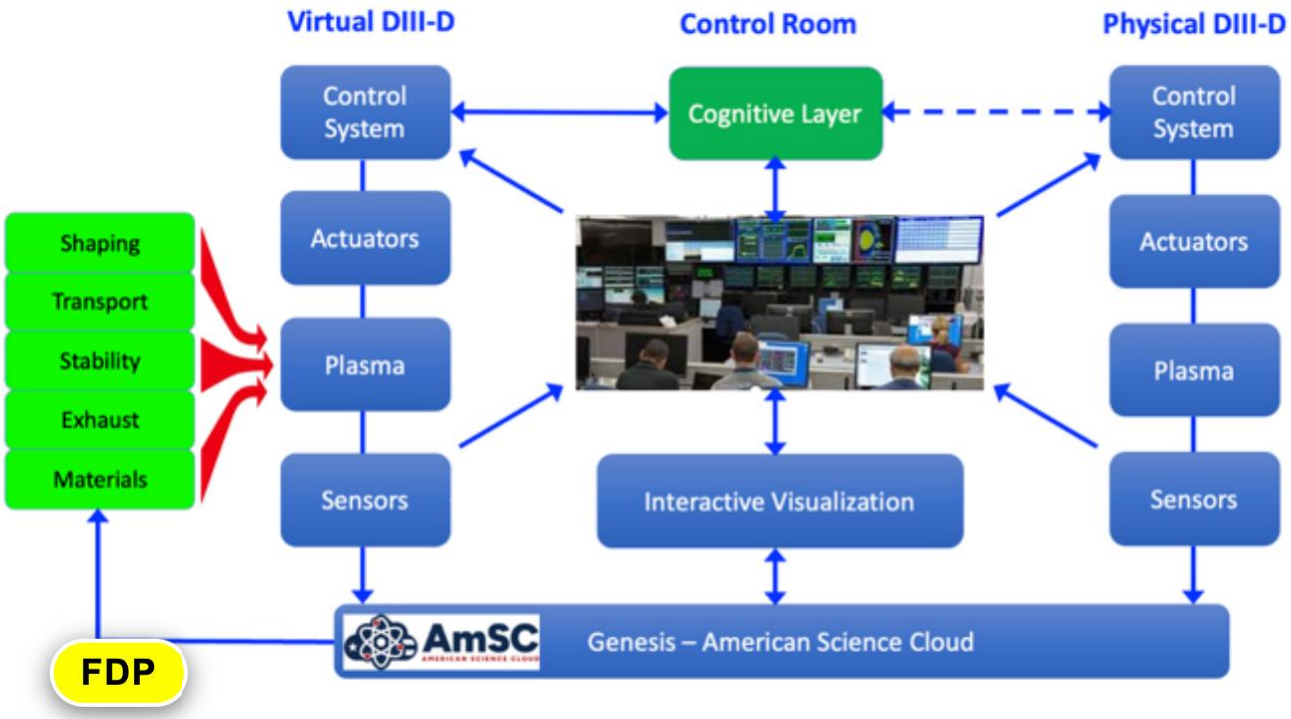
Pointing the agent at the AmSC inference service

- The LLM is a pluggable backend — any OpenAI- or Anthropic-compatible endpoint
- We point it at the AmSC inference service
- Everything stays in a controlled, US-based environment
- Public model for open work, AmSC for sensitive work (as institutional policies / data-usage agreements dictate)



The DIII-D digital twin takes advantage of this entire stack

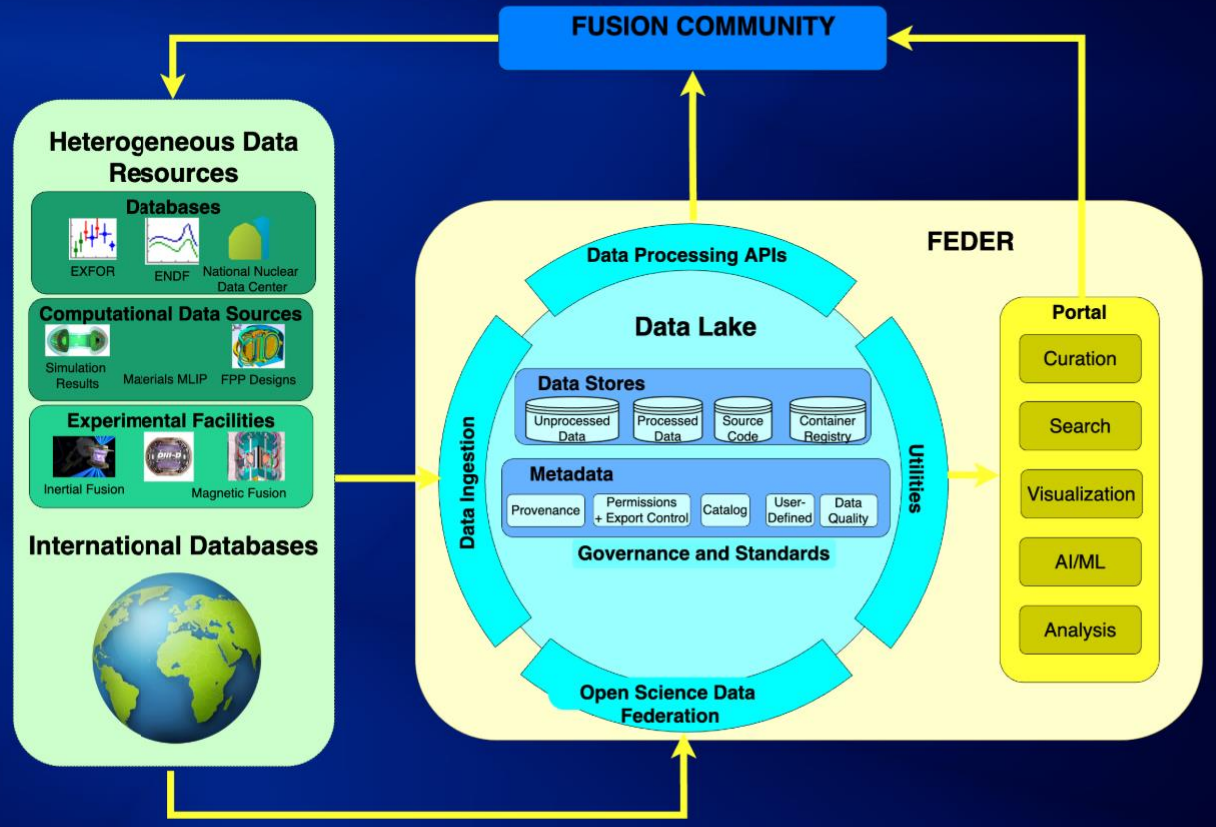
- Pelican + libfdpio for data access from the virtual PCS
- TokSearch for evaluating between-shot vPCS simulation runs at scale
- Agent layer available end-to-end for inspection and steering



FEDER extends FDP to more data types and sources

Fusion Energy Data Ecosystem and Repository

- Engineering data, e.g. MOOSE workflows
- Materials databases
- More tokamaks, inertial fusion experiments
- Federated data lake — *logically centralized, physically federated*
- Built on the same FDP stack



FEDER data is processed into a knowledge graph for richer agent context

Ingestion does more than store data: it builds a knowledge graph the agent can reason over.

- Currently developing formal ontology and designing architecture
- Graph captures relationships data alone can't:
  diagnostic ↔ *shot* ↔ *campaign* ↔ *model* ↔ *paper*
- The agent gains grounding it didn't have before:
  - Disambiguation: "which Ip do you mean?"
  - Provenance: which dataset, which version, which prior model
  - Discovery: "show me all shots with these conditions and a published analysis"
- Lays the foundation for cross-device, cross-experiment reasoning under FEDER

Wrapping up

FDP today: a full stack from federated bytes to natural-language queries.

- One common interface, any machine in the world, with no facility network access required
- Multi-device today: DIII-D + MAST through a shared IMAS schema; C-Mod, WEST, KSTAR in the works
- Regional caches live at NERSC and SDSC; agent reasoning runs on the AmSC inference service
- Substrate for both the DIII-D digital twin and FEDER

Mostly familiar patterns (federated access, a shared schema, a grounded agent) assembled to fit a facility-scale problem.

GET INVOLVED

sammuli@fusion.gat.com

clarkm@fusion.gat.com

Reach out for access, integration questions, or to join FEDER's working groups.

ga-fdp.github.io

FDP FUSION DATA PLATFORM DIII-D GENERAL ATOMICS