





August 2022

Industrial Experience Deploying Heterogeneous Platforms for Use in Multi-Modal Power Systems Design Workflows

A. Gallo, I. Claydon, E. Tucker, R. Arthur
GE Research

Hello and welcome. The title of our paper is "Industrial Experience Deploying Heterogeneous Platforms for Use in Multi-Modal Power System Design Workflows". We intend to primarily discuss today what is implied from that title – platforms for enabling workflows across heterogeneous systems.

The workflow systems that we will describe have been derived iteratively from our industrial experience and our associations with, and use of, various national laboratory facilities and software packages. Today we are applying these workflow tools at GE in several ways including in machine-learning enabled designs of multi-modal power systems.

My name is Andy Gallo and I'm a software architect at GE Research. I represent here today a team including members of our mechanical engineering research and high-performance computing leadership teams who have been working in a codesign fashion for several years now refining these concepts and applying them in real-world industrial problems.

Real World Example: Gas Turbine Design

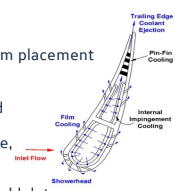
Reducing Low Cycle Fatigue

A classic trade-off

Thermodynamic Performance vs Part Life

Traditional Design Process:

- Meshes, cooling holes and film placement described manually by SME
- 2-3 design options evaluated
- Each process step is separate, and human-driven.
- Low, no ability to integrate field data



The goal: Evaluate millions of design options

Workflow Enablement

- Describing a design practice in workflow is a critical first step to introducing augmented decision-making via novel AI/ML/simulations
- Atomization, modularity and loose coupling are table stakes.
- Machine automation, optimization, validation and sharing is only possible to the extent that we can gather the necessary metadata and match/surpass human performance

Software Defined Everything

- In a heterogeneous platform, all aspects of the infrastructure are software defined and controllable.
- Future workflows will be increasingly M x N, requiring software enabled definition of tenancy capabilities, requirements and contracts.
- Building good software means Knowing Controlling and Reviewing... A full and continuous lifecycle approach

Common Data Model

- In a heterogeneous, software-defined modeling system, everything is data
- Capturing & sharing workflow metadata enables everything from model training & decision provenance to zero trust security & multi-tenancy
- Decisions will increasingly require workflows that span domains of expertise, but the platform must be domain language agnostic

As a concrete example: the trade-offs between thermodynamic performance and part life when designing the placement of cooling holes and film on a blade presents a classic challenge when designing turbomachinery to be both highly performant and highly durable. Historically, the industry has relied on deep domain knowledge and decades of validated, human-driven experience to govern these design practices and provide new insights. While simulations have been an important aspect of informing the overall design of turbomachines for decades, the cost of computation, bottlenecks in hardware and software performance, challenges of integrating data and models across different scales of fidelity, and a myriad of other lesser challenges, have constrained our ability to adopt high fidelity and machine learning-directed design decision systems into daily practice.

Resolving this bottleneck has taken years of effort and investment in hardware, software, talent and methods, and will continue to require years more of our focus as we continue to mature our capabilities. That said, we have been governed by a consistent set of principles thus far and throughout, and we would like to highlight them here briefly.

First, the design practice must become workflow enabled, less application-centric and prone to “swivel chair” type gymnastics on the part of the user. Further, those workflows must be modularized, atomized, and loosely coupled, while at the same time designed to capture all of the metadata required to enable future automation and optimization of workflows.

The second guiding principle is that the software definition of everything is paramount. Starting from the network layer up, the next generation of technologies are becoming more abstract and software defined than ever before. Good software development and sustainment practices must become daily practice even for those who build our computing infrastructures, design next generation models and methods, and develop and use the forthcoming wave of model-directed turbomachinery design tools.

Finally, as shown as far back as von Neumann, everything is data, even the software used to analyze data and the decisions we make with that data are data. And metadata? That's data too. The more effective our system can be in capturing data in a low friction manner, and the more seamlessly we can enable sharing in both context-free and context-rich ways, the more value we generate for our product design practices and ultimately our customers.


Designing a heterogenous design platform

Insights & lessons learned

The marketplace context

"Cambrian explosion", high barriers to entry

- Heterogeneous complexity → **loosely coupled** data, tooling & compute fronted by a secure **API**
- To yield actionable insights at all stages of the **digital thread** → **metadata** for (data, tools, compute, et al) – 5Ws, provenance

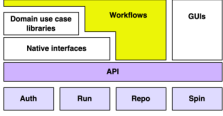


- Need traceability & model fidelity sufficient to **trust** digital test in lieu of physical
- Value in **ML** trained from and used to train simulations, must include model validation, UQ, and decision provenance.

"The workflow is the app."

Software Architecture Insights

Architectural "API View"
Workflow as a sequence of system utterances, organized into **4 subsystems**, each with a small set of system verbs embodied in an **API**



Auth: authenticate & authorize use within tenancies
Run: async job exec & chaining (data & control flow)
Repo: put / get / find data & metadata
Spin: (de)provision resources

Interfaces

- Flexible IDEs, Python, & Web interfaces
- Allows for domain-specific utilities, libraries
- Supports diverse user communities & integrations

Workflow Types

Type 1: Intra-Job
Example:
Iterative optimization of a design space exploration and visualization using in situ analysis and steerage of high fidelity CFD

Type 2: Intra-Site
Example:
Workflow composed of high fidelity, GPU-accelerated CFD visualized using in situ techniques and used to inform the training of a NN-informed design space exploration of cooling hole placement

Type 3: Inter-Site
Example:
Workflow that enables democratization of inter-agency resources - see: MxN app readiness (2021-2023 NERSC grant)

4

The core principles of abstraction, atomization, and loose coupling are a necessary response to the Cambrian explosion of novel capabilities that are emerging and are set to continue to emerge in the marketplace. We would be remiss if we did not acknowledge that we are building on the shoulders of giants here, enjoying a privileged position in understanding the wave of hardware acceleration and the incredible breadth of novel, software-addressable data, network and edge analytic systems thanks in no small part to our relationships with the US DOE-led Exascale Computing Project and the decades of experience our product engineering partners have in understanding the fundamental mathematics that underpin the design methods and simulations used to design our products. Similarly, other industries, most certainly including the automotive industry, have led the way in model-based, virtual validation of the performance of complex combustion systems and thought leaders in the US DOD and DOE continue to provide crucial insights into the metadata required to inform uncertainty quantification and decision provenance.

As Dr. Messer of Oak Ridge said in an interview recently, in supercomputing today (quote) "there is no one 'killer app'." (unquote) Increasingly in our experience we see that our computing needs are not app-centric and that, in a sense, "the workflow is the app". Almost never does an HPC job get run without some pre- and/or post-processing, and these are the most simplistic of the potential emerging use cases which grow to include examples involving coupled CFD simulation and machine learning, as we will present here.

In order to provide a digital thread which helps build trust in downstream dependent decisions and realize the business productivity enablement which comes from streamlining critical workflows, we've developed a workflow system which is built on four pillars – namely **Auth**, **Run**, **Repo**, and **Spin** – each defined by a small set of essential functional verbs. These subsystem names are hopefully self-descriptive, though we will explain them further in a moment, as are the obvious parallels with other tools of similar or even exact same names – a major aspect of this paper is to assert that these commonalities which suggest a rollout or a refactoring of various independent workflow approaches is now coming into view.

In our system, the functions of these four logical subsystems are exposed for use via a secure RESTful API layer, upon which native interfaces are layered, most popularly in Python, and additional GUI interfaces for example for the Web.

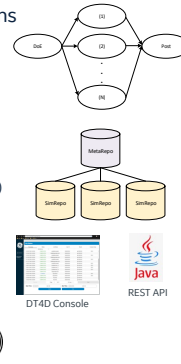
Continuing the refactoring, from our experience, we have come to see workflows in a heterogeneous system built upon these pillars as describable in three categories, or types. The first type of workflow are **intra-job** workflows, those running within a specific job, what we might think of as a typical HPC runtime allocation of nodes and other resources, which themselves can be further suballocated and orchestrated among one or more collaborating applications. An example of this type of workflow might be an in-situ controller which takes samples from a running HPC simulation and feeds them to a concurrently running machine learning training process. The second type of workflows are **inter-job** or in other words **intra-site** workflows, within an enterprise, in which sequences of jobs are executed in one or more potentially interdependent workflows. A typical HPC scheduler is an example of an intra-site workflow controller. Finally, the third type of workflows are **inter-site**, between enterprises and across security perimeters, with each site wishing to collaborate in an inter-site workflow providing the four functional pillars of Auth, Run, Repo, and Spin.

The refactoring which we present here is likely to continue and be iterative, even further informed by the similar-to and related efforts of others, and we'll talk today at the end about what we think comes next.

What is Digital Thread 4 Design (DT4D)?

A framework for macro-workflow programming across distributed heterogeneous computes

- **Common job model** unifies diverse compute platforms & runtime implementations
 - **Run** HPC / CPU / GPU hosts, Windows or Unix, on-prem or cloud, BYO scheduler or use DT4D agents
 - **Tool-chaining** within a job, asynchronous **job-chaining** across heterogeneous computes
 - e.g., high fidelity CFD data used to inform ML models trained to perturbate low fidelity CFD DoEs
- **Managed data** across all job chains using **data repos** that respect **tenancy**
 - **Repo** Sim(Tool)Repo / MetaRepo stores objects decorated with system & user custom metadata
 - **Data tenancy** organized by any user and use grouping, private metadata namespaces
 - Digital thread logged in **RunRepo** with **provenancial search** – relates the data to the job chain(s)
- **Multiple interfaces** to compute & data, to model & monitor jobs
 - Visual workflow IDEs, **Py4DT4D** SDK, **REST API**, DT4D Console
- **Security**
 - User authentication & tenant membership; secure tokens for APIs LDAP/OAUTH
 - Comprehensive data & compute security – **data & nodes scoped to a tenant**
 - Designed to comply with US ITAR, CUI, and FedRamp requirements
- **Dynamic provisioning** of cloud computing resources



We'll now give a brief overview now of the implementation of this software architecture for heterogeneous workflows at GE. Digital Thread for Design, or DT4D for short, is a framework for macro-workflow programming across heterogeneous computes including HPC, dedicated ML hardware, as well as commodity workstations and enterprise VMs. It is characterized by the four subsystem pillars and their layered interfaces. The **Run** subsystem interfaces with our HPC scheduler and distributed “runner” agents deployed on non-HPC singleton nodes providing job start, stop, and status query. A Run subsystem component tracks job status events and fires other registered handler jobs when so-triggered, either by a job status change event, or a data change event in the Repo subsystem, providing for both control flow and data flow workflow modeling. All events and their control and data associations are logged for later interrogation as a digital thread.

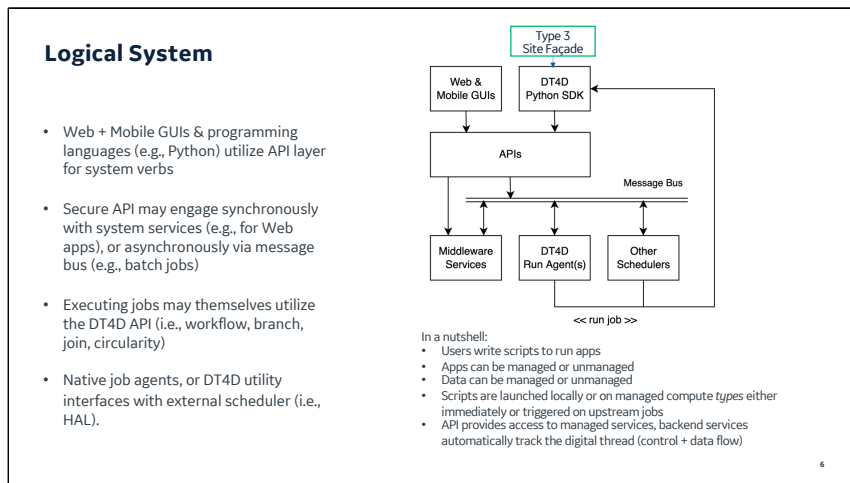
The **Repo** subsystem provides data and metadata storage separated by user tenancy and permits general put, get, and find operations. Data can be stored in any managed addressable location or locations, which permits performant fetch based on locality. The storage medium itself, which typically for GE is an S3 object store, might just as easily be the URL of a live data feed, or a bare filesystem or HDFS reference. Each data entity under management, including tools, is fronted by a descriptive metadata sheet, and together this so-called MetaRepo constitutes a card catalog index of the entities – with both system-standard and user-

customizable metadata tags available.

The DT4D Auth subsystem is built upon the internal GE security enterprise SSO system, providing functions such as login, tenancy identification, and the return of a secure token which can subsequently be used to make authorized API calls.

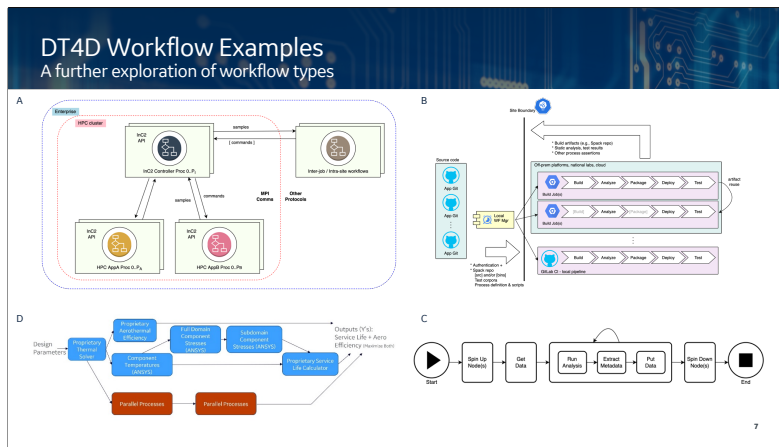
Upon the API we have layered a native Python interface which also provides a local runtime, allowing user-local jobs to be first class citizens in the workflow ecosystem, their actions part of the digital thread. Web interfaces exist for querying job status, digital thread and Repo interrogation, and the rendering of application and workflow-specific, declarative user interfaces, for example, those permitting real-time workflow steerage.

The Spin subsystem – for the instantiation and freeing of ephemeral computing resources largely in what we think of as “the cloud” – is from an implementation point of view a topic for future work, but it starts with a fair amount of real-world experience using today’s service providers such as AWS and Azure and others currently being utilized by GE, and there is general line of sight to an abstract Spin implementation which generally wraps existing cloud vendor and aggregator APIs. Spin is the only optional subsystem in the model, but like other subsystems, Spin at its essence embodies a small set of actions – list the available a la carte and bundled compute options, spin them up, and spin them down. Presentation of the cafeteria of available compute types from which to spin will be implemented by the specific service provider – anything from singleton commodity grade VMs to entire turnkey high-performance clusters are possible, potentially tailored by tenancy, with pricing metadata. The creation and management of resources, now included within the scope of Software Defined Everything, is now part of the workflow itself.



Logically, the DT4D system is organized into components communicating via an enterprise message bus, which is an intra-site workflow construct. Various resource schedulers connected to the bus receive asynchronous tenant-scoped requests to run jobs and emit back status. These events are interrogated for the purpose of triggering downstream workflow legs and are logged for audit.

In contrast, inter-site communication for enabling what we're calling type 3 workflows would suggest more portable transports suitable for wide-area networks, such as HTTP. Here at the top of the diagram we show the inter-site workflow interface as a thin façade on a native system interface, in this case that of DT4D, which contains characteristics of the four architectural subsystems. An inter-site workflow controller contains much of the same middleware components as the similar intra-site controller system – including a component to track event handling, one for audit logging, various user interfaces, and a tenant-scoped metadata repository – simply extended to include a multi-site point of view.



Let's look at a few examples of the three types of workflows as we've categorized them. (All all real but C...)

Clockwise from top left, we show a combination type 1 and type 2 workflow – in other words, an in-situ workflow with hooks to wider enterprise workflow services. Here, an in-situ workflow is being executed as a collection of three co-launched applications – two physics-based simulations, and an in-situ controller written in Python which observes interim simulation results. When needed by the workflow, the in-situ controller emits an event to our intra-site message bus, via the API, containing the definition of a declarative GUI, which is subsequently observed by the GUI user, populated, and on submit a message providing simulation steerage is sent back to the running in-situ workflow.

Next, in the top right we show an inter-site workflow providing MxN application readiness, i.e., multi-platform, multi-app CI/CD. Each of the N sites exposes a type 3 workflow API, as shown on the prior slide, implementing the four workflow pillars of Auth, Run, Repo, and optionally Spin, permitting secure transfer of source code and test suites to the remote site, and the execution of a site-specific intra-site type 2 workflow to build and test each of M user apps. Such a multi-platform CI/CD seems a requirement of efficient usage of site allocations, avoiding big-bang type deployments of new code. GE is no stranger to complex codes nor HPC, but we're not embarrassed to say that building applications on the growing set of platforms we use is a continuing challenge. HPC build tools like Spack are helping to manage complexity and lower barriers to entry – by extension, a reusable multi-site CI/CD pipeline which includes quality assurance and human interfaces has potential to be a game-changing factor in democratizing use of shared resources.

In the bottom right, the third workflow based on Spin could be implemented either internally as an intra-site workflow, assuming the site supported such cloud-like instantiations, or by an individual perhaps as part of a zero-trust security architecture as an inter-site type 3 workflow. Here compute resources in the cloud, selected from a set of presented options, are allocated and freed as part of the workflow itself, giving the user total control over their own cost function.

Finally, we show an intra-site workflow implemented by our engineering research team which uses HPC-based CFD to produce data for a surrogate model being trained and assessed on ML-specific hardware, illustrating what in our experience is the increasingly common heterogeneous nature of practical industrial use cases. Since this is a talk on our experience with software, we'll dwell on that aspect of the exemplar. In this case, the HPC and ML nodes are behind distinct schedulers, but that need not be the case. As mentioned, the refactoring presented in this paper is informed by prior art, for example the NERSC Superfacility API, which provides a job and data movement interface to Perlmutter, Cori, and other systems. With a single Auth system, in this case secure tokens issued from NERSC's Iris system, we can ask a question – would we consider this collection of HPC systems many sites with a shared Auth, or a single site with many individual compute types within it – Perlmutter, Cori, and the like. Furthermore, on a given cluster like Perlmutter we now see both GPU-accelerated and non-accelerated nodes – a single platform has multiple compute types within it. This is also the case in GE's DT4D compute nodes which come in many flavors from both hardware and pre-prepared application configurations. The software architecture is flexible enough to adapt to the relative viewpoint of the physical enterprise.

Final Thoughts



Next Steps & Areas for Collaboration

- Further refactoring of the workflow types & subsystems model; unification of tooling
 - Messaging
 - Resource Types
 - Standard APIs
 - Zero trust security
 - Metadata using FAIR principles
- Plug-and-play MxN CI/CD
- Multi-order modeling, model quality assess & notate
- Workflow visualization & navigation
- Spin

Democratization

Acknowledgements



We'll conclude today with some topics for potential future work and collaboration.

- For starters, we see further refactoring of the workflow model as presented, with convergence in constructs between the three types of workflows. This might include inter-process messaging formats, metadata definitions of intra-site and exposed inter-site computing resources, the standardizing of APIs, open and zero-trust security, and further refinement of the metadata model along FAIR principles.
- Second, a plug-and-play MxN CI/CD seems necessary to provide democratizing application readiness across a wide range of compute platforms. The CI/CD workflows themselves are enabled by mature and unified programmatic constructs across all three workflow types, further lowering barriers to entry in the creation of applications and workflows which take advantage of a growing number of shared computing resources.
- More effort is needed to fully exploiting the benefits of multi-order modeling, in which multiple simulations, solvers, and models of different fidelity and therefore cost are selected and orchestrated singularly or in ensembles, for example using an in-situ optimization workflow. Models under Repo management, being effectively just data, can be decorated with metadata such as its region of domain applicability and competency, any bounds on confidence, and other operational parameters. These entities can be continuously

decorated with this metadata representing their cumulative performance in real use cases over time, validated perhaps by real and perhaps live sensor field data, with goals including the increasing of trust in virtual test, and formalisms in uncertainty quantification. At GE, some of these concepts are being applied in training models in what has been loosely termed a more “humble AI”.

- Another topic, improvements in workflow visualizations and navigations – of the past, present, and alterable future of the workflow, including its data associations and other provenancial information, with rollup metrics and summaries for executive consumption. In general, more work is needed in understanding the user experience to better enable the broadest adoption. For example, more leveraging of in-situ visualization techniques, and better ways to express the intent or phenomenon of interest. A user interface which observed the user’s actions and outcomes, as also represented in the resulting digital thread, might be positioned to learn best practices, to recognize for example events of interest in the simulation and automatically take appropriate action.
- Finally, more work is needed on the Spin subsystem, and its implications for tenancy, data classification, zero-trust security, and Software Defined Everything, including the enablement of independent verification and audit.

And, we will end here by mentioning just some of the organizations and software R&D teams who influenced this work in various ways and to whom we are grateful and to whom we look forward to talking and engaging with in a continuing effort to further develop these ideas: their realization, dissemination, and potentially their democratizing adoption.

Thank you.

