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

Andrew Gallo[0000-0002-2711-7589], Ian Claydon[1][0000-0002-9010-3585], Eric Tucker[1][0000-0001-5244-0997] , Richard Arthur[1][0000-0003-2960-9891]

[1] General Electric – Global Research Center Niskayuna NY 12309
Andrew.Gallo@GE.com

Abstract. We will present our industrial experience deploying software and heterogeneous hardware platforms to support end-to-end workflows in the power systems design engineering space. Such workflows include classical physics-based High Performance Computing (HPC) simulations, GPU-based ML training and validation, as well as pre- and post-processing on commodity CPU systems.

The software architecture is characterized by message-oriented middleware which normalizes distributed, heterogenous compute assets into a single ecosystem. Services provide enterprise authentication, data management, version tracking, digital provenance, and asynchronous event triggering, fronted by a secure API, Python SDK, and monitoring GUIs. With the tooling, various classes of workflows from simple, unitary through complex multi-modal workflows are enabled. The software development process was informed by and uses several national laboratory software packages whose impact and opportunities will also be discussed.

Utilizing this architecture, automated workflow processes focused on complex and industrially relevant applications have been developed. These leverage the asynchronous triggering and job distribution capabilities of the architecture to greatly improve design capabilities. The physics-based workflows involve simple Python-based pre-processing, proprietary Linux-based physics solvers, and multiple distinct HPC steps each of which required unique inputs and provided distinct outputs. Post-processing via proprietary Fortran and Python scripts are used to generate training data for machine learning algorithms. Physics model results were then provided to machine learning (ML) algorithms on GPU compute nodes to optimize the machine learning models based on design criteria. Finally, the ML optimized results were validated by running the identified designs through the physics-based workflow.

Keywords: heterogeneous computing, HPC, workflow, metadata, data provenance, digital thread.

1 Background

The complex design of modern systems creates challenges ranging from selecting from a myriad of trade-offs to guiding product design, manufacturability, and sustainment. Improving the performance and reliability of modern gas turbine power generation systems, for example, requires identifying actionable insights to extending asset life and operating cycles and/or improving overall aerothermal efficiency. The dimensions and applications of turbines and their components vary during the design, fulfillment, sustainment, and operational optimization phases of their product lifecycle. Design decisions must account for differences in scale, fidelity, and robustness across domains of competence, a process further confounded by the need to understand and trade-off between the multi-variate factors contributing to turbine performance. In addition to delivering the loosely-coupled and workflow-enabled functionality required to manage and resolve differences such as design contradictions, the underlying design system must enable the development and continual improvement of model-based design confidence and collaborative data-driven decision-making. The capability to capture and maintain searchable user and system-definable metadata tags, integrated with industry standard identity and access management capabilities will enable the development of digitally verifiable decision provenance [1].

We intend to demonstrate how our experience developing and deploying multiple generations of Gas Turbine products and the heterogeneous compute, modeling [2] and test platforms [3] required to design, field and sustain the same has informed the architecture, design and deployment of our next generation of digital thread capabilities[1, 4]. We show how our tooling, and that of the national labs, aligns with this model and demonstrate how our current capabilities are being used to develop, generate, orchestrate, and deliver an ensemble of design tools in service of real-world engineering design and product fulfillment challenges.

2 Advances & Outlook

We assert that four novel capabilities were required, at the advent of our work, in order advance the state of practice.

1. The ability to loosely-couple increasingly heterogenous data, tooling and compute in a modern, API-oriented manner and deliver actionable insights at all stages of the digital thread [5].
2. The ability to capture, characterize and describe data, tooling, job, and authorization information in machine-ready repositories, express their content via machine-readable metadata tags and describe their capabilities as machine-actionable verbs.
3. The interfaces and translational services required to enable the integration of machine learning capabilities into design space exploration, design space mapping, rapid candidate screening, field fault analysis, etc.

4. The establishment of sufficient modeling and simulation fidelity [6] and digital trust systems to enable human decision-makers to trust digital design systems and practices.

Having developed these capabilities, in some cases apace and in partnership with other industry, academic and government partners, we are now beginning to introduce them into practice in our next generation product design, fulfillment and sustainment practices. While the specific example articulated in this paper is partially instructive of the changes in practice underway, we believe that we are still at the very beginning of enabling the transformation in design practice first envisioned at the advent of our program.

3 Workflow Enabled Model Based Engineering

3.1 Requirements and Analysis

Identifying, characterizing, understanding, and generating actionable insights associated with the engineering design challenges noted above is best enabled through a co-design enabled approach [7] that brings together expertise in computational hardware and systems, relevant engineering domains and mathematical methods and software engineering.

Enabling the current and next generation of gas power turbines necessitates computation hardware and systems that are increasingly heterogenous. While some models benefit from access to heterogenous, hyper-converged systems, others may be better suited to loose-coupling and execution on a hyperscale cloud. At the other end of the computational spectrum, some models may run adequately on desktop-class systems, or specialized hardware embedded at or near a test stand or physical product installation. For the purposes of the example laid out in this paper, the computing environment may include a variety of architectures, including workstations and workstation-scale virtual machines, hyperscale cloud instances, and HPC servers – any of which may include accelerators such as GPUs. This provisioning of differently-abled nodes for different application purposes is typical in modern computing centers, as pioneered and advanced by the Leadership Computing Facilities at the national labs [8].

The wide array of engineering domains required to realize these actionable insights requires an equally wide array of engineering applications, each with its own performance characteristics, domain languages and interfaces. Combined with the diversity of scheduling (batch, interactive) and pre- and post-processing capabilities commonly available to engineering practitioners, further impetus is lent towards a coherent system able to provide asynchronous and loose coupling for workflows and with functionally ready integration mechanisms for interoperability and translation between domain languages and system interfaces. The loose coupling approach applies at various scales, including, as stated, between applications collaborating in an orchestrated workflow, as well as between elements of the same application, for example, in multi-physics co-simulation. In so much as a single application is not likely to elegantly encapsulate the end-to-end engineering process, it follows that figuratively “the workflow is the app.”

Therefore, loose coupling necessarily becomes the dominant paradigm for heterogeneous applications. Loose coupling implies contractual interfaces between collaborating components, and thus permits a wide range of implementations – in applications, in workflow constructions and executors, schedulers, and better adapting to growing diversity in the underlying hardware architectures [9]. Asynchronous loose coupling of components is often achieved with message-oriented architectures and event handlers [10].

A core tenet of co-design is pragmatism in implementation. To advance capability and performance, applications must leverage programming abstractions, composition frameworks, and numerical methods libraries which in turn conform to algorithms and data structures that efficiently exploit the state of the art in (hardware) system architecture. Advances in domain applications should proceed following technical progress in the underlying software and hardware infrastructure. The application workflow system must therefore facilitate agile adoption of improved implementations at any level of the ecosystem and communicate novel and enhanced capabilities to the users of the system.

Data produced and consumed by the workflows varies by type, size[1], and importantly by security classifications and control requirements; requirements both internal to an organization as well as established through legal and regulatory compliance. Establishing a data tenancy, or a grouping of similar authenticated users, human or otherwise, along with their access to the data required for the relevant design process/working, requires the collection, validation, storage, and processing of a complex array of metadata. A multi-tenant system must enable and allow for the establishment of logically, and sometimes physically segregated data tenancies and must contain systems and practices for storing and strictly protecting the auditable metadata required for each tenancy. In an ideal situation, every data tenancy, indeed every data element within the tenancy, is accompanied by several critical metadata elements required to enable decision provenance: *who* did or is doing *what*, *when* and *where* is it happening, and *why* [11]. Said differently, which individual or system is running each job, in what role, in each environment, and for what business purpose.

Data proliferates, and data discovery is a well-known problem [12]. To support FAIR data objectives across the heterogeneous computing ecosystem, the data tenancy metadata described previously must add descriptive metadata that includes evidence of the runtime ("the where") which produced or consumed it. When the execution is extended over a distributed multi-step workflow, it is necessary to notate the chain of jobs and their data I/O dependencies for reporting of interim results as well as end-to-end decision provenance.

Up to this point we have defined, discussed, and described solutions for use cases involving two types of workflows – one type operating intra-job (for example in-situ multi-physics workflows), and another type which chains individual jobs together into

[1] Size matters – data size and locality must be addressed during workflow implementation and execution. Stated simply, the data must move to the compute, or the compute must be available where the data resides. A workflow system which provides abstractions over data location and site-specific application execution would be useful, as described in this paper, although we will not address the topic of "big data" specifically.

a broader workflow.[2] This second type is still intra-site, and includes the conduct of most HPC schedulers, which effectively run another job when some upstream job completes. While intra-job workflows are always executed on behalf of a single user identity, intra-site workflows may be for the same or different users within the same enterprise. An example is a GE workflow that performs physics-based computations on an HPC system and when complete triggers a second job running on dedicated GPU nodes to perform a ML training, launched under a different scheduler for the same or even a different GE user.[3] The examples of different users being orchestrated within the same workflow include inter-departmental handoffs in an end-to-end design process.

Finally, there is an emergent third type of workflow, inter-site, in which the executing jobs span computing sites on behalf of the same user, albeit perhaps with distinct instances of their identity in each location.[4] An example of this third type is a single researcher workflow which includes jobs running sequentially on both GE nodes and under grant on national laboratory machines[5], the latter via the Superfacility API (SFAPI) [14]. A framework for rapid development of such cross-site workflows would provide encapsulated abstractions for each site's distinct authentication scheme and job running syntax, as well as translation of job status codes into some common interoperable set, and we discuss this below.

A quick scan of the online documentation for SFAPI shows RESTful endpoints in categories – "system health", "accounting information", "compute" including submit, read, and cancel job, "storage" and "utilities" providing upload and download of data, and a separate endpoint to provide the second leg of a multi-factor authentication process. Job status messages returned by these API endpoints map to those of Slurm and fall into several general categories – initializing or pending, running in some form, terminated normally or abnormally, and various informational messages [15].

DT4D or Digital Thread for Design, a system for orchestration of intra-site and inter-site workflows at GE, and discussed in more detail below, exposes its own API, with

---

[2] This is like other categorizations of workflow types. [13]

[3] In our compute model, as we will see below, this can be implemented as one site or two – as a single site with two distinct runtime "compute types" within it sharing the same enterprise authentication scheme, or as two distinct sites which happen to have the same authentication scheme in common.

[4] On one site the user's identity might be "sbrown", on another "Susan.Brown", etc.

[5] While the national laboratory computing facilities are valuable resources, for industrial applications their utility is practically limited to non-proprietary workloads. A further impediment to their integrated use is the necessary national laboratory security perimeter, and while the Superfacility API alleviates some of those impediments, the SFAPI is not yet deployed across all national facilities. Workflows which span sites including commercial sites like GE's have their own issues with security perimeters. As we will show later in this paper, the design of an inter-site workflow system needs to be realistic about the lack of ease of bi-directional data flow – it is not a trivial matter to open a communication channel into a corporate network, and a system which does not depend on such access likely has a higher probability of adoption.

similar token validation, and endpoints which fall into several of the same general categories – run and inspect job status, transfer data.[6] Granular job profile, status and results, along with associated data tenancy metadata are captured and communicated within the system, allowing for either human or machine learning guided refactoring and optimization of the design process.

3.2 Conceptual Design of the Workflow Framework

A system for heterogeneous workflows – one for example not governed by a single scheduler or designed for single tenancy – which also meets the requirements can be conceptually thought of as having four pillars. Each pillar represents a category of contractual interfaces which must be implemented by a given collaborating computing site provider. These same pillars apply to inter-site and intra-site workflow types – a given computing site must implement these functional areas for its own purposes, and if it intends to be collaborative with inter-site workflows, it must also expose these functions.[7]

- Auth: provide for user authentication and authorization, e.g., link to a data tenancy
- Run: execute a given job or jobs in the user, workflow or otherwise determined runtime, with relevant translations across interfaces and job orchestration systems
- Repo: a means to put to and get data and its metadata from managed store(s)
- Spin: provision computing resources, e.g., in the cloud[8]

Auth takes many forms – some examples include a priori identity token dispensing to be used for just-in-time session token generation, as well as interactive logins. Run implementations are similarly diverse, but its functions fall into some general verb categories – run job, check job status, cancel job – the set of actions is notably terse. To this we add event handling for job chaining – set / unset event handler, list active handlers. Successfully implementing Repo functionality requires support for all common file system types (block, blog, object, etc.) but also necessitates interactivity with the relevant metadata. That metadata must itself be stored and made available via an enterprise metadata catalog and associated APIs. Those APIs themselves express common set of verbs: “put”, “find”, “get”, etc. The industry has made broad advancements in Spin implementation and while the services and verbs available in these environments are broadly similar, they are often expressed within proprietary frameworks that limit portability.

The heterogeneity of computing facilities now also necessitates the Run subsystem exposing a means for declaration of the compute type on which the job needs to run. This could mean, on an HPC system for example, to target the run specifically at available GPU-accelerated nodes. In a commodity node farm, this could mean targeting at a

[6] SFAPI has a richer administrative API.

[7] Intra-job i.e., in-situ workflows, do not require all these pillars – e.g., their authorization to run is validated before running, their computing pre-provisioned, etc.

[8] The GE Spin component is not yet implemented. Conceptually it utilizes existing vendor APIs such as AWS, VMWare, etc.

named node type which has been pre-prepared with certain commercial engineering software.[9] Heterogeneous intra-site workflows under the same enterprise identity require a runtime model which incorporates and reconciles multiple schedulers, for example, by normalizing their job status return codes. Inter-site workflows require the same.

The concept of "job" includes not just what are traditionally thought of as tasks run by the scheduler, but also, potentially, the Repo and Spin functions as well. With event handling this permits end-to-end workflows which minimize the cost function.

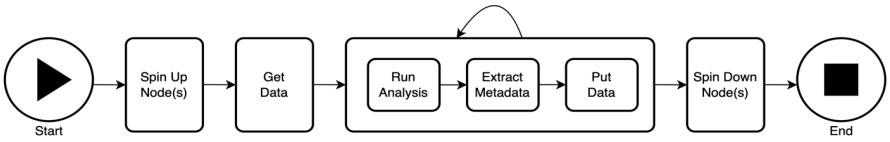

Fig. 1. multi-job workflow with dynamic node provisioning

Upon this logical 4-part API layer, we can stack native language interfaces such as in Python, GUIs for human interaction for visibility into the running workflow and interrogation of the resulting digital thread. Under each logical section of the API, multiple implementations are possible, for example, AWS, MS Azure, and Google Cloud APIs fit under the Spin façade.

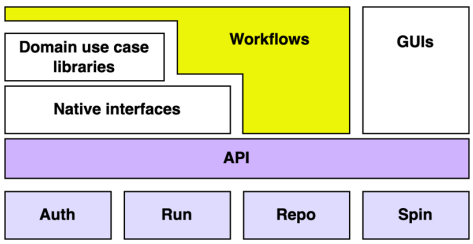

Fig. 2. DT4D API stack

3.3 Workflow Framework Implementation

The GE internal "Digital Thread for Design" (DT4D) DT4D framework has fully implemented tooling – middleware, APIs, and native interfaces – for each of the workflow types described above.

[9] While the use of containers is ideal for this purpose, and is implementable within DT4D, practical limitations in system security requirements, commercial software license terms, etc. necessitate the ability to sometimes understand and interface with statically configured systems and runtimes.

Intra-job workflows are launched on a given HPC system in a Multiple Programs, Multiple Data (MPMD)[16] modality, with our "inC2" Python / C++ library providing a simplifying messaging abstraction for extracting interim simulation results, computing derived values, using these to steer the simulation, and as desired transporting informational messages to other enterprise systems, as further described below.[10]

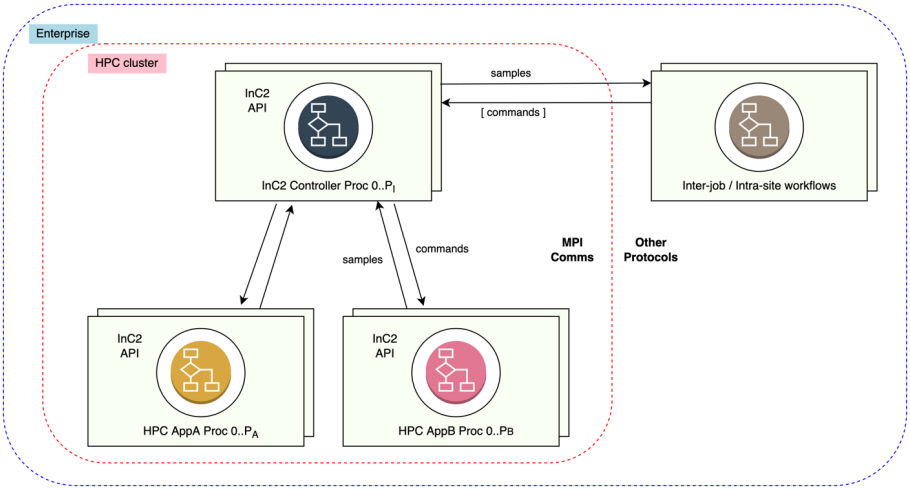

Fig. 3. In-situ intra-job workflows communicating with inter-job / intra-site enterprise workflows and interactive controllers.

GE enterprise intra-site multi-job workflows are enabled by the DT4D system. This system implements the key elements of the conceptual Auth, Run, and Repo subsystems.

- Auth functionality is accomplished through lightweight integration with an existing, token-based GE internal OAuth service.
- Run uses a COTS message bus to dispatch jobs to several schedulers, and job status is similarly transported on the bus. For un-clustered singleton nodes, a "Runner" agent is deployed to the node as a bus listener and assigned at runtime to a specific tenancy. All job launch information is recorded in a "Run Repo" – which tool, version, arguments, etc., as well as the full status sequence – to permit reproduction.
- Repo implementation uses a central metadata index to front several object stores containing both data and tools which can be loaded into the job context and tracked at runtime; thus, the digital thread includes both the historical control and data flows.

[10] An examination of prior art included ADIOS2 [17] which also provides an MPI-based transport but also implements far more functionality than our use cases demanded, and thus we erred on the side of simplicity.

The logical subsystems are exposed by a RESTful API and a native Python interface which provides the runtime harness to implement local jobs which are authenticated as first-class citizens of the ecosystems. The API is also used by a GUI which can be Web or mobile and provides a job monitoring interface,[11] system uptime information, and a metadata-driven directory tree view of data under Repo subsystem management. [12]

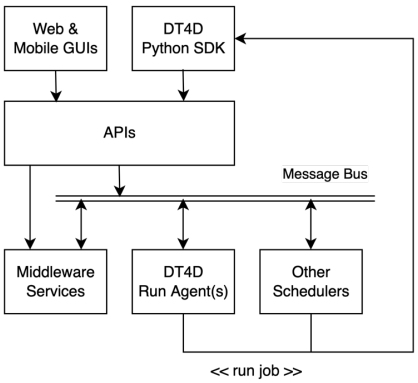

Fig. 4. DT4D high-level deployment diagram

Inter-site workflows are enabled by the "Local Workflow Manager" (lwfm) tool [18]. This thin Python façade provides native methods for the major verbs in each of the Auth, Run, and Repo subsystems. A collaborating Site implementation – be it based on the Superfacility API or GE's DT4D, or some other – is accomplished through a light-weight interface available within lwfm intended to capture 'signatures' of relevant system details. For example, the "NERSC" Site driver would translate the lwfm signatures into SFAPI calls, returning job status in lwfm canonical form. Like DT4D, the lwfm provides a local job status monitor service to handle workflow job chaining.[13,14]

---

[11] A current side-project involves the inC2 library permitting a simulation application to transmit an informational job status message which contains a declarative GUI definition – e.g., a set of simple form control parameters. The DT4D GUI can then render this application-specific form. Use cases include interactive simulation steerage.

[12] Future work includes defining a formal interface and resolving conflicts across collaborating Auth subsystems, for example, in a situation where collaborators from multiple labs necessitates identity federation. Future run optimization can be accomplished using Run Repo and hardware profile metadata. The implementation of the Spin component is also planned.

[13] Future work includes adding a cross-site GUI which conceptually subsumes that of DT4D.

[14] Examination of prior art included ALCF Balsam [19]. We also looked at LLNL's Flux [20], and at RADICAL-Pilot [21], which seemed directed at two of the three workflow types we describe. LANL's BEE addresses reproducibility by using application containers and extending the runtime model with pre-execution CI/CD [22]. Pegasus is centered on an API for describing *a priori* DAGs. [23] Many other projects approach the workflow space from different non-comprehensive points of view, and each has its own potential strengths and weaknesses [24]. The build vs. buy decision is multi-faceted.

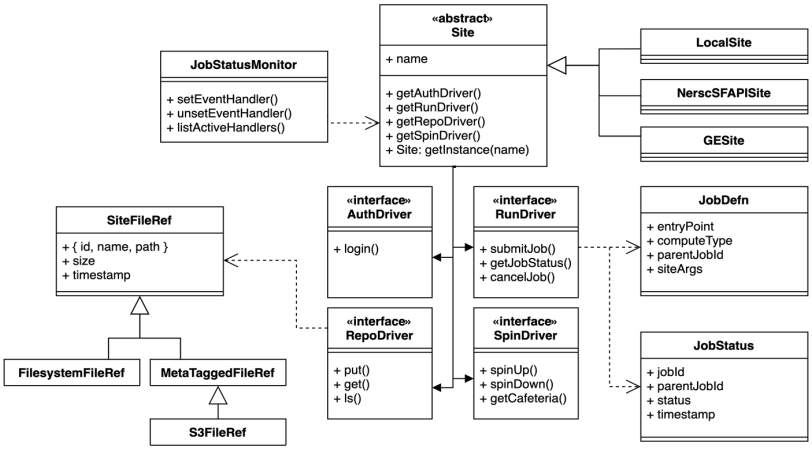

Fig. 5. UML model for lwfm software, showing the Site interface and its four sub-interfaces

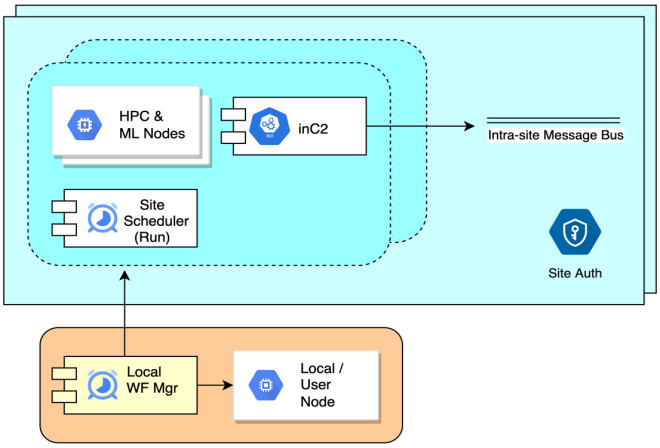

Fig. 6. lwfm orchestrating inter-site workflows, each consisting potentially of their own intra-site and intra-job workflows.

In the tooling which supports each of the three types of workflows we address – intra-job (“inC2”), intra-site (“DT4D”), and inter-site (“lwfm”) – we elected to represent the workflow in terms of actual code, specifically Python for its ease of use and wide adoption and provided libraries for interacting with the workflow constructs and APIs exposed for the type or combined types of workflows which are applied for each engineering business use case. There are many examples in the literature of workflow systems which utilize visual or marked-up representations of the workflow as a directed graph. Visual representations can be useful for some users and for documentation but

for other advanced users coded representations are often preferred.[15] An additional problem in visualizing arbitrary workflows arises from its dynamic nature – the workflow can alter its future self, and thus the static and dynamic representations of the workflow differ, the latter being only fully realized when the workflow is complete, and is often more challenging to capture in-flight being subject to the compute site security perimeter constraints as described.[16]

3.4 Workflow Framework Applied: MxN Application Readiness

Complex design processes often necessitate workflows that span multiple sites and data tenancies. I.e., a researcher with access to both a company-managed supercomputer as well as one or more grant allocation on national laboratory or academic systems. When contemplating novel workloads, it would be normal to make a pre-validating run of an application on local hardware (Stage), debug it, then move the run to a larger machine for the actual simulation (Production). Many applications must be recompiled,[17] and revalidated, on the target machine architecture before actual use for both basic functionality and for performance optimization. This process may need to be repeated as application code and/or underlying system operating system, kernel, library etc. changes are made. As the number of collaborating sites increases, the problem becomes burdensome. As the number of collaborating applications increases, the problem becomes intractable. Thus, assistive, and potentially democratizing tooling is required – the M application by N platform readiness problem is a use case for inter-site workflows and can be implemented with lwfm and the pertinent site drivers.[18]

---

[15] It's possible of course to generate a graph representation of a Python script from its abstract syntax tree – a tactic we showed to users to a somewhat unimpressive response – the raw code was the preferred medium.

[16] The visualization and navigation of dynamic and graph-oriented workflow traces, including with references to the data produced and consumed at each step, was demonstrated in the early phases of this project, but it is a subject of near-term future work to better understand how to apply such visualizations to real use cases.

[17] The team has utilized Spack [25] and CMake in the build system. GitLab CI is used for internal CI/CD.

[18] NERSC 2021 Director Reserve Allocation, project m3952, "MxN Application Readiness", A. Gallo PI.

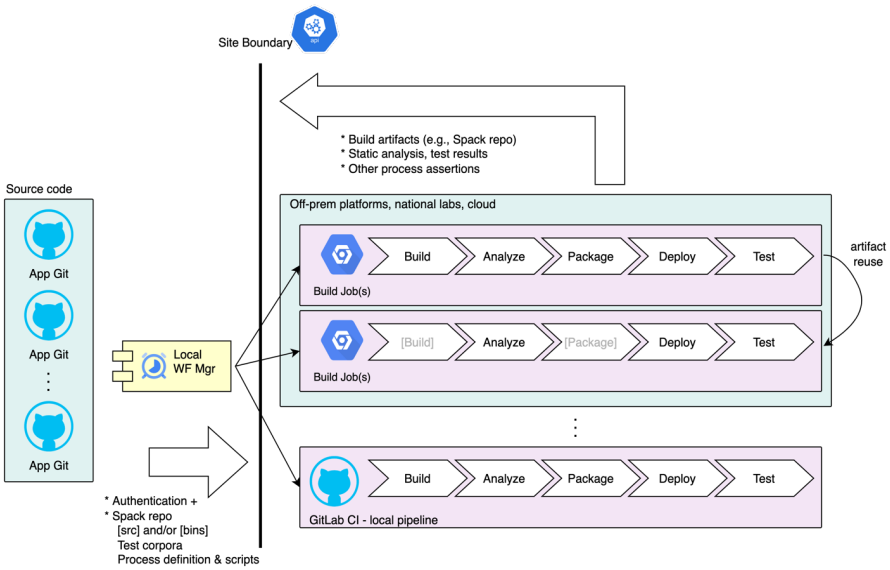

Fig. 7. MxN CI/CD – lwfm orchestrates inter-site workflows to maintain readiness of M applications on N target platforms.

4 Workflow Framework Applied: Reducing Low Cycle Fatigue

4.1 Requirements & Implementation

The flexible architecture of DT4D provides engineers the tools to orchestrate and standardize workflows, easily "get" and "put" data to storage Repos and create surrogate models of complicated components. A problem of industrial interest is that of low cycle fatigue (LCF) failure of gas turbine blades stemming from the cyclical operation of the turbine. This weakening of the metal and continual stress-based deformation generates cracks and failures of the blade limiting the life of the blade impacting both the customer's ability to guarantee performance and the business' ability to market gas turbine products. Gas turbine airfoils, shown below, in their simplest form, can be cast metal airfoils with an array of cooling holes arranged on the surface which blanket the surface with cooling air increasing the life of the blade. The design of these airfoils is a delicate balance between two dominant factors, the LCF failure and aerothermal efficiency. These factors combine to impact business decisions ranging from duty cycle management and pricing to outage scheduling and upgrade planning. To maximize sales expediency and confidence in the ability to rapidly move from a customer's requirements to an optimized design is paramount.

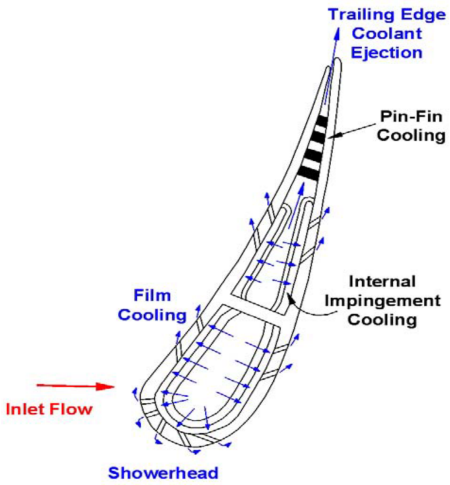

Fig. 8. Gas Turbine Airfoil Cooling Blade Schematic [26]

Optimization of a blade design requires many parameters with only one discussed here; that is the cooling layout and how it impacts the delicate balance between the mechanical life of the blade and the aerothermal efficiency of the overall system. For example, an engineer could improve the longevity of the blade by overcooling the blade surface, but this additional cooling air reduces the thermodynamic performance of the engine reducing the total amount of power the engine can produce [27]. Traditionally, optimization of the arrangement of the cooling parameters has been a manual process led by domain expertise, historical design practice, and field operating knowledge generally allowing two or three designs to be analyzed prior to delivering a finalized recommendation, solution, or product to the customer. True optimization of a design requires many more iterations and must encompasses a broader variety of operating conditions and degrees of freedom than is allowed by the current state of practice. In our implementation, the DT4D system enables this by creating uniform repeatable workflows that feed into neural network (NN) training algorithms. These uniform repeatable workflows contain proprietary and commercially available analysis tools and model many different permutations of the blade and the surrounding subsystem to train a NN. These trained networks then exhaustively permutate millions of design iterations in seconds rather than the hours or days it would take to run traditional physics-based workflows. Validation of resultant optimized designs is accomplished via DT4D allowing for further development of design confidence, including through recursive iteration of workflow-enabled model training. This process can reduce design times by approximately fifty percent while providing more thoroughly optimized and validated designs.

4.2 Process Workflow Described

An optimization workflow for the gas turbine blade is comprised of three sequential steps. The first is a physics-based workflow to generate the required training data for

the NN which has been previously reported in Tallman et al. [28]. The second step consists of training the NN, exhaustively exploring the designated space, and identifying the optimum results. The final step is validation of the optimum NN results which, simply put, is rerunning the physics workflow again with the design parameters identified by the NN. At their core the physics-based workflow and NN exploration algorithm are Python-based scripts that setup the requisite arguments for individual jobs including input and output lists to get from and put to the repositories along with the proper job triggering information. These workflows are designed to be repeatedly and routinely executed by a range of individual users over time.

4.3 Training Data Generation

In analyzing the gas turbine blade a physics-based workflow like that shown was utilized to improve the life of these critical gas turbine components.

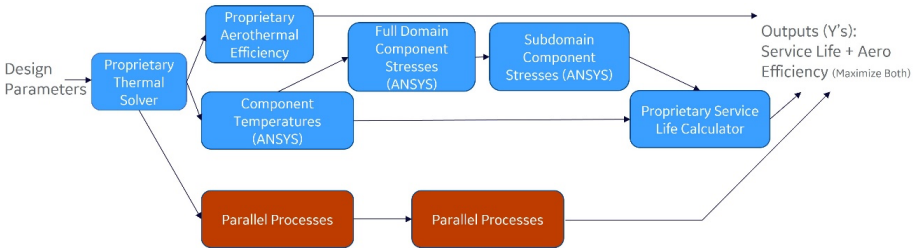

Fig. 9. Training data generation workflow

The workflow begins with the definition of the DOE design parameters which are separated into individual cases and subsequently fed into a proprietary thermal analysis tool. The outputs from this tool are then passed into three parallel processing paths including the top path which uses proprietary software to calculate the impact the design parameters have on the aerothermal efficiencies, the bottom path which simply represents any additional tools that the end user would desire to include in the design cycle, and the central path which is the focus of this discussion. All but the final element along this path involves the integration of commercial ANSYS tools into the workflow. This pathway involves calculating the component temperatures which are used to calculate full and critical sub domain component stresses. These temperatures and stresses are reunited in the final tool on this path, a proprietary service life calculator, to arrive at an array of life cycle values for the given design. The final step, alluded to in the figure, is the unification of the resulting file formats to a single format and the combination of all results into proper formats for training the NN.

4.4 Surrogate Model Creation and Execution

Leveraging open-source Python-based libraries, a NN was developed utilizing training data generated via the initial physics workflows described above. The trained NN was used to exhaustively search the area of interest for design parameters that provided optimum behavior. These design parameters were then pruned from the bulk results and formatted to be fed back through the physics workflow as a validation step utilizing the same Python code that generated the training data.

4.5 Validation and Analysis

The third workflow step is repeating the physics modeling workflow with the optimized conditions from the neural network. This necessary validation step provides multiple benefits. The first is that it identifies the true accuracy of the NN and determines the confidence in the results. If the NN is found to be accurate, it allows the engineering community to identify trends in the data that the original DOE space contains and identify trends which require expansion of the original DOE space to fully capture. Additionally, it provides training data to the NN for the next cycle of DOE points further improving accuracy of the models. If the initial NN does not possess the required accuracy, further training data can be generated using either the optimized design parameters from the NN or a newly generated set of DOE points similar to the initial training data.

5 Conclusion

Designing, fulfilling, and sustaining the next generation of gas turbine power generation systems, as well as continuing to improve the operating performance of the current generation of fielded products requires a model-based, digital thread orchestrated approach to developing product insights. The complexity of managing differences in time scale, fidelity, and robustness, combined with the desire to increase degrees of freedom and explorations of design options necessitates a design system which is capable of orchestrating workflows that span multiple heterogenous systems and data tenancies. It must also capture and communicate the metadata required to develop robust decision provenance and facilitate translation between domain languages, workflow process steps and job/solver types. Through such a system, versatile machine learning techniques can directly integrate into the design practice itself and enable continual observation and optimization of the workflows, repos and processes that underpin the design practice.

In real engineering research and development use cases, DT4D acts as a force multiplier to empower a single engineer to virtually evaluate a multitude of design permutations efficiently and effectively in a hands-off manner, while inherently more seam-

lessly unifying the design process workflow for collaborations among multiple engineers within a community. The system can also provide results from previously completed workflows from across the design community, potentially expediting the generation of relevant synthetic training data to create surrogate models and to validate that the models are of adequate confidence as to inform decision-making. These capabilities greatly accelerate the multidisciplinary engineering process, promote earlier discovery of design contradictions and improvements, and facilitate more agile customization of designs to meet customer specifications. The described integration of heterogenous computational platforms and software architectures provide the infrastructure for the powerful DT4D capabilities.

The Digital Thread for Design framework enables the development and continual improvement of model-based confidence and collaborative data-driven decision making and delivers significant in design-to-decision cycle time, with the opportunity to increase, by multiple orders of magnitude, the number of design alternatives considered. The framework facilitates consistency and integration of physics-based, numerical and machine learning methods for engineers to capture, execute and communicate the workflow, job, data tenancy and authentication metadata contributing to comprehensive decision provenance.

While much work remains to be done, both to augment the scope of capabilities within the framework and to expand its application within the design practice, DT4D is now in regular use providing the next generation of digital thread capabilities.

6 Areas for further study

There is a seemingly unending list of further areas of opportunity for focused development, study, and validation in a field of work as broad as the Model Based Engineering (MBE) of a gas turbine power generation system and the broader turbomachinery marketplace. The development, fulfillment and sustainment of the underlying workflow-enabled design framework required to sustain this MBE practice has a similarly broad and multi-variate set of opportunities. We highlight a few.

- Spin: As previously mentioned, the DT4D framework envisions and accounts for a Spin subsystem as an element of any collaborating compute site. Certainly, there is value in the ability to characterize, containerize and or configure the complete ‘data+compute+job’ tenancy or tenancies required for a given workflow and then to rapidly deploy that tenancy. The complexity of managing differences in functionality, interfaces, and SKUs across the proliferation of hyperscale cloud providers, the opportunity to engage across a breadth of national lab supercomputers available under grant and increasing heterogeneity of both HPC systems and computational methods on our internally developed systems all but ensure inefficiencies and lost times in design practices that rely on manual configuration and intervention.
- Multi-order modeling: While the example practice outlined in this paper incorporates iterative NN model training and design criteria refinement between a single thermal analysis tool of a given order, it is theoretically possible to orchestrate a workflow of solvers such that the level of simulation fidelity available to train the

NN is varied based on the design criteria and that variability is in turn used to select the order/fidelity of the thermal analytic solver in the ensuing step. Integration of inc2 functionality for near-real time extraction of model data may further increase opportunities to reduce the cycle time to develop and validate high fidelity design recommendations. Frameworks for training and evaluating bespoke surrogate models and ensembles can further reduce development costs.

- Federated Auth for MxN: Resolving the complexity of managing person, system and data identity and the required authentication and trust systems remains an open area of research. While COTS federated authentication systems have continued to mature since the advent of our initial work on this framework, the introduction of multi-factor authentication, lack of common standards and practices associated with meeting regulatory control standards, challenges with protecting and recovering lost identities and other challenges have prevented the introduction of truly seamless, secure, and robust identity systems [29].
- Workflow Visualization, Runtime Navigation, and Democratization: The workflows described in this paper – which can dynamically at runtime set (and unset) promises for future triggered actions and spawn new jobs across a distributed computing landscape – poses significant problems relating to visualizing and debugging running workflows, and/or navigating their complete post hoc digital threads in the presence of separating enterprise security perimeters. In addition, the human experience of programming in such a heterogenous and distributed environment can be better studied, as can the potentially democratizing impact of a simplifying model of cross-site computing such as that presented in this paper.

References

1. Arthur R (2020) Provenance for Decision-Making. In: Medium. https://richardarthur.medium.com/provenance-for-decision-making-bf2c89d76ec2. Accessed 6 Jun 2022
2. Digital Thread for Design | GE Research. https://www.ge.com/research/technology-domains/digital-technologies/digital-thread-design. Accessed 6 Jun 2022
3. Feeling The Burn: Inside The Boot Camp For Elite Gas Turbines | GE News. https://www.ge.com/news/reports/feeling-the-burn-inside-the-boot-camp-for-elite-gas-turbines. Accessed 6 Jun 2022
4. High Performance Computing | GE Research. https://www.ge.com/research/technology-domains/digital-technologies/high-performance-computing. Accessed 6 Jun 2022
5. Hatakeyama J, Farr D, Seal DJ Accelerating the MBE Ecosystem Through Cultural Transformation
6. GE Research Uses Summit Supercomputer for Groundbreaking Study on Wind Power | GE News. https://www.ge.com/news/press-releases/ge-research-uses-summit-supercomputer-groundbreaking-study-wind-power. Accessed 6 Jun 2022
7. Ang J, Hoang T, Kelly S, et al (2016) Advanced Simulation and Computing Co-Design Strategy

8. Compute Systems. In: Oak Ridge Leadership Computing Facility. https://www.olcf.ornl.gov/olcf-resources/compute-systems/. Accessed 6 Jun 2022
9. Coughlin T Compute Cambrian Explosion. In: Forbes. https://www.forbes.com/sites/tomcoughlin/2019/04/26/compute-cambrian-explosion/. Accessed 6 Jun 2022
10. What do you mean by "Event-Driven"? In: martinfowler.com. https://martinfowler.com/articles/201701-event-driven.html. Accessed 6 Jun 2022
11. Arthur R (2020) Machine-augmented Mindfulness. In: Medium. https://richardarthur.medium.com/machine-augmented-mindfulness-e844f9c54985. Accessed 6 Jun 2022
12. Wilkinson MD et al. (2016) The FAIR Guiding Principles for scientific data management and stewardship. Sci Data 3:160018. https://doi.org/10.1038/sdata.2016.18
13. Deelman E, Peterka T, Altintas I, et al (2018) The future of scientific workflows. Int J High Perform Comput Appl 32:159–175
14. NERSC SuperFacility API - Swagger UI. https://api.nersc.gov/api/v1.2/#/status/read_planned_outages_status_outages_planned__name__get. Accessed 6 Jun 2022
15. Slurm Workload Manager - squeue. https://slurm.schedmd.com/squeue.html. Accessed 6 Jun 2022
16. (2022) Multiple Program Multiple Data programming with MPI. CFD on the GO
17. Godoy WF, Podhorszki N, Wang R, et al (2020) ADIOS 2: The Adaptable Input Output System. A framework for high-performance data management. SoftwareX 12:100561. https://doi.org/10.1016/j.softx.2020.100561
18. Gallo A (2022) lwfm
19. Salim MA, Uram TD, Childers JT, et al (2019) Balsam: Automated Scheduling and Execution of Dynamic, Data-Intensive HPC Workflows. arXiv
20. Ahn DH, Bass N, Chu A, et al Flux: Overcoming Scheduling Challenges for Exascale Workflows. 10
21. Merzky A, Turilli M, Titov M, et al (2021) Design and Performance Characterization of RADICAL-Pilot on Leadership-class Platforms
22. Chen J, Guan Q, Zhang Z, et al (2018) BeeFlow: A Workflow Management System for In Situ Processing across HPC and Cloud Systems. In: 2018 IEEE 38th International Conference on Distributed Computing Systems (ICDCS). pp 1029–1038
23. This research used the Pegasus Workflow Management Software funded by the National Science Foundation under grant #1664162
24. Arthur R (2021) Co-Design Web. In: Medium. https://richardarthur.medium.com/co-design-web-6f37664ac1e1. Accessed 6 Jun 2022
25. Gamblin T, LeGendre M, Collette MR, et al (2015) The Spack package manager: bringing order to HPC software chaos. In: SC '15: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp 1–12

26. Han J-C, Wright L (2006) Enhanced Internal Cooling of Turbine Blades and Vanes. In: Gas Turbine Handbook. Department of Energy - National Energy Technology Laboratory, p 34
27. Acharya S, Kanani Y (2017) Chapter Three - Advances in Film Cooling Heat Transfer. In: Sparrow EM, Abraham JP, Gorman JM (eds) Advances in Heat Transfer. Elsevier, pp 91–156
28. Tallman JA, Osusky M, Magina N, Sewall E (2019) An Assessment of Machine Learning Techniques for Predicting Turbine Airfoil Component Temperatures, Using FEA Simulations for Training Data. In: Volume 5A: Heat Transfer. American Society of Mechanical Engineers, Phoenix, Arizona, USA, p V05AT20A002
29. How Apple, Google, and Microsoft will kill passwords and phishing in one stroke | Ars Technica. https://arstechnica.com/information-technology/2022/05/how-apple-google-and-microsoft-will-kill-passwords-and-phishing-in-1-stroke/. Accessed 6 Jun 2022