HPCCloud: A Cloud/Web-Based Simulation Environment

Patrick O'Leary
Kitware Inc
Santa Fe, NM 87505
oleary@kitware.com

Mark Christon
Los Alamos National Laboratory
Los Alamos, NM 87545
christon@lanl.gov

Sébastien Jourdain
Kitware Inc
Santa Fe, NM 87505
jourdain@kitware.com

Chris Harris
Kitware Inc
Clifton Park, NY 12065
harris@kitware.com

Markus Berndt
Los Alamos National Laboratory
Los Alamos, NM 87545
berndt@lanl.gov

Andrew Bauer
Kitware Inc
Clifton Park, NY 12065
bauer@kitware.com

ABSTRACT

Advanced modeling and simulation has enabled the design of a variety of innovative products and the analysis of numerous complex phenomenon. However, significant barriers exist to widespread adoption of these tools. In particular, advanced modeling and simulation: (1) is considered complex to use; (2) needs in-house expertise; and (3) requires high capital costs. In this paper, we describe the development of an end-to-end, advanced modeling and simulation cloud platform that encapsulates best practices for scientific computing in the cloud, and demonstrate using Hydra–TH as a prototypical application. As an alternative to traditional advanced modeling and simulation workflows, our Web-based approach simplifies the processes, decreases the need for in-house computational science and engineering experts, and lowers the capital investments. In addition to providing significantly improved, intuitive software, the environment offers reproducible workflows where the full path of data from input to final analyzed results can be saved, shared, and even published.

Categories and Subject Descriptors

H.4 [**Information Systems Applications**]: Miscellaneous; C.2.4 [**Computer Communication Networks**]: Distributed Systems—*Cloud Computing*

General Terms

Management, Measurement, Performance, Design, Economics, Experimentation, Standardization

Keywords

Cloud computing, HPC, Integrated simulation environment

1. INTRODUCTION

Today's solutions for high-performance computing in the cloud provide infrastructure as a service (IaaS), which extends the computational resources available and lowers the required hardware capital investments. But that only addresses one of the barriers to full adoption of the advanced modeling and simulation lifecycle.

For capabilities that are not a core competency of an enterprise, management is hard pressed to justify why the needs of the business cannot be met with a software as a service (SaaS) solution. In the world of enterprise software, on-premises solutions are becoming rare. However, until recently advanced modeling and simulation applications have required classic on-premise solutions and deployments.

The economies of scale that come with the cloud combined with the innovation and agility of SaaS independent software vendors has led to a world of primarily SaaS-based enterprise applications.

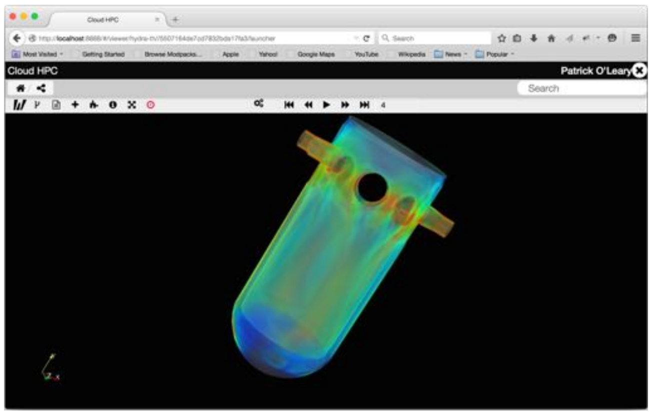


Figure 1: The end-to-end Cloud/Web-based simulation environment, HPCCloud, simplifies the complexities of defining simulations, submissions, and visualizations all from the same browser application.

Hence, a simulation environment that leverages IaaS to lower capital costs, but also delivers innovative, interactive and richer applications that are easier to use via SaaS to an industry's core competency employees would enable the full adoption of the advanced modeling and simulation lifecycle.

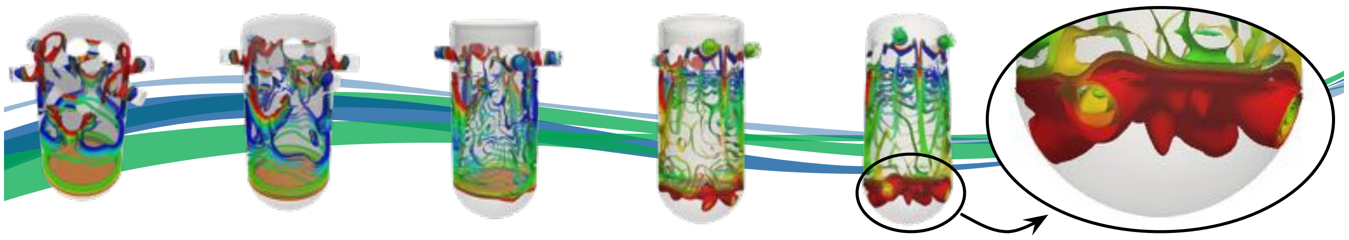


Figure 2: Simulating reactor flow in the lower-plenum during pump startup sequence with Hydra-TH. We see pressure contours in the lower plenum at startup that indicate transient behavior possibly associated with the lower-plenum flow anomaly.

Past investigations concluded a traditional website consisting of HTML, CSS and sometimes a little javascript as an impractical solution for the requirements of advanced modeling and simulation applications. End-users don't just want websites to work and look good on every device, they also want them to be more interactive with three-dimensional graphic content. The creation of new standards (HTML5 and CSS3) allow developers to build richer Web applications for both mobile and desktop browsers.

We have created a Cloud/Web-based simulation environment platform that utilizes new Web technologies to deliver an innovative, interactive SaaS advanced modeling and simulation environment responsive to the end-users' needs. The platform, implementing our Cloud/Web-based simulation environment, makes several contributions to the "traditional simulation" mode.

Data/Task Management System. Our simulation environment takes a traditional simulation and enables interactive, collaborative advanced modeling and simulation in the cloud leveraging a data/task management system. This, in turn, creates a viable solution for many small and medium manufacturing and engineering firms. Our platform:

- Enables authentication and authorization workflows while providing data security from malicious or unintended access or use.
- Creates clusters in clouds through asynchronous distributed task queues utilizing an extended data model and RESTful API.
- Supports elaborate workflows, which include analysis preparation (i.e., pre-processing), simulation including transformative analysis (parameter estimation, sensitivity analysis, uncertainty quantification and optimization), and results analysis including visualization (i.e., post-processing).
- Offers reproducible workflows where the full path of data from input to final analyzed results can be saved, shared, and even published as supporting information.

Simple Input Deck Definition. We created a reusable toolkit for defining simulation input decks, called SimPut. SimPut completely differentiates itself from other mechanisms providing a graphical user interface for defining simulation input decks by:

- Separating the presentation from the input deck keyword/value and format. Thus, the required data entry can be delivered via a front-end in a manner that enables use by non-expert scientist, engineers and technicians.
- Creating built-in documentation within the front-end for a description of the required simulation information.

Integration of 3D Pre- and Post-processing Tools. We added integration for ParaViewWeb applications. ParaViewWeb is a collection of components that enables the use of ParaView's visualization and data analysis capabilities within Web applications. This unique capability:

- Provides access to ParaView servers running on cloud resources launched within the HPCCloud simulation environment.
- Makes it possible to develop and integrate unique three dimensional applications to support pre-processing and post-processing.

Finally, we have exposed the platform of our Cloud/Web-based simulation environment to the scientist through a Web interface that allows him/her to make sophisticated decisions for simulation.

In the sections that follow, we illustrate how HPCCloud meets the goals for the future advanced modeling and simulation.

2. RELATED WORK

HPCCloud has a number of contributions, while building on a vast amount of previously disseminated results, specifically concerning cloud computing, such as those described by Kenneth Wong [30]. Therefore, we quickly review related work for data/task management system, input deck definition and three-dimensional pre-processing and post-processing tools.

Data/Task Management System. The data/task management system curates input and results while performing tasks required of the simulation environment. One option is to start with a data management system for scientific applications [26]. Berman et al [14], Benson et al [13] and countless others have used this approach. HPCCloud extends the data management system to support tasks for creating clusters on clouds, submitting simulations (or visual applications), monitoring simulations (visual applications), and ingesting assets, artifacts and state associated with these tasks.

Input Deck Definition. Our platform provides a universal mechanism for describing a graphical user interface for providing simulation information in the form of an input deck. Eclipse Integrated Computational Environment [25], Computational Model Builder [20], and Salome Platform [27] all provide desktop applications for producing simulation input decks. These tend to present the generic information in the same manner (key-value pairs) as presented in the ASCII input deck file. HPCCloud provides a simple interface to adapt the presentation in a more domain specific manner. The end results are similar to one-off domain specific computational environments such as *Latec* developed by Heulers et al [19]. HPCCloud also integrates help (or the user manual) directly to lower entry requirements of the end-user. This shift moves the end-user requirements from computational expertise to domain expertise.

3D Pre- and Post-processing Tools. Until recently, there simply wasn't any real solution for Web-based simulation visualization other than sharing some selected images, as depicted in Long et al [24]. Work by the teams associated with ParaView [12, 1] and VisIt [15, 7], both of which depend on the Visualization Toolkit [28, 2], have created remote visualization frameworks suitable for modern browsers. HPCCloud encapsulates the ParaViewWeb [22] framework to promote the development and integration of unique domain specific three-dimensional pre- and post-processing applications.

Early adopters of Web-based simulation environments failed to (1) simply communicate with the backend server and to (2) create simple interactive components [23]. They struggled to communicate with network-unaware applications depending on things like the Remote Method Invocation (RMI) mechanism and the Common Object Request Broker Architecture (CORBA). Python wrapping (or similar mechanisms in other languages) turns applications from network-unaware into network/Web-aware. HTML5, CSS3, modern JavaScript ES6, customized REST API, mobile-first designs and single-page applications have made creating an easy to use front-end for Web-based simulations possible.

3. APPROACH

We have created HPCCloud, our implementation of the Cloud/Web-based simulation environment platform. HPCCloud is

1. Easy to use,
2. Usable by industry's core competency employees; and
3. Eliminates high capital costs.

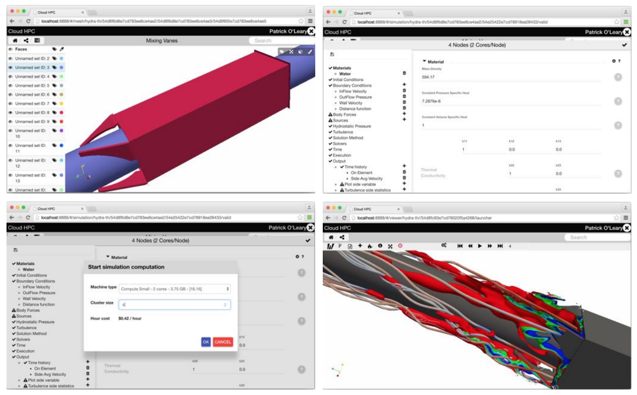


Figure 3: HPCCloud implementation demonstrating an end-to-end simulation environment. (Upper-Left) The mesh tagging tool that leverages ParaViewWeb to assign meaningful names to elements, materials, and side sets. (Upper-Right) Defining the input deck for Hydra-TH, specifically the material properties of water. (Lower-Left) Describing an HPC cluster to be created in the AWS Cloud and submitting the defined simulation. (Lower-Right) The visualizer tool that again leverages ParaViewWeb for analyzing and visualizing the simulation results.

The platform supports a complete simulation lifecycle, which removes the barriers to using advanced modeling and simulation throughout industry. First and foremost, HPCCloud strategically eliminates the need to hire non-core competent computational science and engineering experts, by offering a simple to use interface from the comfort of their browsers. From an industry perspective, these positions represent recurring labor costs that in most cases are unable to perform any traditional role for the firm. Hiring these experts represented a leap of faith that many companies are not willing to take in adopting advanced modeling and simulation workflows. Finally, the platform leverages the cost effective cloud computing embraced by other facets of their enterprise computing architecture.

3.1 Data/Task Management System

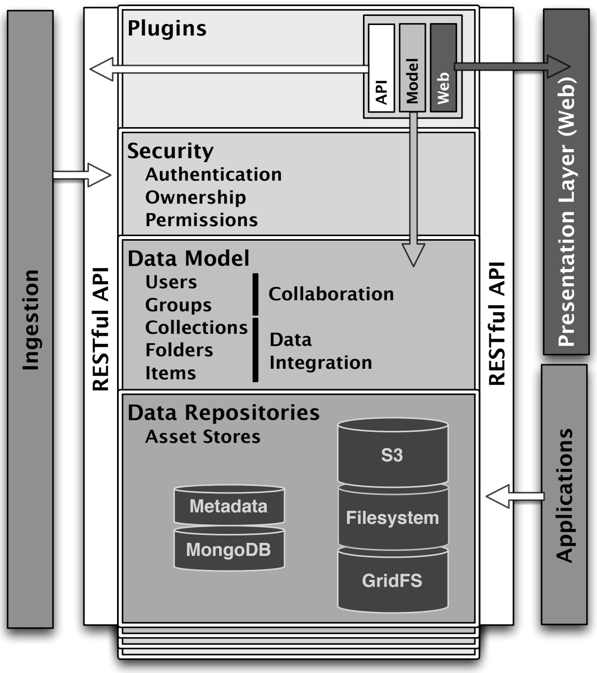


Figure 4: The Girder architecture.

For a data management system, we leveraged Girder [4]. Girder is a data management toolkit. The goal of Girder is to provide developers with a scalable data management system, so that they can focus on building great applications. To meet this goal, Girder is designed to be robust, fast, scalable, extensible, and easy to understand. Built on Python, Girder is a complete back-end (server side) technology that can be used with other applications via its RESTful API, or via its own front-end. Girder is open source, licensed under the Apache License, Version 2.0.

Girder utilizes a representational state transfer (REST) architecture style. RESTful APIs typically communicate over the hypertext transfer protocol (HTTP) with the same verbs (GET, POST, PUT, DELETE, etc.) used by web browsers to retrieve web pages and send data to remote servers. The components of the Girder architecture, shown in Figure 4, will be discussed in the following subsection.

3.1.1 *Girder Data Management System*

Our platform requires a data repository(ies) to store files, metadata and entities, and Girder provides a variety of asset stores to meet this need. In Girder, some files are simply links to external unified resource locators (URLs), all others must be contained within an Asset Store. Each Item may contain any number of arbitrary key-value pairs, termed *metadata*, which Girder stores in a Mongo database (MongoDB [5]).

In Girder, an asset store is an abstraction representing a repository where the raw bytes of Files are actually stored. The following are some current concrete implementations of Girder asset stores: an Amazon Web Services' S3 bucket; the local filesystem of the server using content-addressed storage; and/or directly within the MongoDB using the GridFS model.

Girder's base data model provides several components necessary for collaboration and data integration required for the Cloud/Web-based simulation environment. For collaboration, we have *user(s)* and *group(s)*. Users can be granted permissions on resources in the system directly, and can belong to groups. One of the main purposes of groups is to allow role-based access control. Resources can grant access to groups rather than just individual users. These entities are key to sharing assets and artifacts of simulation workflows with colleagues (or other users).

Girder's data model also provides Collection(s), Folder(s), Item(s) to support data integration. Collections are the top level objects in data organization. There can be many folders and items within each collection, and the collection itself is also an access controlled resource. Folders can contain a combination of folders and items. Folders have permissions set on them, and the Items within them inherit permissions from the Folders that containing them. Items, which may contain 0 or more files, are the basic unit of data in Girder.

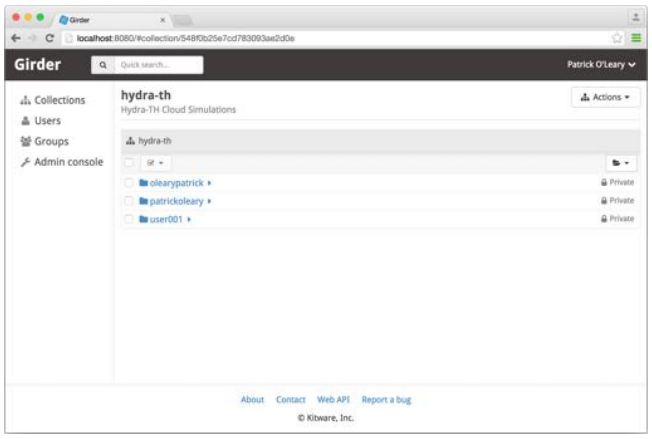


Figure 5: The Girder administration Web-pages.

Security is a salient component of any data management system. The three fundamental tasks of security are identity management (authentication), access control (authorization) and data security. Developers can use Girder's RESTful API or administration Web-pages, depicted in Figure 5 to manage users, groups and permissions easily. Role-based access control is built into Girder, and fine-grained permissions can be customized down to the User level. Ownership is defined on Collections, Folders and Items and the ownership can be a User or Group.

Data security (data integrity) involves the protection of the system from malicious or unintended access or use. Girder addresses the following data security concerns. Data security techniques are fluid, but Girder currently addresses: session management, database injection, cross-origin resource sharing (CORS), cross-site scripting (XSS), and cross-site request forgery (CSRF).

Finally, the base data model, RESTful API, Web content, and security built into Girder will not support every application. Therefore, Girder provides the capability of extending or modifying these components through plugins. This plugin framework is designed to allow Girder to be as flexible as possible, on both the client and server sides.

3.1.2 *Cumulus Plugin*

The Cloud/Web-based simulation environment platform must build, configure, monitor, and manage clusters in the Cloud. To meet this requirement, we developed Cumulus, a Girder plugin. Cumulus is used to extend Girder's base data model and RESTful API to support running tasks.

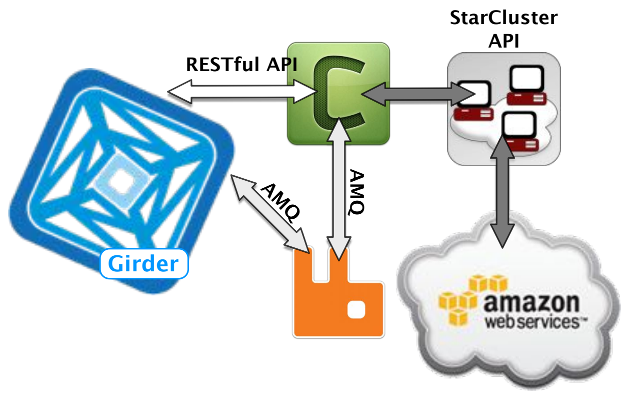


Figure 6: Data flow diagram demonstrating the Cumulus architecture.

Cumulus extends Girder's base data model with the following entities: cluster configurations, clusters, scripts, jobs and tasks. All REST endpoints are asynchronous so any long running tasks are offloaded to Celery, a distributed task queue [3]. Celery requires a message broker to communicate between clients and workers, we leveraged RabbitMQ, an open source message broker software that implements the Advanced Message Queuing Protocol (AMQP) [6].

The RabbitMQ Broker is used to hold a queue of messages (tasks). Girder, through the Cumulus plugin, uses the Python Celery client to put messages (tasks) on the queue (Girder is the producer), and a Celery worker pulls them off (the consumer). RabbitMQ allows Girder to communicate this asynchronous work.

Cumulus offloads long running tasks to Celery to prevent the platform from being unresponsive, and to provide a very simple linear scaling for the overall system.

Our high-performance computing (HPC) model, for the platform, is based on cluster computing. A computer cluster consists of a set of loosely or tightly connected computers that work together so that, in many respects, they can be viewed as a single system. This system has dominated the landscape for advanced modeling and simulation since the late 1990s [29].

To implement our model in the platform, we depend on StarCluster [21]. StarCluster simplifies the process of building, configuring, and managing clusters of virtual machines on Amazon's Elastic Compute Cloud EC2 cloud. For the purposes of submitting jobs, StarCluster configures Sun Grid Engine (SGE) when the cluster is started. All other tasks, such as monitoring and data movement, are handled by scripts invoked using Celery.

3.2 Simple Input Deck Definition

In the process of creating a reusable toolkit for defining simulations for the Cloud/Web-based simulation environment platform, we created a universal Web-based interface, called SimPut, for defining simulation input decks. SimPut:

- Works for any simulator;
- Provides built-in documentation; and
- Exports the input deck to a Web-friendly intermediate representation.

SimPut completely differentiates itself from other tools providing a graphical user interface for defining input deck simulation information because the presentation is not tied to the traditional input deck keyword-value pairs and format. This flexibility allows platform developers to create a front-end for a particular simulation that can be used by common scientists, engineers and technicians from a firm, not strictly by specialized computational scientist or engineers.

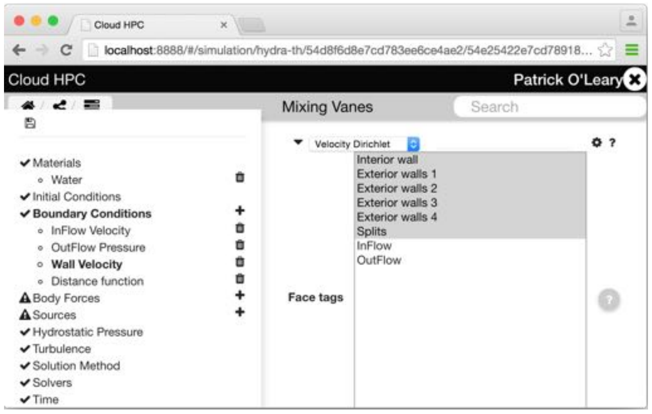


Figure 7: An end-user is presented with HPC-Cloud's version of SimPut to edit and define a simulation input deck. Specifically, SimPut presents the required information to define a velocity Dirichlet boundary condition, in this case a wall velocity, for Hydra–TH.

3.2.1 *Intermediate Representation*

SimPut uses a Web-friendly json file, `simulation.json`, as the intermediate representation of simulation attribute information. The `simulation.json` in Listing 1 depicts the simplest intermediate representation used with SimPut.

The syntax for the intermediate representation json file, `simulation.json`, is as follows:
"type" (required) – the type of simulator from a predefined list integrated into SimPut.
"external" (optional) – for Hydra–TH, our test-bed CFD simulation code, external is required and holds mesh tags to simplify the input definition process. The following categories are currently exposed for mesh-based simulators:

- "material-tags" – material sets (labels and values) describes blocks in the mesh of a specific material type.
- "element-tags" – element sets (labels and values) representing individual elements in the mesh.
- "side-tags" – side sets (labels and values) depicts *faces* of the mesh that might be used to define boundary conditions.

"data" (optional) – Stores the previously defined attributes using SimPut.

```
{
   "type": "openfoam"
}
```

Listing 1: simulation.json file to be virtually or actually dropped into SimPut, depending on whether it is within the HPCCloud platform or the standalone SimPut application.

3.2.2 *Simulation Module*

`simulation.js` is a simulation module bundled by webpack [10] consisting of a json file, `definitions.json`, containing input deck definitions; a javascript file, `transform.js`, that transforms the Web entered data; a directory, `html`, of html help documentation; and a Jade template file, `export.jade`, to export the formatted simulation input deck.

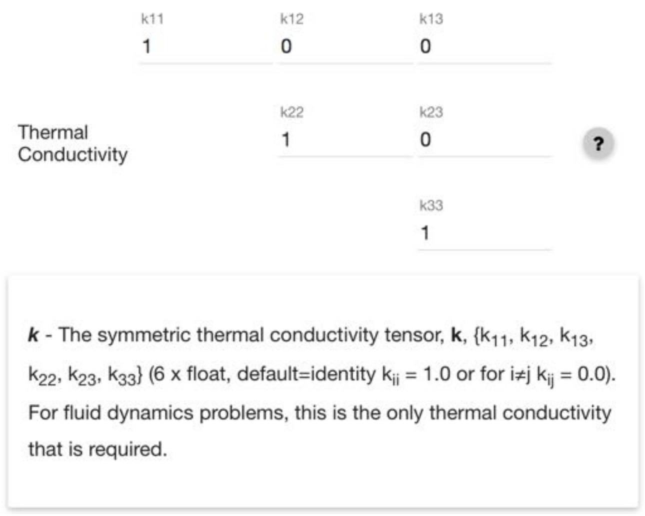
The `definitions.json` defines the simulator input deck parameters completely for all paths. The main required categories include: *order* – the order to present the views in SimPut; *views* – the organization of attributes within a view; and *definitions* – the definition of attributes, typically keyword-value pairs, of the simulation including: type, format and style. The `transform.js` is a JavaScript file that transforms the extracted Web-form data to the Jade data model for input deck generation. It consists of specialized extraction functions to pull the data from the forms for each attribute defined in `definitions.json`. The `help` directory contains an html file per possible element in input deck. These are presented as html, not as plain text, to provide an expanded palette to express the intent of the desired input. Finally, the `export.jade` file is a Jade template that formats the input deck per simulator format requirements.

The `simulation.js` module, for each simulator to be exposed in HPCCloud, must be distributed with the platform. These modules are extremely lightweight, that is less than a megabyte per simulator, and can be used within HPCCloud or via a server-less, standalone desktop Web application.

3.2.3 *Presentation Layer*

The presentation layer for SimPut is rich and simulation aware. The keyword-value pair format of the input deck does not have to be presented without thought to the end-user. Instead, the information can be requested in a sequence and form that makes the conceptual intent clear to common scientists, engineers and technicians. SimPut does not require computational science and engineering expertise (see Figure 7). Rather, it requires domain knowledge of the problem.

Thus, the view can take a form that more closely resembles the intent of the requested information. In Figure 8, we see that the thermal conductivity tensor of a material is presented as an upper 3-by-3 matrix as opposed to a list of keyword-value pairs (k11=1.0, k12=0.0, k13=0.0, k22=1.0,



	k11	k12	k13	
	1	0	0	
Thermal Conductivity		k22	k23	?
		1	0	
			k33	
			1	

k - The symmetric thermal conductivity tensor, **k**, {k_{11}, k_{12}, k_{13}, k_{22}, k_{23}, k_{33}} (6 x float, default=identity k_{ii} = 1.0 or for i≠j k_{ij} = 0.0). For fluid dynamics problems, this is the only thermal conductivity that is required.

Figure 8: The presentation of the thermal conductivity tensor in SimPut (top) and the help entry for the same element of the input deck (bottom).

k23=0.0, k33=1.0). Documentation is also placed as close as possible to the element. Simply hit the question mark next to the element, and the help information is presented below the element. There is no popup window blocking access to either the element or the help, and there is no need to bring up another application that presents help documentation.

3.3 Integration of 3D Pre- and Post-processing Tools

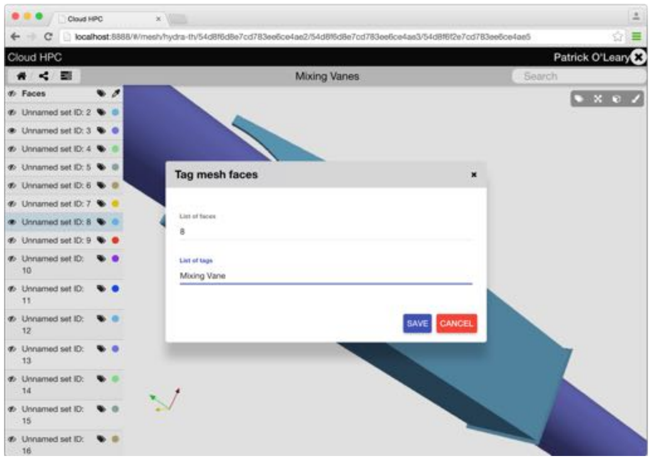
To be efficient and effective, the Cloud/Web-based simulation platform requires three-dimensional graphics and visualization for both pre-processing and post-processing to enable simple to complex advanced modeling and simulation workflows. Hence, we leveraged ParaViewWeb the Web analysis and visualization framework.

3.3.1 *ParaViewWeb*

ParaViewWeb is a collection of components that expose ParaView's analysis and visualization capabilities within Web applications. ParaViewWeb relies on the latest HTML 5.0 technologies, such as WebSocket [11] and WebGL [9], and works with both desktop and mobile Web browsers.

At its core, ParaViewWeb simplifies communication with a ParaView server running on a remote visualization node or cluster using a lightweight API, which utilizes Autobahn's Python and JavaScript WAMP [8] client libraries. Using this API, Web applications can easily embed interactive three-dimensional visualization components leveraging the ParaView and/or Visualization Toolkit (VTK).

For the platform, we enhanced Cumulus, the Girder plugin, to route front-end request for ParaViewWeb applications to the ParaView servers running on private networks that Cumulus launched from Amazon Web Services cloud resources. Cumulus stores a map of routes to the remote ParaView servers, and supports the front-end requests to the Apache server.

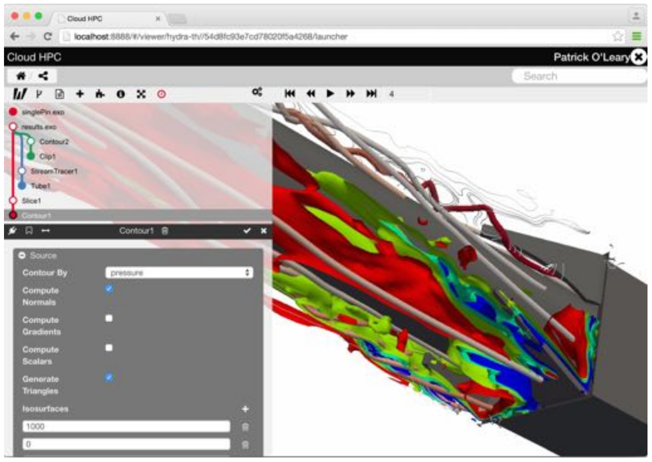


Tag mesh faces

List of faces: 8

List of tags: Mixing Vane

Figure 9: The mesh tagging application allows the end-user to visualize and annotate material, element and side sets of the simulation mesh. In this image, we see set ID 3 (in purple), which represents a nuclear reactor fuel pin, and set ID 8 (in blue), which represents a mixing at startup that indicate transient behavior possibly associated with the lower-plenum flow anomaly. The end-user uses a pop-up dialog to assign set ID 8 as a mixing vane.



Contour By: pressure

Compute Normals: ✓

Compute Gradients: ☐

Compute Scalars: ☐

Generate Triangles: ✓

Isosurfaces: 1000, 0

Figure 10: Visualizer's interface provides access and control to all of the ParaView framework. In this example, we have built a sophisticated visualization pipelines of the pressure (isosurfaces and isolines) and velocity (streamlines colored by vorticity) in water flow through a mixing vane around a single nuclear reactor pin.

3.3.2 *Mesh Tagging*

In several advanced modeling and simulation workflows, the mesh represents the discretization of the problem, or specific geometric model, based on the employed numerical methods. For pre-processing, it is convenient to decompose the mesh into entities, sets and tags. Entities are elements of the mesh such as vertex, edge, triangle, etc. Sets are arbitrary groupings of mesh entities and/or other sets. Tags are named data which can be assigned to the mesh as a whole, individual entities, or sets. The combination of these three elements represents a powerful yet simple interface for representing metadata or application-specific data. For example, entities, sets and tags can be used together to describe geometric topology such as boundaries in a mesh.

The mesh tagging application shown in Figure 9 (powered by ParaViewWeb) allows the end-user to provide meaningful names to material, element, and side sets. These meaningful names will in turn be presented to the end-user in SimPut, when defining a simulation's material properties, initial conditions, boundary conditions, forces, and sources.

3.3.3 *Visualizer*

Visualizer allows interactive analysis and visualization that is similar to the capabilities of the ParaView Qt application.

The platform leverages the Visualizer application, depicted in Figure 10 for interactive analysis and visualization of simulation results. Note that the control panels can be hidden by clicking on the ParaViewWeb logo. With Visualizer, the end-user seamlessly develops sophisticated analysis and visualization pipelines based on ParaView sources and filters with complete control of the sources, representations and views. Visualizer is easier to use than the ParaView Qt application, because the environment is setup (MPI client/server, file access, ...) for the end-user, without degradation in visual quality, utility or performance.

4. RESULTS

The innovations in data/task management system, simple input deck definition, and integration of three-dimensional pre- and post-processing tools combine to make the Cloud/Web-based simulation environment platform easy-to-use. In this section, we discuss results from complete advanced modeling and simulation workflows on important industry problems.

4.1 Hydra–TH CFD Software

For testing purposes, we utilized Hydra–TH [16, 17] developed as part of the Hydra Toolkit (Hydra) for scalable scientific simulation, led by Dr. Mark A. Christon, Senior Scientist at Los Alamos National Laboratory. The Hydra architecture provides lightweight, high performance, and reusable code components for agile application development. Hydra has been used for methods and algorithms research with a broad array of physics, discretization techniques, and solution methods. Currently the toolkit supports finite-element solvers, multiple hydrodynamics solvers, ridged-body dynamics, and more. Additionally, finite-volume solvers are also available.

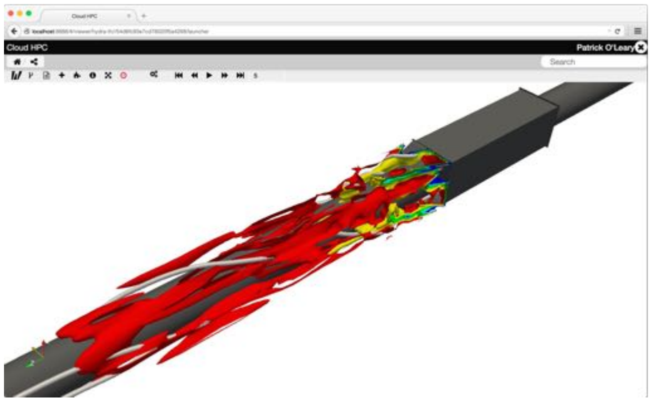


Figure 11: Visualization of the GTRF problem: The single rod, spacer, and mixing vanes (in grey) with pressure isosurfaces and velocity streamlines.

Hydra–TH has been developed for the Consortium for Advanced Simulation of Light-Water Reactors (CASL) to create a computational capability that enables the simulation of the thermal–hydraulics processes inside a nuclear reactor at unprecedented fidelity. These simulations can use tens of thousands of compute cores on the largest supercomputers in the world and enable the detailed resolution of turbulent flow fields and their interaction with the reactor fuel assembly.

4.1.1 *Grid-to-Rod Fretting Problem*

The grid-to-rod fretting (GTRF) problem in pressurized water reactors is a flow-induced vibration problem that results in wear and failure of the rods in nuclear fuel assemblies. Currently, it has not been possible to completely characterize the flow-induced fluid-structure interaction (FSI) problem for the GTRF problem. Indeed, given the incompressible nature of the coolant, the relatively high Reynolds number, and the flexible character of the fuel rods and spacers, the FSI problem at the reactor core scale is daunting. Recent advances with Hydra–TH may be found in references [16, 18].

In this paper, we ran a Smagorinsky subgrid-scale model turbulence with Smagorinsky model constant of 0.18 for a representative single rod with three spacers. Hydra–TH is used to compute the time-accurate and fully three-dimensional flow field for the single rod.

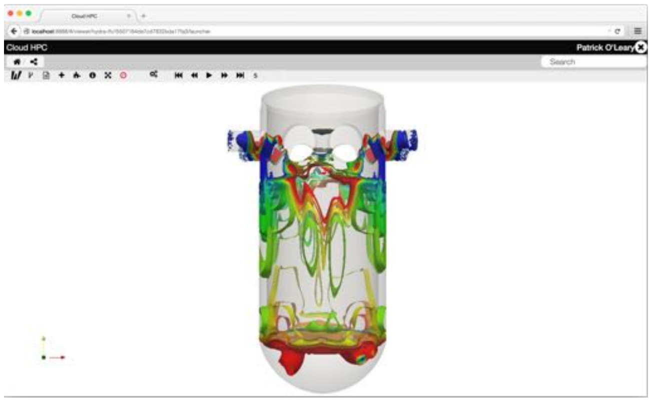


Figure 12: Visualization of the LPFA problem: The vessel (transparent grey) and pressure isosurfaces at startup.

4.1.2 *Lower Plenum Flow Anomaly Problem*

The second problem involved simulating a known reactor flow anomaly, Lower Plenum Flow Anomaly (LPFA), with Hydra-TH to develop a better understanding of the sensitivity of the flow distribution to differential inlet flow.

We ran a Spalart-Allmaras model turbulence for a representative reactor vessel. Hydra–TH is used to simulate reactor flow in the lower plenum during the pump startup sequence with an objective of identifying actions to reduce (or at least control) the variation.

4.2 Performance

Timings were performed on both the GTRF and LPFA problems, but, given their similar problem size and performance characteristics, we simply report the results of the GTRF problem.

The Hydra–TH input deck (4 KB) and single rod with spacer mesh (232 MB) represented an approximately 4 million cell simulation that was run for 1000 time steps. The *Running* time included computation, MPI-based communication, output of parallel results file (total of approximately 5.5 GB) at every 10 timesteps, and dumping a restart file (approximately 500 MB) at the 0 timestep and 1000 timestep.

We leveraged Amazon's EC2 resizable compute capacity in the cloud for all performance results. In particular, we utilized the C4 instances featuring high frequency Intel Xeon E5-2666 v3 (Haswell) processors with enhanced networking and clustering.

For the Figure 13 and Figure 14, a c4.large EC2 instance was used, which provides 2 cores, 3.75 GB of main memory, and dedicated EBS storage with 500 Mbps throughput.

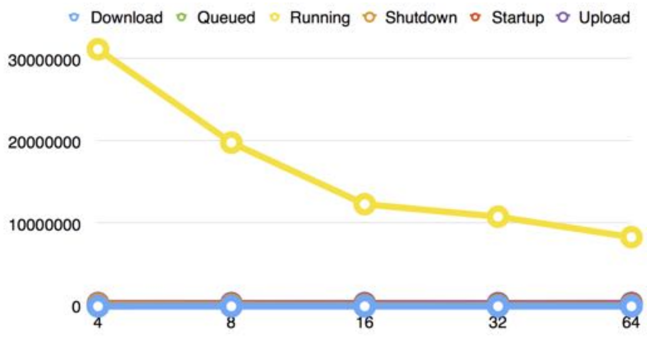


Figure 13: Timings, in milliseconds, of Cloud-based run of a single pin with a mixing vane from 4 to 64 nodes (on 2 core nodes).

In Figure 13, we see expected speedup similar to that of our on-premise high-performance computing cluster. As the number of cores increases from 8 cores to 128 cores (in the graph this maps to 4 nodes with 2 cores each to 64 nodes with 2 cores each), we see the overall simulation time decrease from approximately 500 minutes to 141 minutes. The turnover represented after 16 nodes or 32 cores simply implies that the money spent moving to 64 cores and 128 cores provided a smaller return for investment.

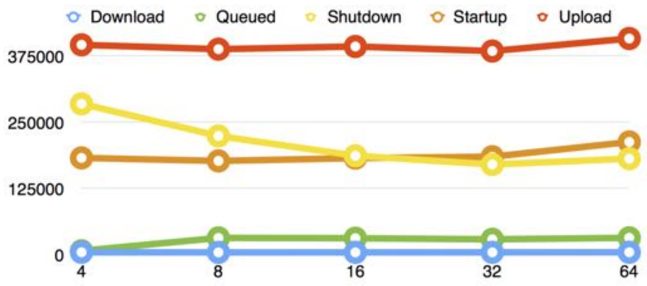


Figure 14: The performance in milliseconds of just download, queued, shutdown, startup and upload.

Figure 14 depicts the other time costs surrounding the simulation including:

- Download - The time to ship the input deck and mesh to the cluster nodes from the data management system;
- Queued - The time spent in the Sun Grid Engine queuing system;
- Shutdown - The time to shutdown (release) the EC2 instances of the cluster;
- Startup - The time to startup a cluster from EC2 instances; and
- Upload - The time to upload the results to the data management system.

The download and queued time for a simulation is negligible. Starting up and shutting down a cluster is fairly uniform at approximately two minutes regardless of whether we are creating a single node cluster or a 64 node cluster. We believe that most advanced modeling and simulation end-users would be overjoyed with the platform's time until run performance when compared to the time spent in queue with their on-premise resources. The most concerning metric in Figure 14 is the upload time. As our problems grow, both our download and upload times will grow, but this concern can be remediated by creating persistent high-performance storage instance offered by OrangeFS, Lustre, and other customized solutions.

Table 1: EC2 Instances

Model	vCPU	Memory	EBS Throughput
c4.large	2 Cores	3.75 GB	500 Mbps
c4.xlarge	4 Cores	7.5 GB	750 Mbps
c4.2xlarge	8 Cores	15 GB	1,000 Mbps
c4.4xlarge	16 Cores	30 GB	2,000 Mbps

Table1 shows the EC2 instances used in the timings depicted in Figure 15 measuring the effects of node core density.

With a 10 Gbps Ethernet interconnect, we see interprocessor communication time increasing with the number of cores per node. We would expect the communication time to flatten out as the interconnect speed increases. But, even if the interconnects don't evolve as fast as compared to on-premise resources, eventually simulations on both EC2 and on-premise resources present opportunities from modified algorithms that take advantage of multiple core and possibly many core architectures.

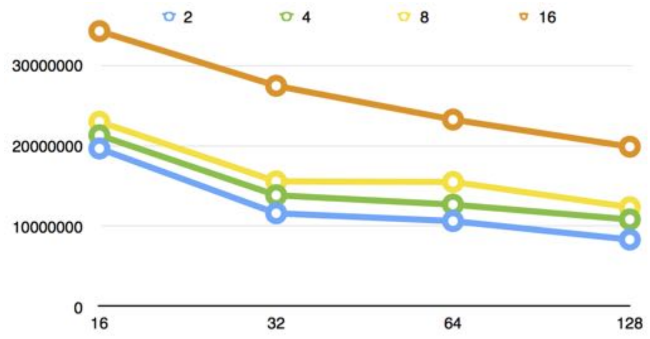


Figure 15: Timing, in milliseconds, of Cloud-based run of a single pin with a mixing vane from 16 to 128 cores where we varied the node core density from 2 cores per node to 16 cores per node.

5. CONCLUSION

We have developed a novel platform for an end-to-end, advanced modeling and simulation Cloud/Web-based simulation environment that encapsulates best practices for scientific computing in the cloud. As implemented in this paper, the platform makes advanced modeling and simulation (1) easy to use, (2) by industry's core competency employees; and (3) eliminates high capital costs.

We have demonstrated the platform using open-source tools and shown how scientists and engineers can easily explore industry relevant advanced modeling and simulation problems. Our results demonstrate significant improvement in ease of use based on the end-to-end solution, especially when considering the number of applications required to represent the complete workflow. The results section demonstrates that the performance is comparable to on-premise resources. Finally, we have shown that HTML5, CSS3, modern JavaScript ES6, customized REST API, mobile-first designs and single-page applications have made creating easy to use and understand front-ends for Web-based simulations possible.

6. ACKNOWLEDGMENTS

This work was funded by Mr. Dan Funk, NEAMS Program, Office of Nuclear Energy, Department of Energy (DOE), Fast Track SBIR award DE-SC0010119 and Phase I SBIR award DE-SC0012037. Funding for Mark Christon, Markus Berndt, and Andy Bauer was provided by the Consortium for Advanced Simulation of Lightwater reactors (CASL). We are indebted to Annalisa Manera and Victor Petrov, both of the Nuclear Engineering and Radiological Sciences at the University of Michigan, for the GTRF model, and Rose Montgomery, of the Tennessee Valley Authority, for the LPFA model. Finally, we thank Scott Wittenburg for a number of the visualizations that appear in the paper.

7. REFERENCES

[1] ParaView. Online - http://www.paraview.org/, June 2010.

[2] VTK. Online - http://www.vtk.org/, June 2010.

[3] Celery. Online - http://www.celeryproject.org, April 2015.

[4] Girder. Online - http://www.tangelohub.org/girder/, April 2015.

[5] MongoDB. Online - https://www.mongodb.org, April 2015.

[6] RabbitMQ. Online - http://www.rabbitmq.com, April 2015.

[7] VisIt. Online - http://visit.llnl.gov, June 2015.

[8] WAMP. Online - http://wamp.ws, April 2015.

[9] WebGL. Online - https://www.khronos.org/webgl/, April 2015.

[10] webpack. Online - http://webpack.github.io, April 2015.

[11] WebSockets. Online - https://www.websocket.org, April 2015.

[12] J. Arhens, K. Brislawn, K. Martin, B. Geveci, C. Law, and M. Papka. Large scale data visualization using parallel data streaming. *IEEE Computer Graphics and Applications*, 4(21):34–41, 2001.

[13] D. A. Benson, K. Clark, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and E. W. Sayers. GenBank. *Nucleic Acids Research*, 2013.

[14] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, and P. E. Bourne. The Protein Data Bank. *Nucleic Acids Research*, 28(1):235–242, 2000.

[15] H. Childs, E. S. Brugger, K. S. Bonnell, J. S. Meredith, M. Miller, B. J. Whitlock, and N. Max. A contract-based system for large data visualization. In *Proceedings of IEEE Visualization 2005*, pages 190–198, 2005.

[16] M. A. Christon. Hydra-TH Theory Manual. Technical Report LA-UR 11-05387, Los Alamos National Laboratory, September 2011.

[17] M. A. Christon, J. Bakosi, M. M. Francois, B. Nadiga, M. Berndt, and A. K. Stagg. A hybrid incremental projection method for thermal-hydraulics applications. *in preparation for CASL Special Issue Journal of Computational Physics*, 2014. ((Los Alamos National Laboratory LA-UR 14-28406).

[18] M. A. Christon, R. Lu, J. Bakosi, B. Nadiga, Z. Karoutas, and M. Berndt. Large-eddy simulation, fuel rod vibration and grid-to-rod fretting. *in preparation for CASL Special Issue Journal of Computational Physics*, 2014. (Los Alamos National Laboratory LA-UR 14-28497).

[19] L. Heulers, F. Fernex, and N. Leclaire. LATEC : A generic workbench dedicated to criticality calculations. In *International Conference on Nuclear Criticality 2011. ICNC 2011.*, September 2011.

[20] A. Hines, S. Howington, B. White, O. Eslinger, C. Kees, M. Farthing, R. O'Bara, R. Blue, Y. Yaun, A. Bauer, and B. King. Computational Model Builder (CMB): A Cross-Platform Suite of Tools for Model Creation and Setup. *2009 DoD High Performance Computing Modernization Program Users Group Conference*, pages 370–373, 06 2009.

[21] C. Ivica, J. Riley, and C. Shubert. StarHPC – Teaching parallel programming within elastic compute cloud. In *Information Technology Interfaces, 2009. ITI '09. Proceedings of the ITI 2009 31st International Conference on*, pages 353–356, June 2009.

[22] S. Jourdain, S. Wittenburg, and P. O'Leary. Python Enabled ParaViewWeb for HPC Analysis and Visualization. In *Python in HPC Workshop, International Conference for High Performance Computing, Networking, Storage and Analysis (SC '14)*, November 2014.

[23] J. Kuljis and R. J. Paul. A Review of Web Based Simulation: Whither We Wander? In *Proceedings of the 32Nd Conference on Winter Simulation*, WSC '00, pages 1872–1881, San Diego, CA, USA, 2000. Society for Computer Simulation International.

[24] J. Long, P. Spencer, and R. Springmeyer. SimTracker - Using the Web to track computer simulation results. In *Proceedings of the1999 International Conference on Web-Based Modeling and Simulation*, August 1999.

[25] A. McCaskey, T. Patterson, and J. J. Billings. Modernizing Simulation Input Generation and Post-Simulation Data Visualization with Eclipse ICE @EclipseCon North America 2015, online -

http://goo.gl/v1n6wx, March 2015.

[26] R. Moore. Data Management Systems for Scientific Applications. In *Proceedings of the IFIP TC2/WG2.5 Working Conference on the Architecture of Scientific Software*, pages 273–284, Deventer, The Netherlands, The Netherlands, 2001. Kluwer, B.V.

[27] A. Ribes and C. Caremoli. Salome platform component model for numerical simulation. In *Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International*, volume 2, pages 553–564, July 2007.

[28] W. Schroeder, K. Martin, and B. Lorensen. *An Object Oriented Approach to 3D Graphics*. Kitware, Inc., 4 edition, 2004.

[29] T. Sterling, D. J. Becker, D. Savarese, J. E. Dorband, U. A. Ranawake, and C. V. Packer. Beowulf: A Parallel Workstation For Scientific Computation. In *In Proceedings of the 24th International Conference on Parallel Processing*, pages 11–14. CRC Press, 1995.

[30] K. Wong. A Map to Simulation on the Cloud @Desktop Engineering, online - http://goo.gl/ijvbuc, July 2014.