Galaxy

A framework for multi-site, cross-platform simulation and optimization

Galaxy Team (Directed Energy Directorate of the Air Force Research Lab and Stellar Science Ltd Co)
POC: galaxy-support@stellarscience.com

1 The Challenge of Heterogeneous Computing Resources

Today a practicing scientist or engineer has access to a wide range of computing resources, as illustrated in Figure 1. Often the most powerful resources available are dedicated high-performance computing (HPC) systems, such as the Department of Defense (DoD) Supercomputing Resource Centers (DSRCs) operated by the DoD High Performance Computing Modernization Program (HPCMP). Using an HPC system effectively for simulations can present many challenges, however. Moving a simulation workflow from a local workstation onto an HPC system requires many non-trivial steps. A user needs to learn the system, port software, structure work into jobs, write Portable Batch System (PBS) job scripts, monitor queued and running jobs, and synchronize files back and forth between systems. Furthermore, because of differences in computer architecture, available software and compilers, PBS queue policies, and general system configuration, much of this work must be repeated for each HPC site and every newly-commissioned system.

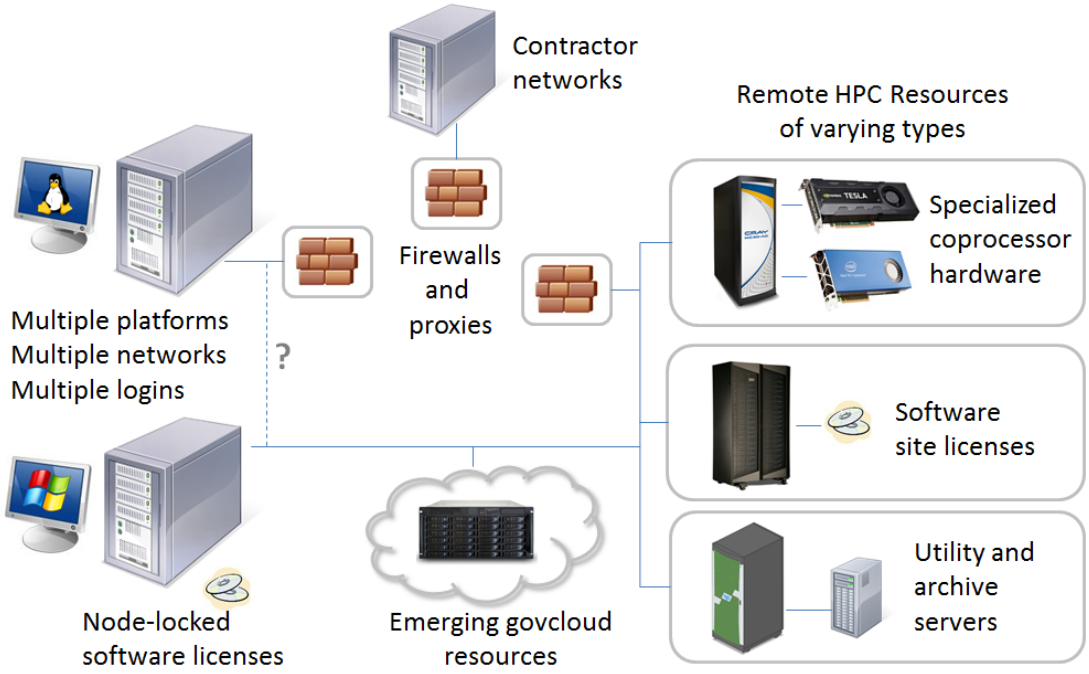

Figure 1: The Heterogeneous Computing Environment

Similar difficulties can arise when using other types of shared resources as well. Even within an organization, users sometimes must deal with multiple networks, logins, and platforms, making it cumbersome to move computational work between systems. Efforts to automate a workflow can also be frustrated by administrative barriers such as firewalls and restrictions on installing new software.

Location constraints are another facet of the challenge. Sometimes simulation steps are tied to a specific location due to memory requirements, available compilers, node-locked software licenses, or other constraints. Splitting a workflow across sites is complex and inevitably requires manual intervention for steps such as job submission and file transfer, which require user login credentials.

Together, these challenges impact productivity in several ways. Due to the high up-front cost of accessing powerful resources, some users work locally and miss an opportunity for faster or more-detailed simulation results. Users who transition to working on HPC systems face extra manual work in their simulation workflow. Finally, and perhaps most significantly, the overhead involved discourages users from pursuing complex and ambitious computational tasks like optimizations or end-to-end simulations that involve multiple simulation tools.

2 The Galaxy Framework

With the productivity challenges of using HPC systems in mind, the Galaxy Team has built a framework called *Galaxy* that streamlines the design, execution, and sharing of simulation workflows on HPC systems. Galaxy was conceived as a subproject of the High-Power Microwave (HPM) HPC Software Applications Institute (now known as the Directed Energy Software Institute [DESI]), which was funded by the HPCMP to focus on virtual prototyping of HPM weapons. Despite its origin in the HPM domain, however, Galaxy is a fully general framework that can be used off-the-shelf for any simulation task, and its user interface can be customized for any domain of interest.

Development on Galaxy is ongoing, but an active community of HPM engineers and scientists uses Galaxy in daily work, and the user base has recently begun expanding into other domains such as laser modeling and antenna characterization. Galaxy has significantly boosted the productivity of its users, enabling them to achieve results that would have been unreachable with their old workflows. In the past, AFRL's HPM division performed optimizations by developing custom tools and scripts for individual designs. Compared with the old approach, Galaxy saves considerable work by providing easy access to a suite of powerful optimization algorithms and making it trivial to parallelize a problem across multiple sites. On one recent large optimization run, Galaxy found a design that was 10% more powerful and 2% more efficient than the starting point, which had been tuned using an older method. That run utilized all the available DSRC systems and ran for two weeks, yet required only one week of set-up time.

The overall architecture of Galaxy is summarized in Figure 2. The Galaxy framework is a suite of tools and web services that address the main aspects of a simulation workflow, including interactive design and debugging, large-scale production runs, data analysis, and various administrative tasks. Galaxy also allows work groups to share common job configuration data for the available HPC systems and to search and download past simulation models and results.

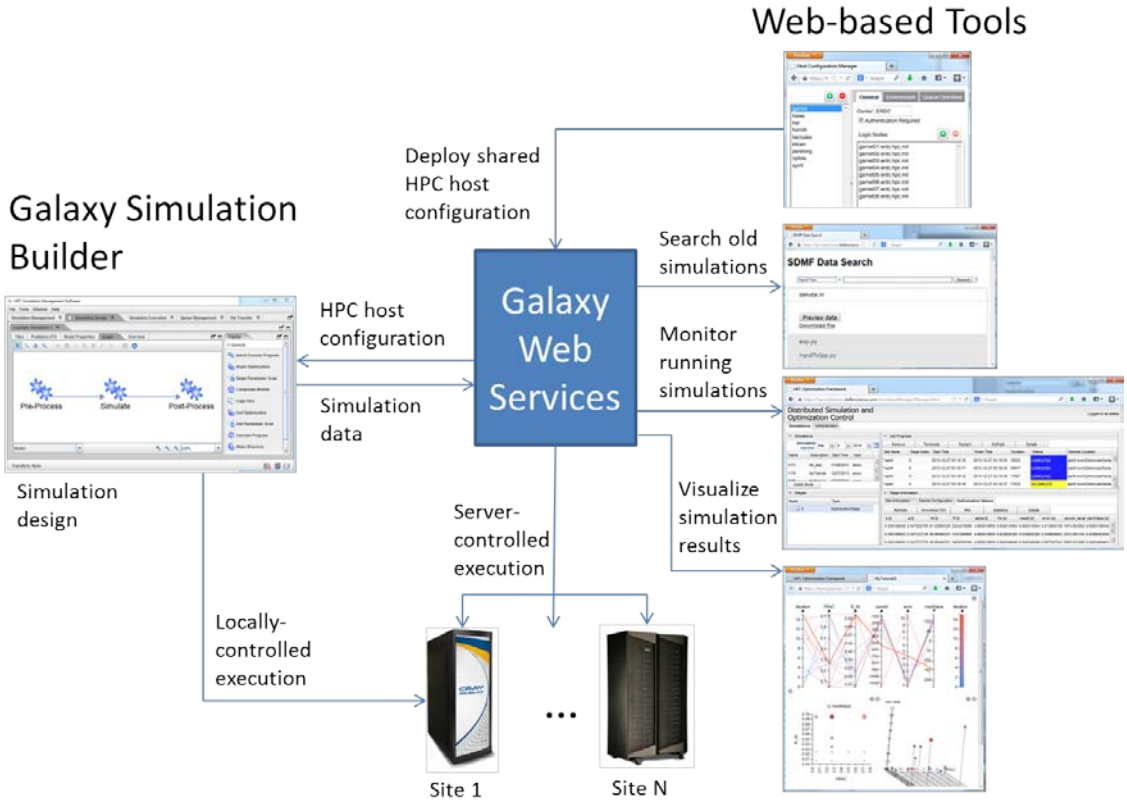

Figure 2: Galaxy Overview

In order to make Galaxy flexible and maximally portable, much of the core Galaxy functionality is implemented as web services. Using HTTP for communication allows Galaxy to operate on HPC systems where other ports and protocols are blocked due to security concerns. The loose coupling among components also allows Galaxy to be deployed in a variety of configurations, as shown in Figure 3. Finally, web services offer the flexibility to implement client functionality in a web browser, in a desktop application, or both.

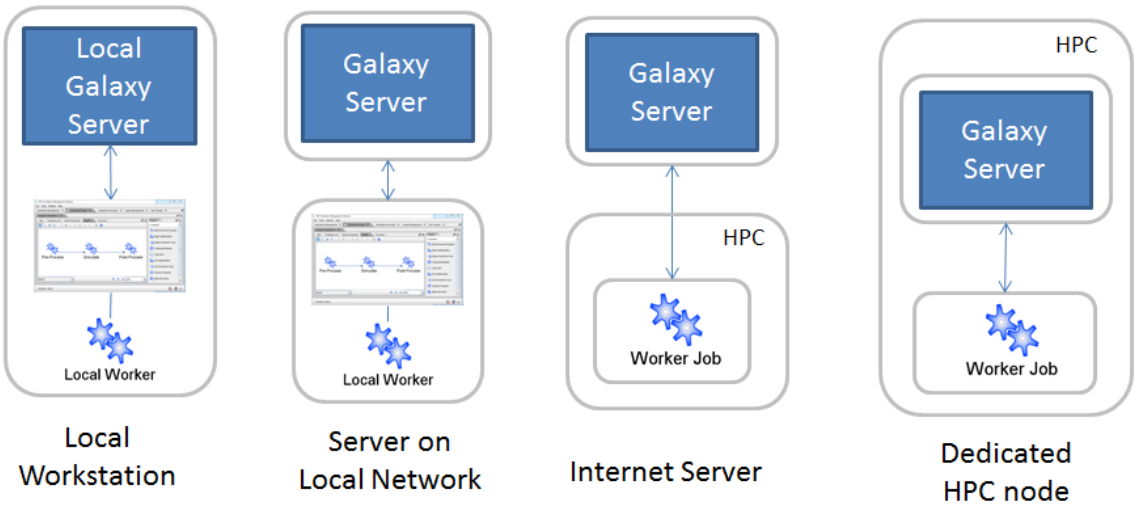

Figure 3: Galaxy Deployment Alternatives

3 Simulation Design

Galaxy users begin by defining a simulation in the main graphical user interface (GUI), a cross-platform Java application called the Galaxy Simulation Builder[1]. Simulations in the Galaxy Simulation Builder are composed of *modules*, each of which represents a logical task such as running a simulation or post-processing its output files. There are also some special modules that represent control logic, such as the beginning and end of optimization phases and parameter scans. This representation of a simulation captures the logical tasks and the data flow between tasks, independent of the details of running the simulation on a particular HPC system. An example of a simulation with an optimization phase is shown in Figure 4.

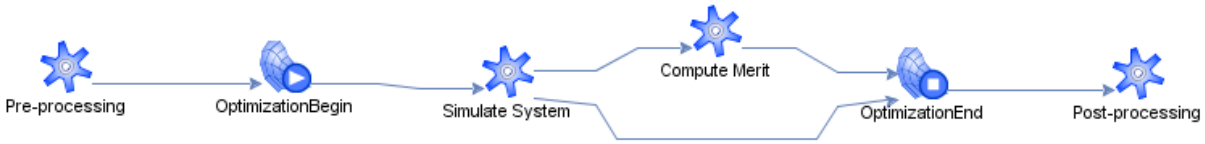

Figure 4: Simulation Graph with Optimization

The default and built-in set of Galaxy Simulation Builder modules can be used to create and execute arbitrary simulations. Extensions such as custom modules, integrated file editors, and preferences pages allow the Galaxy Simulation Builder to be customized for specific domains. Expanding the capabilities of the Galaxy Simulation Builder is as simple as implementing a few Java interfaces and including the resulting library on the class path at runtime.

Perhaps the most important Galaxy Simulation Builder addition is the functionality for adding custom modules for simulation use, which allows the creation of a domain-specific user interface. For example, instead of using a generic executable module to represent an application—in which a command and its arguments are specified as if the user were using the command line—a custom module can be added that displays command-line options as check boxes or even includes a multistep wizard for configuring how the domain-specific application will run. Users can then drop one of these modules into a simulation—complete with the application's familiar icon—and configure it in a natural way. Cooperating modules can automatically configure themselves when connected to each other during design, eliminating much of the error-prone process otherwise associated with creating simulations. Custom modules can also perform additional up-front validation to find configuration problems before the simulation is run. Figure 5 shows an example properties panel for a custom module that was built for the Abaqus finite element package.

[1] Previously it was called the HPC Simulation Management System (HSMS).

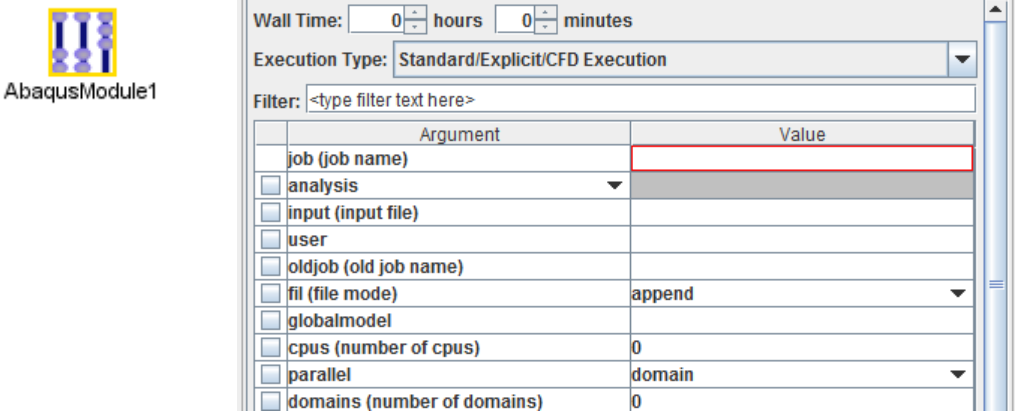

Figure 5: Abaqus icon (left) and Abaqus Custom Module Properties (right)

In addition to simulation editing functionality, the Galaxy Simulation Builder includes several features that speed up the process of interacting with remote HPC systems. The user can connect to a remote system by clicking on its icon and entering login credentials, using either a password or common access card (CAC). Galaxy then manages a secure connection to the remote system and uses the connection to execute remote commands and queries, insulating the user from the command line. For example, a built-in queue management tool shows current PBS queue status information for all connected systems. Another powerful built-in tool, the transfer manager, can transfer files back and forth to any connected system, and supports pausing and resuming of transfers. Figure 6 shows the Galaxy Queues and Transfers tabs.

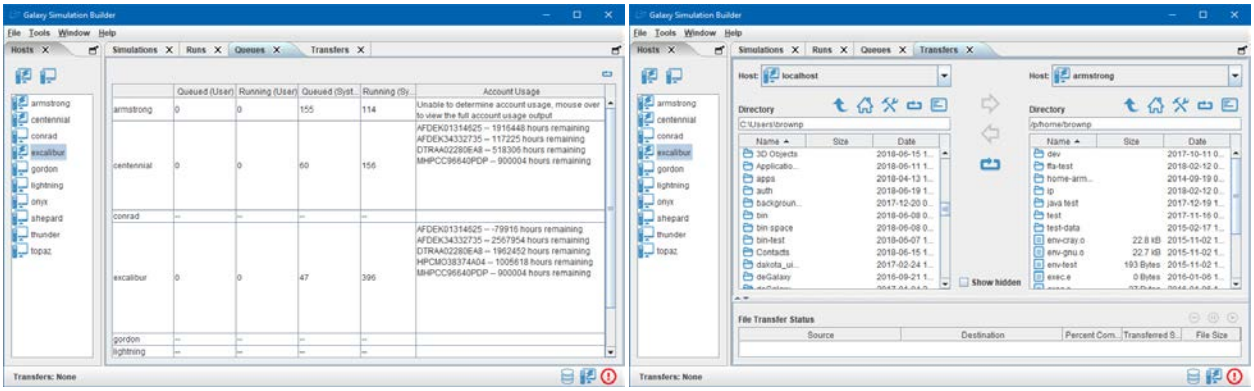

Figure 6: Queue Management and File Transfer Tools

4 Simulation Execution

Galaxy supports two major simulation execution modes: locally-controlled and server-controlled. In the locally-controlled mode, the Galaxy Simulation Builder executes a simulation module-by-module on either the local machine or a single remote system. In server-controlled mode, Galaxy Simulation Builder initiates the simulation by launching worker-spawning jobs, and then a Galaxy web service takes over and drives the simulation. A server-controlled simulation can make use of any number of remote systems, along with the local machine that is running the Galaxy Simulation Builder. Galaxy therefore makes it easy to go from a local simulation to a large optimization that spans multiple systems and runs for weeks or even months.

Galaxy manages the details of executing jobs on remote systems. The Host Manager web service maintains the system information needed to generate PBS job scripts and run message passing interface (MPI) executables on the available HPC systems. The Galaxy Simulation Builder imports that information and uses it for locally-controlled execution or passes it on to the remote systems for server-controlled execution. If necessary, users can customize the host information, as shown in Figure 7. The host information usually will be defined once by an administrator and then shared among a group of users. Users are thus enabled to focus on the high-level logical data flow of their simulation and are shielded from site-specific execution details.

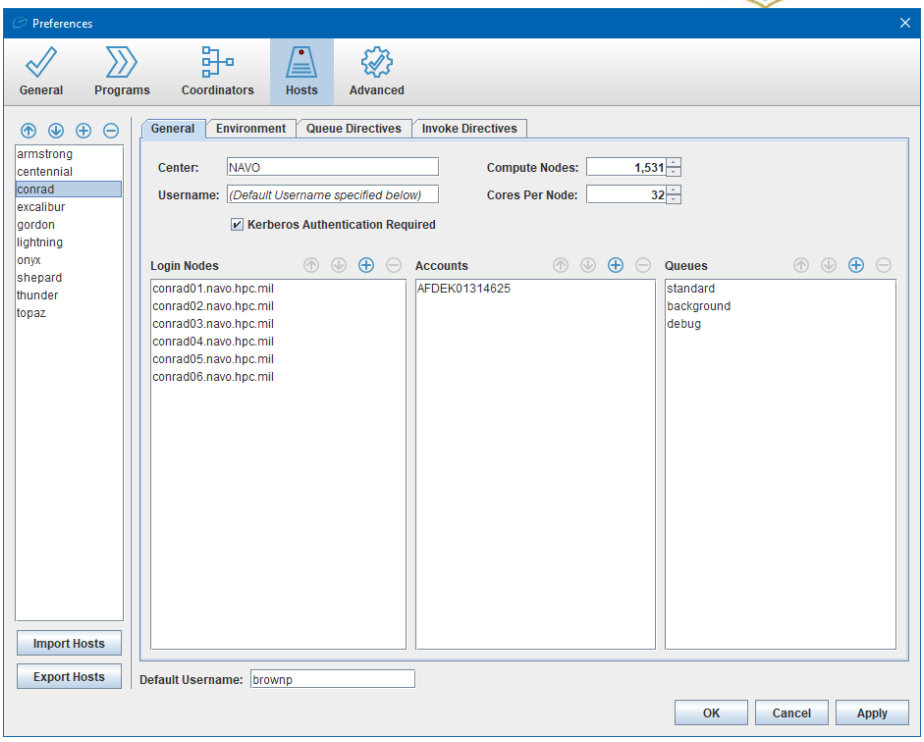

Figure 7: Editing Host Configuration in the Galaxy Simulation Builder

After launching a server-controlled execution on remote machines, the user can close the Galaxy Simulation Builder and allow the Galaxy server to work independently. To check on the simulation process or terminate a simulation, the user can open the web-based Simulation Manager tool, shown in Figure 8. This tool tracks the status of each site along with each PBS job submitted during the simulation. It displays important job details, including error messages and exit codes for any failed jobs. If the simulation contains an optimization or uncertainty quantification (UQ) phase, the Simulation Manager also shows the input and output variables for each evaluated point.

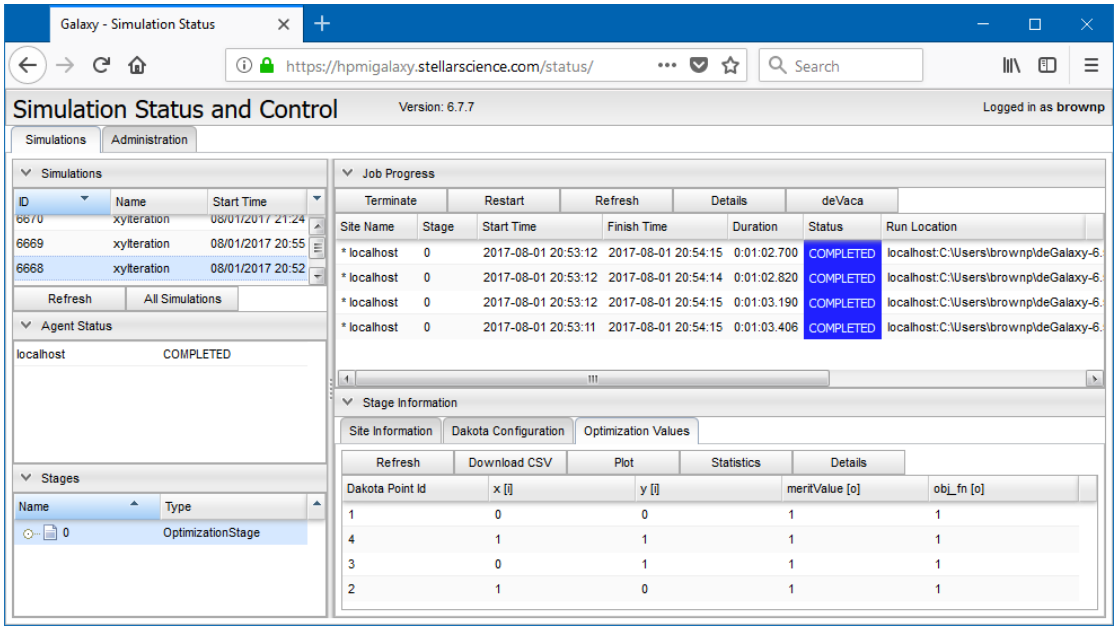

Figure 8: Simulation Manager

5 Optimization and Uncertainty Quantification Capabilities

Galaxy supports a broad range of optimization and UQ algorithms, provided by the Design Analysis Toolkit for Optimizing Terascale Applications (DAKOTA) system developed at Sandia National Laboratories. Galaxy's internal architecture treats DAKOTA as a "point iterator"—a black box that submits points to be evaluated and then consumes the results. The Galaxy server itself is agnostic about optimization algorithm details; it simply maintains a point evaluation queue, dispatches points, and collects the results.

To run a DAKOTA optimization within Galaxy, the user must provide a DAKOTA problem input file, a script to assign input variables, and a merit (objective function) script. The DAKOTA problem input file defines the independent variables and optimization constraints and specifies the optimization algorithm to use. During an optimization run, Galaxy translates each point's input variable assignments into JavaScript object notation (JSON), and the user's input variable script must read the JSON and apply those assignments to the simulation. The objective function is completely under the user's control, and it can depend on any simulation output.

DAKOTA contains many types of optimization algorithms, including algorithms for global, local, multi-objective, and reliability-based optimization. It can also handle both discrete and continuous design variables. Among current Galaxy users, the most popular optimization method for continuous variables is the asynchronous parallel pattern search (APPS), because this method is robust and has internal parallelism that Galaxy can exploit. For problems with discrete variables, non-gradient-based methods like the coliny_ea genetic algorithm are available.

DAKOTA also supports uncertain variables, including both "aleatory" random variables that follow a known distribution and "epistemic" variables that can only be described by a range. DAKOTA can perform Monte Carlo analyses and reliability analyses using the mean-value method, along with a few other types of UQ analyses.

Galaxy can also use DAKOTA to generate and execute experimental designs and parameter studies. However, Galaxy also has its own native parameter sweep capability that includes a customizable reduction step and works with locally-controlled simulations that do not utilize either DAKOTA or the Galaxy server.

6 Result Visualization

Galaxy includes a web-based interactive visualization capability that enables the user to browse the results of an optimization or UQ and gain insight into system responses. Figure 9 shows sample results from an optimization run. The left-most diagram is a parallel coordinate plot that shows all design variables and system responses. Axes on the parallel coordinates plot can be selected to create 2D or 3D scatter plots showing the value of the objective and the relationship of design variables to one another. In addition, ranges on the parallel coordinates plot can be selected to highlight or remove data from the scatter plots, enabling focus to be directed to more important data points.

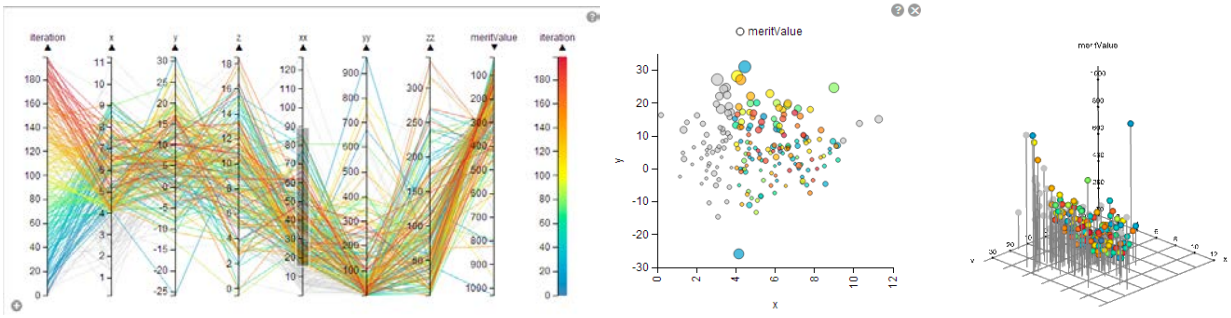

Figure 9: Web-based Interactive Visualization

As a complement to the interactive visualization capabilities, the Simulation Manager can display descriptive statistics for any optimization or UQ run, including a covariance matrix and pairwise estimates of mutual information. The tool can also export the data in comma separated value (csv) format for easy import into other analysis tools.

7 Security, Efficiency, and Scalability

While many codes can scale across the nodes of an HPC system, Galaxy's server-controlled execution mode distributes computations efficiently across multiple HPC sites. This unique ability to span sites raises several special design issues that the Galaxy architecture addresses in novel ways.

Security concerns determine much of the overall structure of the system. Galaxy's web services do not store user credentials and cannot initiate connections to remote hosts. Therefore, in order to communicate with multiple HPC sites, the Galaxy web services

must run on an external server that all the sites can reach. Furthermore, the service that coordinates remote workers must provide a polling-based interface because it cannot send asynchronous outgoing messages.

In light of those security constraints, launching a server-controlled execution run is a three-step process. First, Galaxy Simulation Builder obtains a run-specific secret key from the Galaxy server. Next, Galaxy Simulation Builder uses its internally-managed secure connections to launch a persistent execution agent on each remote site, passing along the secret key. Finally, the execution agents begin polling the Galaxy server for work to execute, using the secret key for identification. Both Galaxy Simulation Builder and the remote agents can verify the identity of the Galaxy server by checking its public certificate via the standard HTTPS protocol.

Although large HPC sites generally provide a batch scheduling system such as PBS for internal load management, efficient multisite simulation requires a top-level layer of load balancing as well. To avoid simulation bottlenecks caused by overloaded sites, Galaxy provides a unique load-balancing capability based on just-in-time point assignment. The remote agent on each site queues worker jobs when there are points waiting to be evaluated on the Galaxy server, but the workers do not get assigned to a specific point until they begin executing. This approach ensures that optimization algorithms with serial point dependencies are not waiting for a point to be evaluated on a particular site when other sites are free. As a natural consequence of just-in-time assignments, the faster, lightly-loaded sites will get more of the points, speeding up the simulation.

In order to support long-running optimizations across many sites, Galaxy is robust and circumvents both site failures and job failures. For example, if an HPC site goes down for maintenance during a run, Galaxy will simply stop using that site and will direct the points to the other sites. In the case of spurious, unexplained job failures, Galaxy will retry the point several times before considering it an error. In the case of module-level errors, such as programs that crash or exceed their allotted wall time, Galaxy defers the error handling to the optimization engine. DAKOTA can be configured to handle an error in several ways, from simply aborting to attempting to work around the bad point.

Galaxy also includes several efficiency-enhancing capabilities that compensate for quirks of HPC batch scheduling systems. Often HPC systems have a minimal job granularity—typically involving one shared-memory node, which may have up to 64 processing cores. In order to schedule small tasks efficiently, Galaxy packs as many workers as possible onto a single node. Galaxy can also schedule subtasks of an optimization point within a single PBS job. This subscheduling capability is an important part of the just-in-time point assignment scheme because it ensures that points will not have internal evaluation delays due to queued sub jobs. Subscheduling also enables embarrassingly-parallel parameter sweeps to be grouped into one large job, which will often get more favorable scheduling treatment than a set of individual small jobs.

8 Galaxy Capabilities Currently in Development

Galaxy continues to be actively developed during FY18. Current areas of focus include:

- Adding support for cross-site simulations, where particular modules must execute on particular sites and output data is transferred automatically from one site to another.
- Adding new sites/resources dynamically to a running optimization.
- Adding support for nested optimizations and parameter sweeps inside optimizations.
- Integrating the APREPRO preprocessor to easily apply DAKOTA assignments to simulation input files.
- Making usability improvements to the Galaxy Simulation Builder DAKOTA editor.
- Developing continued GUI improvements and features.

9 Conclusion

Galaxy offers a user-friendly and responsive environment for defining simulations, along with a sophisticated back-end for executing simulations efficiently across multiple HPC systems. Galaxy also provides a powerful optimization and UQ suite in a multipurpose tool that can be customized for any application domain. Galaxy has been in active use at AFRL/RD for several years, where it has revolutionized the process of design optimization on the DSRC systems, and the user base continues to grow.

The Galaxy Team is interested in incorporating new capabilities into Galaxy and expanding its use into new application domains. For additional information about Galaxy, please contact galaxy-support@stellarscience.com .