LLNL-TR-642019

LAWRENCE LIVERMORE NATIONAL LABORATORY

Lorenz: Using the Web to Make HPC Easier

J. W. Long

August 7, 2013

Lorenz – Using the Web to Make HPC Easier

Executive Summary

Users of High Performance Computing (HPC) centers such as Livermore Computing have historically required expert knowledge of nearly every aspect of the computer center operation in order to take full advantage of the immense resources available. This knowledge takes months or more to develop, and can make entering the world of HPC intimidating for new users.

The goal of the Lorenz project is to improve accessibility and productivity of HPC users by leveraging modern web technologies. Productivity is improved by increasing access to information and simplifying tedious or difficult tasks. New HPC users are less intimidated because the Lorenz web application suite helps coordinate workflow and facilitate the proper and optimal use of HPC resources.

The Lorenz project has defined three initial phases: a system dashboard aimed at making information about the computing center easy to determine at-a-glance, job management including batch job submission and monitoring, and application portals that will provide a high-level interface for defining simulation input.

A new generation of web technologies has enabled the development of applications with the richness usually associated with standalone applications. A modern standards-based web browser is all that's needed to access the multi-faceted Lorenz application suite.

Lorenz has been under development in the Livermore Computing Division at Lawrence Livermore National Laboratory since the spring of 2010. The Lorenz project name honors the famous mathematician and pioneer of modern chaos theory, Edward Lorenz.

Project Background

In the past five years technologies have emerged that have enabled rich applications to be delivered via the web. Map-based sites, personalized web portals, and online content editors have blurred the boundaries between applications and web applications. The Lorenz project leverages some of the same web technologies and concepts from these sites, and applies them towards the problem of HPC usage.

The need for Lorenz arose from the difficulties and frustrations associated with high performance computing from a user's point of view. Not only do users have to model complex phenomena on state of the art machines, develop scalable algorithms, write efficient code, and debug thousands or millions of threads, but they also have to know about the "mechanics" of HPC – the low level details necessary to use the computer center effectively.

Users must know dozens of commands, utilities, and document locations to effectively run simulations at a computer center. They must answer questions such as: "Which cluster should I run on given that my job needs more memory than usual?" "Which of the four parallel filesystems performs best with that cluster?" "What's the trick for telling the job scheduler to not hold my job until a scratch filesystem comes back online?" "Why is my job stuck in pending mode?" "How do I invoke the runtime parallel debugger on my job?" "What maintenance is planned for this week?" "Who else can see my files, and how much room do I have remaining in my quota?"

These are just a few examples of the hundreds of details an HPC user needs to know before being fully productive at a computing center. Users spend weeks or months learning these details. Computing centers are constantly evolving and expanding, making it that much harder for users to stay on top of the best practices.

Staff members likewise must deal with a myriad of information, both for maintaining the systems and for ensuring users have the latest status data and documentation. Having information at the tip of their fingers about the current status of the center and about individual users allows tier-1 and tier-2 support personnel to provide the best possible customer service.

Project Goals

The main goal of the Lorenz project is to help our HPC users focus on their simulations and science, rather than on the computing center and how to use it. To achieve this, Lorenz works to reduce the time it takes to learn how to use resources effectively, to simplify tedious and repetitive tasks, and to make sure that users have access to critical information when they need it.

The approach we've taken is to employ modern web technologies and write or apply software that encapsulates computing center know-how and policies automatically into the users' workflow. The result is an intuitive, responsive, and aesthetically pleasing interface for the Livermore Computing (LC) high performance machines.

Because the breadth of desktop operating systems in use both inside and outside LLNL spans all flavors of Linux, Mac, and Windows, the decision was made to use exclusively web-based technologies in Lorenz. Recent advancements in JavaScript programming, along with coming innovations in the form of HTML5 and CSS3, made the browser the preferred development platform. The goal is to provide the user with a robust application interface that requires nothing more than a modern, standards-based web browser. This approach not only provides convenient access to Lorenz for internal and external users, but also has significant advantages over conventional applications in terms of rapid deployment and updates.

Project Components

The Lorenz project has defined three initial phases: a system "dashboard" aimed at making information about the computing center easy to determine at-a-glance, job management including batch job submission and monitoring, and application portals which will provide a high-level interface for defining simulation input.

Dashboard

The Lorenz dashboard, tabbed "MyLC" at LLNL, is a collection of individual portal applications (portlets) that provide LC users with access to and information about the most vital components of the HPC center. Information that previously required knowledge of dozens of different commands, files, and web pages is now available in one centralized location. This includes account information, bank and disk usage, quotas, cluster utilization, news items, active jobs on LC systems, and more (see Table 1.)

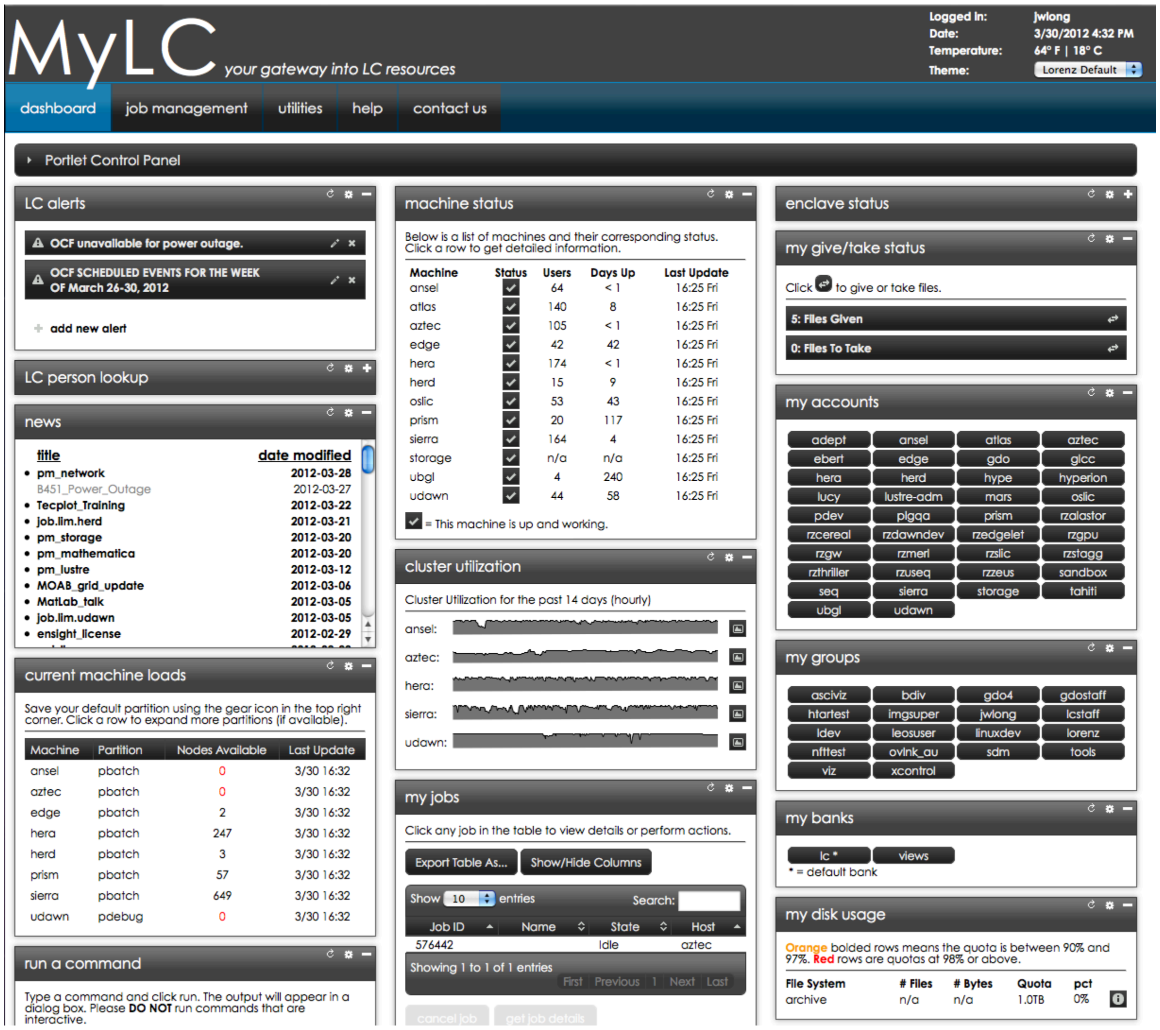
Portlets are completely customizable and personalized to each user. Users can tailor their dashboards to only show information that's most important to them. The design allows individuals to develop portlets and easily incorporate them into the dashboard, and work groups can share portlets amongst themselves.

Portlets currently available to users include:

Portlet Name	Purpose
Archival storage quota	Get information about your HPSS quota
Bank membership	Browse all accounting banks and members
CHAOS update stats	Show information about OS updates on clusters
Cluster utilization	Show hourly utilization for each cluster
Confluence browser	Show a table of confluence spaces to which you have access, with links for getting to attachments, exporting content, etc.
Current machine loads	Show number of nodes available on each cluster
Enclave status	Summary of your HPC Enclave status
LC alerts	Lists topical alerts, pending downtimes, etc.
LC staff away	Show excerpt of the LC staff away calendar
License status	Show status of all licensed software
Login node status	Show availability, load average, etc. of all login nodes
Machine status	Show up/down status, number of users, etc.
My accounts	Show all of the hosts and clusters you have access to
My banks	Show all accounting banks to which you belong
My disk usage	Show your current disk usage on various filesystems
My groups	Show groups to which you belong
My historic cpu usage	Show your usage information for each cluster
My jobs	Show your active and pending batch jobs
My links	List of standard LC and personalized links
My processes	List processes you have running on login nodes
News	Review all active news items
Printer queues	Locate printers and view printer queues
Scratch file system usage	Show status of various scratch filesystems

Table 1: The MyLC dashboard portlets

Figure 1 below shows an example of the MyLC Dashboard. Individual portlets have preferences which allow a user to, for example, view information for all clusters or only the ones to which they have access.



MyLC your gateway into LC resources

Logged In:	jwlong
Date:	3/30/2012 4:32 PM
Temperature:	64° F | 18° C
Theme:	Lorenz Default

dashboard | job management | utilities | help | contact us

Portlet Control Panel

LC alerts

- **OCF unavailable for power outage.**
- **OCF SCHEDULED EVENTS FOR THE WEEK OF March 26-30, 2012**

add new alert

LC person lookup

news

title	date modified
pm_network	2012-03-28
B451_Power_Outage	2012-03-27
Tecplot_Training	2012-03-22
job.lim.herd	2012-03-21
pm_storage	2012-03-20
pm_mathematica	2012-03-20
pm_lustre	2012-03-12
MOAB_grid_update	2012-03-06
MatLab_talk	2012-03-05
job.lim.udawn	2012-03-05
ensight_license	2012-02-29

current machine loads

Save your default partition using the gear icon in the top right corner. Click a row to expand more partitions (if available).

Machine	Partition	Nodes Available	Last Update
ansel	pbatch	0	3/30 16:32
aztec	pbatch	0	3/30 16:32
edge	pbatch	2	3/30 16:32
hera	pbatch	247	3/30 16:32
herd	pbatch	3	3/30 16:32
prism	pbatch	57	3/30 16:32
sierra	pbatch	649	3/30 16:32
udawn	pdebug	0	3/30 16:32

run a command

Type a command and click run. The output will appear in a dialog box. Please **DO NOT** run commands that are interactive.

machine status

Below is a list of machines and their corresponding status. Click a row to get detailed information.

Machine	Status	Users	Days Up	Last Update
ansel	✓	64	< 1	16:25 Fri
atlas	✓	140	8	16:25 Fri
aztec	✓	105	< 1	16:25 Fri
edge	✓	42	42	16:25 Fri
hera	✓	174	< 1	16:25 Fri
herd	✓	15	9	16:25 Fri
oslic	✓	53	43	16:25 Fri
prism	✓	20	117	16:25 Fri
sierra	✓	164	4	16:25 Fri
storage	✓	n/a	n/a	16:25 Fri
ubgl	✓	4	240	16:25 Fri
udawn	✓	44	58	16:25 Fri

✓ = This machine is up and working.

cluster utilization

Cluster Utilization for the past 14 days (hourly)

my jobs

Click any job in the table to view details or perform actions.

Export Table As... | Show/Hide Columns

Show 10 entries Search:

Job ID	Name	State	Host
576442		Idle	aztec

Showing 1 to 1 of 1 entries

First Previous 1 Next Last

cancel job | get job details

enclave status

my give/take status

Click to give or take files.

- **5: Files Given**
- **0: Files To Take**

my accounts

adept	ansel	atlas	aztec
ebert	edge	gdo	glcc
hera	herd	hype	hyperion
lucy	lustre-adm	mars	oslic
pdev	plgqa	prism	rzalastor
rzcereal	rzdawndev	rzedgelet	rzgpu
rzgw	rzmerl	rzslic	rzstagg
rzthriller	rzuseq	rzzeus	sandbox
seq	sierra	storage	tahiti
ubgl	udawn		

my groups

asciviz	bdiv	gdo4	gdostaff
htartest	imgsuper	jwlong	lcstaff
ldev	leosuser	linuxdev	lorenz
nfttest	ovlnk_au	sdm	tools
viz	xcontrol		

my banks

lc * | views

* = default bank

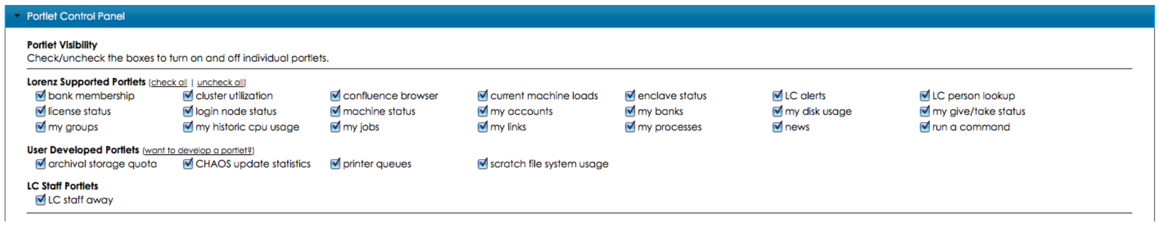
my disk usage

Orange bolded rows means the quota is between 90% and 97%. Red rows are quotas at 98% or above.

File System	# Files	# Bytes	Quota	pct
archive	n/a	n/a	1.0TB	0%

Figure 1. The MyLC Dashboard.

The dashboard consists of a set of individual portlets that can be collapsed, moved around using drag-and-drop, or completely hidden using the Portlet Control Panel, seen in Figure 2. Each portlet is a standalone mini-application with its own data sources and display options. Many portlets allow users to drill down for additional information.



Portlet Control Panel

Portlet Visibility
Check/uncheck the boxes to turn on and off individual portlets.

Lorenz Supported Portlets (check all | uncheck all)

✓ bank membership	✓ cluster utilization	✓ confluence browser	✓ current machine loads	✓ enclave status	✓ LC alerts	✓ LC person lookup
✓ license status	✓ login node status	✓ machine status	✓ my accounts	✓ my banks	✓ my disk usage	✓ my give/take status
✓ my groups	✓ my historic cpu usage	✓ my jobs	✓ my links	✓ my processes	✓ news	✓ run a command

User Developed Portlets (want to develop a portlet?)

✓ archival storage quota	✓ CHAOS update statistics	✓ printer queues	✓ scratch file system usage

LC Staff Portlets
✓ LC staff away

Figure 2. The dashboard's Portlet Control Panel

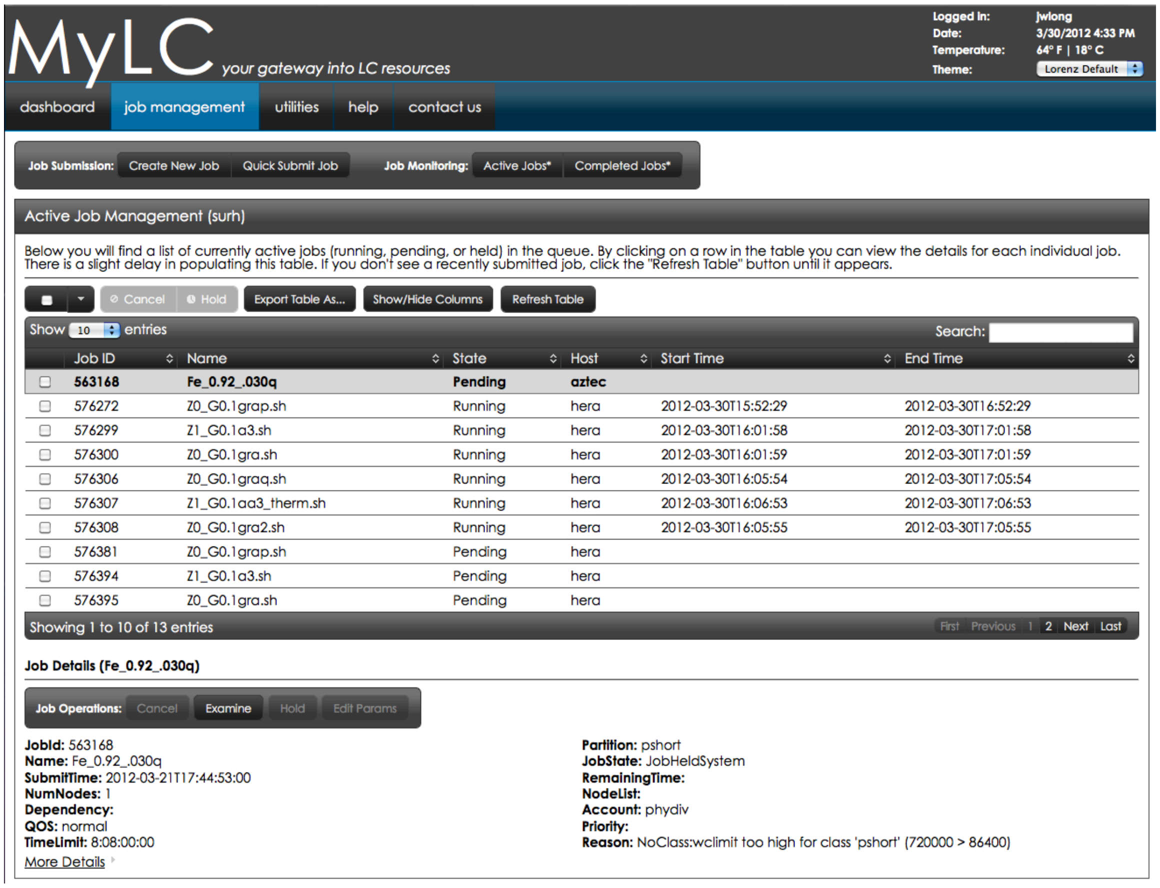
The portlet control panel allows the user to control which portlets are displayed. Portlets are grouped by type: supported (available for everyone), user-developed (available to individual users or workgroups), and staff (available to computing center staff only.)

Job Management

The second major phase of Lorenz was built on the idea that HPC centers are managed around batch processing, and "jobs" (or "runs") progress through several discrete stages: setup, submission, queuing/waiting, running, completion, analysis, and archiving. Lorenz aims to ultimately tie all of these pieces together, including integration with the dashboard and simulation portals. In addition to helping guide users through the process of defining batch parameters, monitoring job progress, and interacting with jobs and job output, Lorenz aims to promote access to additional tools and capabilities which to this point have been limited to power users. Ultimately Lorenz will guide users in choosing the appropriate resources, such as cluster and file system, given the requirements of a job.

Figure 3 shows a snapshot of the Job Management application in Lorenz. This interface allows a user to quickly get a summary of all of his or her pending and running jobs, and to interact with those jobs in a variety of ways – sending them signals, viewing output files, generating stack traces, and modifying batch parameters for jobs that have yet to start.

Lorenz also facilitates creating new batch jobs. The interface helps guide the user towards correctly setting batch parameters by using knowledge of the cluster hardware and batch partitions, the user's banks, and so on.



Logged In: jwlong
Date: 3/30/2012 4:33 PM
Temperature: 64° F | 18° C
Theme: Lorenz Default

Active Job Management (surh)

Below you will find a list of currently active jobs (running, pending, or held) in the queue. By clicking on a row in the table you can view the details for each individual job. There is a slight delay in populating this table. If you don't see a recently submitted job, click the "Refresh Table" button until it appears.

Show 10 entries Search:

	Job ID	Name	State	Host	Start Time	End Time
☐	**563168**	**Fe_0.92_.030q**	**Pending**	**aztec**		
☐	576272	Z0_G0.1grap.sh	Running	hera	2012-03-30T15:52:29	2012-03-30T16:52:29
☐	576299	Z1_G0.1a3.sh	Running	hera	2012-03-30T16:01:58	2012-03-30T17:01:58
☐	576300	Z0_G0.1gra.sh	Running	hera	2012-03-30T16:01:59	2012-03-30T17:01:59
☐	576306	Z0_G0.1graq.sh	Running	hera	2012-03-30T16:05:54	2012-03-30T17:05:54
☐	576307	Z1_G0.1aa3_therm.sh	Running	hera	2012-03-30T16:06:53	2012-03-30T17:06:53
☐	576308	Z0_G0.1gra2.sh	Running	hera	2012-03-30T16:05:55	2012-03-30T17:05:55
☐	576381	Z0_G0.1grap.sh	Pending	hera		
☐	576394	Z1_G0.1a3.sh	Pending	hera		
☐	576395	Z0_G0.1gra.sh	Pending	hera		

Showing 1 to 10 of 13 entries

Job Details (Fe_0.92_.030q)

JobId: 563168
Name: Fe_0.92_.030q
SubmitTime: 2012-03-21T17:44:53:00
NumNodes: 1
Dependency:
QOS: normal
TimeLimit: 8:08:00:00

Partition: pshort
JobState: JobHeldSystem
RemainingTime:
NodeList:
Account: phydiv
Priority:
Reason: NoClass:wclimit too high for class 'pshort' (720000 > 86400)

Figure 3. The Lorenz Job Management application.

Simulation Portal

The third major phase of Lorenz is the development of an application portal, which involves teaming with application code groups to provide a simple graphical interface for their simulations. This, in turn, seamlessly ties into job submission and monitoring. Users will be able to directly define simulation parameters and launch applications using a powerful web front-end. This capability is targeted both at internal LC users as well as potential new users of the resources at a future Livermore Valley Open Campus computing center.

Because many complex multi-physics codes can have thousands of inputs, providing a general GUI for building an input deck from scratch is impractical. However, most users of large applications start with an existing input file, and make some number of small changes to it. The application portal will feature a process that combines trusted and validated input files with a GUI that guides the user toward making the relatively minor changes needed for their particular problem. The input files will be in the form of templates, containing most of the data needed for a particular problem along with parameterization of changeable inputs.

For example, to model the crushing of an aluminum can, more than 99% of the simulation input parameters would not need to be changed from one run to the next. One might want to change the dimensions of the can, or the initial forces on it, but this can be specified with a very simple form without exposing the thousands of other parameters that are necessary to provide the simulation with the information it needs in order to run.

Utilities

In addition to the three core components of Lorenz, we also provide a collection of web tools to simplify every day tasks within LC. These utilities include things like file transfers, diagnostics, job queue statistics, and more. Additionally, through fine-grained permission control, we are able to expose a set of administration tools to manage Lorenz.

Lorenz Architecture

As illustrated in Figure 4, Lorenz utilizes a client-server architecture. On the client side the vehicle typically is a web browser, although it could be any application. The client makes calls via the Lorenz REST API, or LORA, which provides abstractions for computing center resources, and simplifies interactions with data, processes, and jobs within the center.

The web server receives and executes these requests, sends commands and queries to the computing clusters as necessary, takes the raw data from the queries, massages the data as needed, and returns it to the client application in JSON format. Data retrieved from the clusters can be persistent (e.g., files or databases) or computed on the fly (e.g., batch queue commands).

Critical to this process is that the web server is tightly integrated with the rest of the computing center, allowing us to securely and efficiently operate on behalf of the user on any cluster. The integrated Kerberos security environment enables Lorenz to provide this capability.

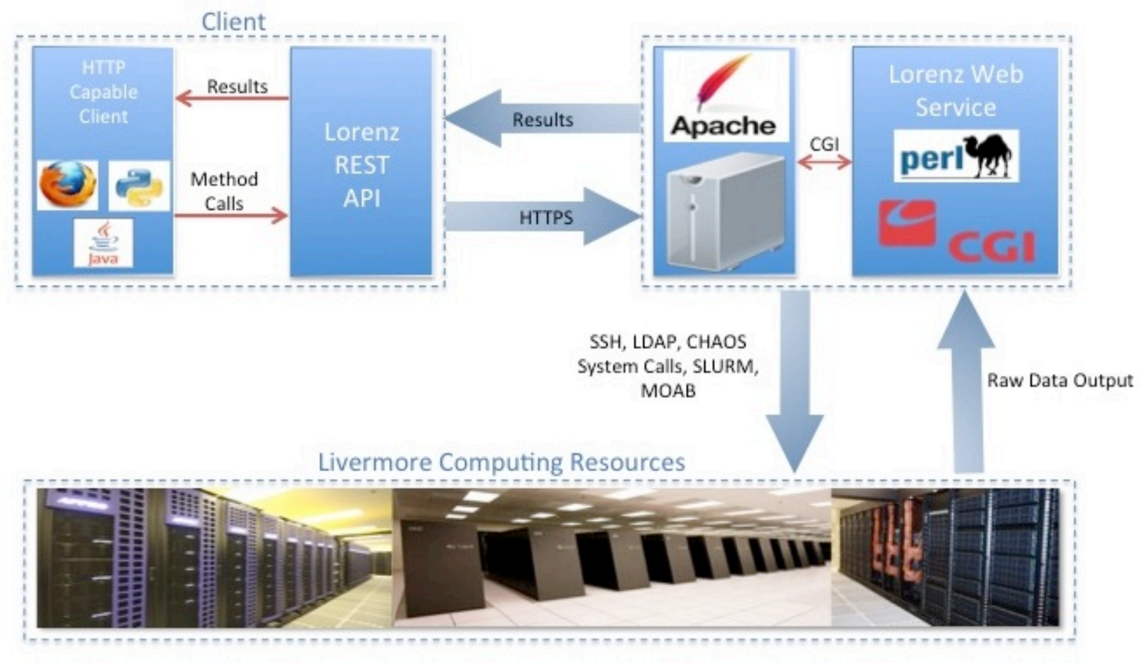


Figure 4: Lorenz client-server architecture at Livermore Computing

Server Strategies

RESTful API

REST stands for Representational State Transfer, and is the predominant architecture style for designing web-based client-server applications. It relies on a stateless, cacheable communications protocol, and is a lightweight alternative to mechanisms like RPC (Remote Procedure Calls) and heavyweight web services such as SOAP (Simple Object Access Protocol) and WSDL (Web Services Description Language.)

The Lorenz RESTful API, or LORA, was modeled after the NERSC Web Toolkit (NEWT) developed at Lawrence Berkeley National Lab. LORA and NEWT are web services that allow one to access and interact with computing resources through a simple RESTful API.

After discussions with our colleagues at NERSC we decided to adopt their API specification and implement it within our own LC computing center. The differences in security architectures and environments between the two sites made sharing server code unfruitful, but by standardizing on the actual API specification it made sharing of higher level client code possible. Towards that goal, NERSC has spearheaded the W3C HPC Web working group, aimed at defining a standard web services API for HPC [http://www.w3.org/community/hpcweb/wiki/Main_Page].

Web Server Environment

The web server environment used by Lorenz at LLNL is a critical element to the success of the project. The web server allows applications to operate on behalf of the user in a seamless and efficient manner. This enables developers to create powerful applications that can provide process improvements and simplify workflow for our users.

The web server is integrated into the security infrastructure of the computing center. Authenticating as an LC user to the web server creates security credentials which are used when that user invokes web services. The Apache suexec module causes the CGI processes to run as the authenticated user, and local enhancements guarantee those processes have access to credentials required for the user to connect to clusters or use other services such as LDAP (Lightweight Directory Access Protocol.) Take for example when submitting a job through the Lorenz job management system – the REST endpoint for job submission will execute a command on a remote cluster via an SSH execute line launched on the web server. This is executed as the user and with his or her credentials and access permissions.

API Overview

Lorenz has numerous APIs and libraries focused on different use cases, as shown in Table 2.

API	Language	Used By	Purpose

LORA	Any	Application Developers	The actual RESTful API, LORA, can be invoked directly from any language that supports web interactions. Users locally have developed Windows and Python applications that invoke LORA directly.
Lorenz::Host Lorenz::User Lorenz::Group Lorenz::*	Perl	Lorenz Developer	Individual Perl modules for accessing specific Lorenz functionality. These modules return Perl native data structures rather than the JSON envelope returned by the REST equivalents. Developers implementing Lorenz internals should use this.
Lorenz::REST::*	Perl	Lorenz Developer	The specific Perl modules invoked by the REST handler, these primarily interact with the lower level Lorenz::Host|User|... modules, but provide additional structure such as a return envelope showing output, error, and status. Data returned is in JSON format.
Lorenz::Simple	Perl	Tool Developers (cluster tools)	Aggregated Perl module for tool developers that makes it simple to access a variety of Lorenz functionality with minimal effort. This should be the default API to use for tools that will run on the clusters.
Lorenz::API	Perl	Tool Developers (remote tools)	An LWP-based Perl module that provides a high level API to the LORA web services. This is for developers creating applications to be run on remote hosts, although it can be used on clusters. Requires authentication since the library acts as a browser equivalent.
Lorenz.js	JavaScript	Web App Developers (LC-hosted apps)	A high-level library that provides access to the LORA REST services to web applications. The calling sequence and function names match the Lorenz::API Perl module.
Lorenz-JSONP.js	JavaScript	Web App Developers (remote-hosted apps)	A high-level library that provides access to the LORA REST services to web applications. The calling sequence and function names match the Lorenz::API Perl module. This version uses JSONP to deal with cross-site scripting issues. (In progress.)

Table 2: The Lorenz APIs

APIs in other languages are not difficult to implement. More detail on these APIs are available in the technical report, “Lorenz APIs and REST Services.”

Data Generation and Retrieval

Livermore Computing puts particular emphasis on retrieving data for the Lorenz REST endpoints in a timely and minimally intrusive manner. LC has several thousand users, and if several hundred are using the MyLC dashboard at any given time the load on services from concurrent requests could impact performance. For example, directly querying active jobs across all clusters might take 10-60 seconds of real time, and having dozens of users making these requests simultaneously would cause our resource management database server to severely bog down. Add to that our goal to have the entire dashboard load within two-three seconds, and it becomes clear we need to pre-fetch and cache as much data as possible.

Of the approximately 30 sources of information used to populate the MyLC dashboard, two-thirds of them come from data that are aggregated and prepared by Lorenz cron jobs running on the compute clusters and stored either in a Redis key-value store or in a globally-mounted file system in flat files. This information retrieval is nearly instantaneous, and allows the dashboard to load extremely quickly.

The downside of this approach is that some data may be slightly out of date. Critical data are collected in a central place every 30-60 seconds, which we found to be a good compromise between timeliness and efficiency. Infrequently modified data are collected hourly or daily, depending on the source. For applications that require up-to-date information, critical LORA endpoints support parameters that will cause live data to be retrieved.

Client Strategies

Web Browser versus Heavyweight Application

A key design goal from the outset was to require nothing more from our users' desktop environments than a modern web browser. Users of Livermore Computing resources employ a diverse range of desktop environments, ranging from all flavors of Linux to Windows and OS X. Deploying a thick client across all major OSes and versions has the primary drawback of difficulty in doing coordinated and timely updates. It was an obvious choice to focus on web-based technologies for the Lorenz client. In addition to rapid and easy deployment of new versions, we've found web development to be easier and faster than comparable thick Java applications.

Web Languages and Strategies

The Lorenz client is written mainly with HTML, CSS, JavaScript (JS), and jQuery. For UI components and widgets we use the jQuery UI library. Additionally, we make heavy use of the jQuery UI widget factory design pattern.

The client takes advantage of a wide array of different optimization techniques such as: image sprites, JS/CSS compression, JS bootstrapping, dependency injection, lazy loading, contextual jQuery, XHR caching, browser caching, and server side rendering. Extensive benchmarking and instrumentation led to these optimizations as well as server-side optimizations, ultimately improving page-load performance by 2-4X.

The implementation of the dashboard portlets use a prototypal inheritance model that is facilitated by the jQuery UI widget factory. It is through this inheritance model that we are able to allow developers to easily develop new portlets by simply inheriting from our portlet base class. This follows the key Lorenz principle of using object-oriented design and development whenever possible.

To provide flexibility and convenience when creating dynamic pages, client side templating is used throughout on the majority of our applications. We use jQuery templates as our templating language. jQuery templates provide ways of creating chunks of HTML and applying an object of attributes to them via the jQuery DOM manipulation methods.

SIML – Simulation Markup Language

The Lorenz simulation portal makes use of forms written in the Simulation Input Markup Language (SIML). These forms are interpreted into HTML format and presented in the user's web browser (see Figure 5). After completing and submitting the form, simulation input files are created on an LC resource, and a batch job is submitted. The user is directed to the Lorenz job management element to monitor progress, as well as to examine the results of the completed simulation.

SIML extends standard HTML with custom widgets and error checking suitable for building simulation input files. This approach takes advantage of the powerful existing mark-up capabilities of HTML, and extends it with a variety of form inputs and validations necessary to create a proper simulation input file.

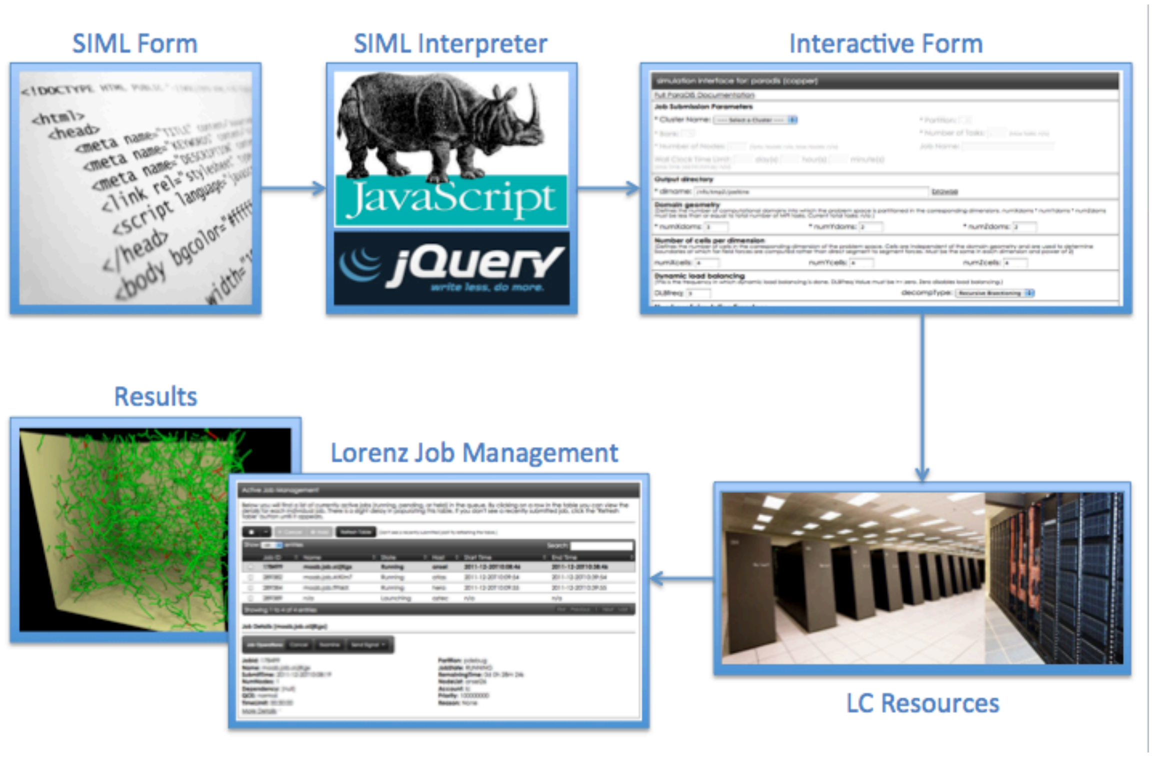


Figure 5: The Simulation Portal workflow

Lorenz Development and Usage

In January of 2011 Lorenz was deployed in beta mode on LC computers. After several months of user and staff feedback, version 1.0 was released on May 17, 2011. User reaction to Lorenz has been very positive. Usage has steadily grown with each release.

The convenient and customizable access to information and services through the MyLC Dashboard has appealed to users and computing center staff alike. This rich and polished user experience helped draw in users, and encouraged them to try other elements of the Lorenz suite of HPC tools.

An important part of the user adoption of any tool is the interface. A combination of development and design skills are necessary for creating a successful web application. An attractive and intuitive interface provides the initial draw, but relevant and powerful capabilities are what retain users for the long term.

Future Work

Users and staff members have provided a long list of additional capabilities and types of information they would like to see in Lorenz. Some of the proposed extensions include

- A center-wide alert system that allows users to subscribe to specific types of events (batch job completion, host downtime, news items).
- A calendar that tracks system modifications and other significant events on all clusters.

- Enhancements to the job management tool to enable more options during job launch and more capabilities for interacting with running jobs.
- Additions to the application portal that help users choose the appropriate resources, such as cluster and file system, given the requirements of a job.

Conclusions

User response to Lorenz has confirmed our belief that there is a place for web-based applications in the HPC world. Rapid evolution of web technologies has enabled the development of full-featured and robust applications. Because users today are immediately at ease using a web interface, there was quick adoption of the Lorenz suite of tools.

The Lorenz tools allow HPC users to focus on their simulations and science, rather than on the computing center and how to use it. These tools reduce the time it takes to learn how to use resources effectively, simplify tedious and repetitive tasks, and make sure that users have access to critical information when they need it.