Bridging VisTrails Scientific Workflow Management System to High Performance Computing

Jia Zhang[1], Petr Votava[2], Tsengdar J. Lee[3], Owen Chu[1], Clyde Li[1], David Liu[1], Kate Liu[1], Norman Xin[1], Ramakrishna Nemani[2]
[1]Carnegie Mellon University – Silicon Valley, USA
[2]NASA Ames Research Center, USA
[3]Science Mission Directorate, NASA Headquarters, USA
jia.zhang@sv.cmu.edu, petr.votava@nasa.gov, tsengdar.j.lee@nasa.gov, rama.nemani@nasa.gov

***Abstract*—**NASA Earth Exchange (NEX) is a collaboration platform whose goal is to accelerate Earth science research, by leveraging NASA's vast collections of global satellite data together with access to NASA's High-End Computing (HEC) facilities. NEX also aims to facilitate the sharing of experimental results as well as scientific processes (workflows) with the Earth science community through integration with VisTrails workflow management system. While VisTrails is used internally, it is not easily accessible from remote computers without directly logging into the NASA HEC systems through two-factor authentication and a bastion host. This paper describes the initial design of an extensible architecture that facilitates easier workflow interaction on NEX, by enabling users to develop and execute workflows in a supercomputing environment directly from their local VisTrails installation. This architecture helps domain scientists seamlessly leverage distributed computing and storage resources and it is potentially applicable to other scientific workflow management software. We further describe the architecture of the VisTrails-HEC plugin (as well as the VisTrails-Amazon plugin) and the implementation of a working prototype to demonstrate the feasibility of our solution.

I. Introduction

The data volumes accumulated by NASA's Earth observing satellites and climate models continues to grow rapidly. Analyzing such a vast amount of data requires significant computing power and data storage, which are usually not available to most research labs and individual researchers. In order to help scientists conduct research and analysis on large Earth science datasets, the NASA Earth Exchange (NEX) [1] project has been established. NEX combines state-of-the-art supercomputing, Earth system modeling, remote sensing data from NASA and other agencies, and a scientific social networking platform to deliver a complete work environment in which users can explore and analyze large Earth science data sets, run modeling codes, collaborate on new or existing projects, and share their results with the community. As NEX provides centralized access, not only does data not have to be moved back and forth to scientists' local places, but also data analysis procedures can be operated remotely on the NEX

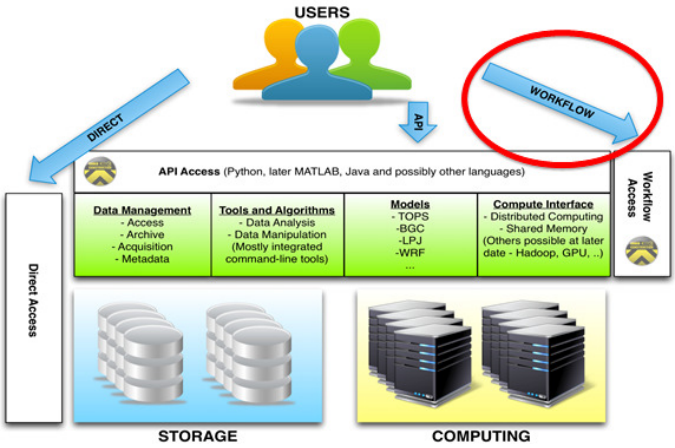

Fig. 1 Different ways of accessing NEX components.

by leveraging NEX's computing power. As Fig. 1 illustrates, NEX offers workflows as one way to access the super-computing resources.

Apart from access to data and computing, as shown in Fig. 1, NEX accumulates a set of user-contributed Earth science models, analysis tools and software utilities in order to promote software re-use and accelerate scientific research. Workflow management tools, such as VisTrails [2], have been used to help researchers to create workflows [3, 4] that define the steps in the scientific process and provide a foundation for repeatability, transparency and software re-use. While NEX components are accessible through different ways to support better compatibility with legacy software as depicted in Fig. 1, the workflow components are most significant, because they are key in accelerating research through science re-use with easy extensibility.

Because of the large volumes of data and complexity of the models involved in research and analysis on NEX, high computing power is typically required to conduct the experiments in a reasonable amount of time [5]. In order to leverage NASA's computing capabilities, NEX has been part of the NASA High-End Computing Capability (HECC) project. HECC has constructed a world-class supercomputing and mass storage environment for conducting large-scale modeling, simulating, and analysis to answer NASA's complex science and engineering questions [6]. As of December 2012, over 11,776 nodes are running at the NASA Ames HEC center on the Pleiades

978-0-7695-5024-4/13 $26.00 © 2013 IEEE
DOI 10.1109/SERVICES.2013.64

supercomputer, interconnected with an InfiniBand (IB) network in a hypercube topology [7].

As Critchlow and Chin [8] indicated, however, there usually exists a gap between workflow design environment and workflow execution on supercomputers. Geoscientists access the HEC environment through a two-factor (SSH+RSA) authentication mechanism. They log into front-end nodes and issue jobs through a scheduler to compute nodes [6]. On the other hand, scientific workflow tools like VisTrails require that all procedures run either locally or as remote services. Although it is possible to run VisTrails locally on the Pleiades system, it requires a number of additional steps. Additionally, Pleiades is part of a secure environment that does not allow access using web services. More critical, it is not well suited for interactive workflow design and execution.

To bridge the gap between scientific workflow tools (e.g., VisTrails) and high-end computing, the motivation for our project is to allow scientists to access the HEC supercomputing environment with minimal knowledge of its operational aspects and minimal modification to their workflows. The direct impacts of this effort are multi-fold: (1) facilitating scientists in leveraging NASA supercomputing capabilities from the graphical interface of scientific workflow tools; (2) automating the process of migrating code and computation from development environment to supercomputing environment; (3) allowing scientists to focus on science; and (4) releasing NEX technical staff from a number of user support activities.

In this paper, we present the architecture that enables bridging between local and remote workflow management systems and the implementation of a working prototype to demonstrate the feasibility of our solution. The remainder of the paper is organized as follows. In Section 2, we describe the current HEC infrastructure and VisTrails to explain the technical challenges. In Section 3, we present our architectural design. In Section 4, we present an intelligent scheduling algorithm. In Section 5, we present prototyping system implementation. In Section 6, we discuss related work. In Section 7, we draw conclusions.

II. PROJECT CONTEXT

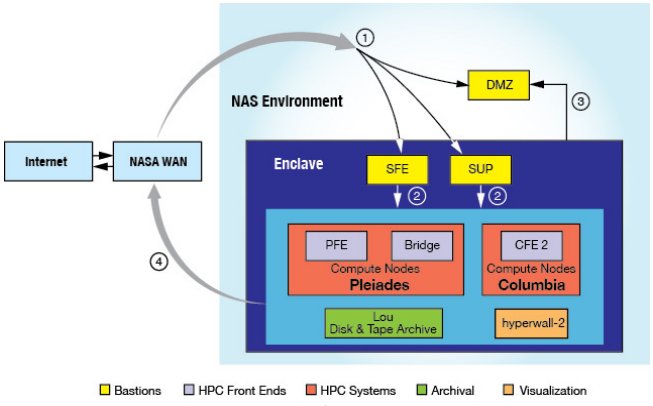

Fig. 2 HEC infrastructure. [5]

Table I. Architectures of computing nodes on Pleiades.

Node Type	# of Nodes	Processors per node	Processor speed	Memory per core
Sandy Bridge	1,728	2 eight-core processors	2.6 GHz	2 GB
Westmere	4,608	2 six-core processors	2.93 GHz or 3.06 GHz	2 GB
Nehalem	1,280	2 quad-core processors	2.93 GHz	3 GB
Harpertown	4,096	2 quad-core processors	3 GHz	1 GB

In this section, we briefly describe the overview of the HEC infrastructure and the VisTrails.

A. *HEC Overview*

As illustrated in Fig. 2, HEC comprises four categories of nodes: Secure Front-End (SFE) nodes, Pleiades Front-End (PFE) and Bridge Nodes, Portable Batch System (PBS) nodes, and four types of compute nodes.

The front-end layer of the HEC infrastructure contains 14 Pleiades Font-End (PFE) nodes and 4 Bridge Nodes. These nodes provide environments for users to perform file transfers, file manipulations, and job submissions. Users are required to first log onto Secure Front-End (SFE) nodes using SSH+RSA two-factor authentication in order to be able to log on to one of the Pleiades Font-End (PFE) nodes and Bridge Nodes.

Pleiades deploys the Portable Batch System (PBS), developed by Altair Grid Technologies, LLC., for all compute job submissions, monitoring, and management. PBS adopts job queues to manage pending work and acts as a scheduler. It dispatches jobs to be run on one or more compute nodes, based on a combination of factors such as mission shares (a certain percentage of CPU's on Pleiades are allocated to each NASA mission directorate), job priority, queue priority, and job size. After users log onto the front-end nodes, they are able to issue commands to interact with PBS to submit and manage their jobs.

There are four architectures of computing nodes that are currently available on Pleiades as shown in Table I and users can specify the architecture type and the number of nodes when requesting compute time through a PBS script.

B. *Vistrails Extension*

VisTrails [2] is a scientific workflow and provenance management software package used in a number of different fields including computer graphics and Earth science research. As shown in Fig 3, VisTrails provides a collection of workflow widgets to allow users to visually design a multi-step executable experiment. In our project, value is gained by creating a solution to allow VisTrails to directly submit workflows as jobs to the Pleiades system from user's

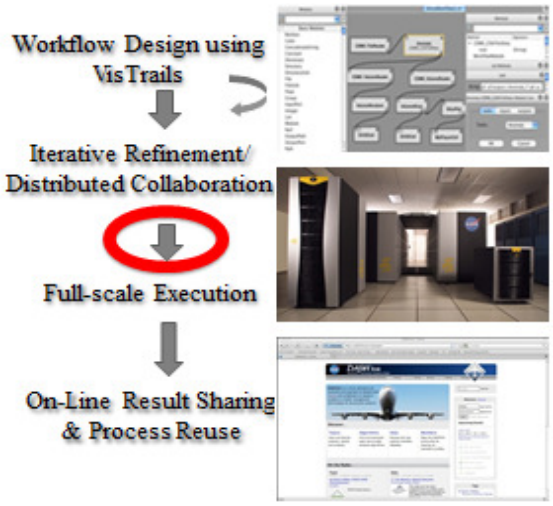

Fig. 3 Extension to VisTrails.

local computer. Processing results are stored on the disk at Pleiades and users are notified when the job completes. Such asynchronous mode facilitates long-lasting VisTrails workflow processing.

C. Technical Issues and Strategies

In order to connect VisTrails to HEC, several challenges were identified. First, NASA Ames HEC facility only allows SSH access. Second, HEC compute nodes designated for large-scale analysis can only be accessed through a PBS scheduler. Third, HEC intends to support a large group of users and work items simultaneously, therefore asynchronous connections HEC are the only option. Fourth, there are four types of node architectures deployed on Pleiades, differing in CPU types, number of nodes, memory size/nodes, and speed.

To address the aforementioned challenges, our strategy is to first identify relevant quality attributes, and then design an architecture around the identified attributes validated by the formal Architecture Tradeoff Analysis Method (ATAM) methodology [9]. The scope of our project is to establish a thin layer bridging between scientists using remote VisTrails installation and the high-performance computing Pleiades system.

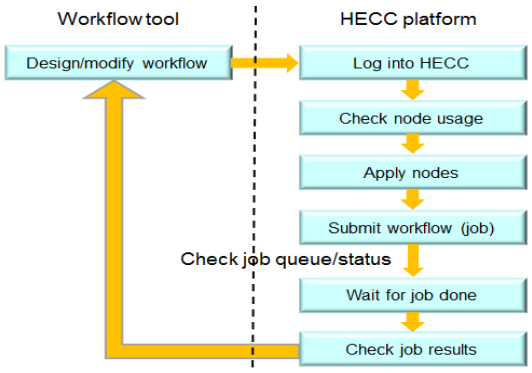

Fig. 4 Workflow-HEC connection workflow.

III. Architectural Design

Our solution is to extend VisTrails with a VisTrails-HEC plugin, in order to provide scientists with a built-in facility to submit workflow processing requests to be run on NASA's Pleiades systems, and to receive updates on the status of their processes. We gather these processes in a middle-tier server, as shown in Fig. 4. More details will be discussed in a later section.

A. Design Principles

We adopted the Architecture Tradeoff Analysis Method (ATAM) methodology [9], a systematic architectural analysis method, to justify and evaluate the architecture designed for the project. Working with the NASA NEX group, we have identified the important driving quality attributes for the architecture as listed in the Table II: security, reliability, availability, usability, performance, scalability, extensibility, interoperability, and asynchrony.

Three key principles in our architectural design are the blackboard, client/server, and publisher/subscriber models. The blackboard architecture model [10] is used to collect different scientists' processing jobs to a central server, while decoupling the scientists from the HPC servers and handling workflow scheduling in a remote scheduling server. Such a remote server will contain a 'blackboard' of requests to be processed and a scheduler that employs an algorithm to launch scripts from a Pleiades Front-End (PFE) node. A thin client and fat server model is used to increase responsiveness on the client side. The scheduling will be moved to the aforementioned designated scheduler server. The publisher/subscriber model is also used to further increase responsiveness of the system. Scientists will subscribe to the scheduler server and receive notifications on the status and progress of their workflow processing requests.

Table II. Quality attributes.

Quality Attribute	Description
Security	System is protected from unauthorized access with data confidentiality.
Reliability	System is stable and data integrity is guaranteed.
Availability	System service is available 24X7.
Usability	System is easy to configure and use.
Performance	System effectively utilizes computing resources.
Scalability	System can accommodate a large number of users and data.
Extensibility	System's functionalities are easy to extend and enhance.
Interoperability	System can interface with different SWF tools and commodity H/W.
Asynchronousness	Client can asynchronously retrieve job execution result from system.

B. Workflow

We studied the current interactions of a NEX user between VisTrails workflow design environment and workflow execution in the HEC system, and the process is summarized in Fig. 4. After designing a workflow using VisTrails, a scientist will need to switch to Unix prompt to log onto HEC using the two-phase procedure. After checking compute nodes availability using HEC commands, she can request a combination of four architecture types of nodes on Pleiades. The scientist can then submit the workflow (a job on Pleiades) to a corresponding job queue and later check the job status using PBS commands. After the job is finished, the scientist can review the job results, and go back to VisTrails to modify the workflow if needed and then repeat the process until she is satisfied with the results.

Our design thus aims to bridge the gap between workflow design tools and the HEC environment, to provide system-level support to allow VisTrails users to conduct the aforementioned HEC-side activities without ever leaving their local VisTrails platform.

C. Architectural Design

Fig. 5 illustrates the architectural design of our VisTrails-HEC plugin. The infrastructure consists of three tiers: front-end, middle-tier, and backend tier. The backend tier is the actual HEC system hidden from NEX users. Three interfaces are leveraged by our system: front-end service is in charge of user log in access control, PBS service is in charge of job scheduling, and computing service is in charge of job execution. The front-end tier is embedded in an HEC plugin, which is implemented as a VisTrails extension module to seamlessly adapt a VisTrails workflow to the HEC computing environment.

The middle tier receives compute requests from the front-end tier, distributes the requests to the Scheduler module, and schedules the requests to run in the backend

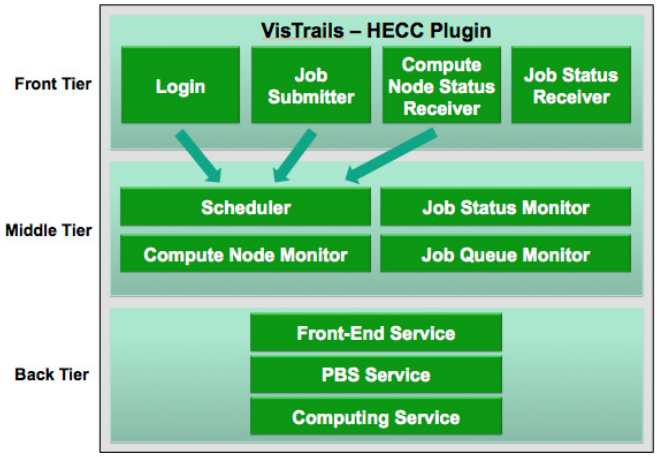

Fig. 5 Architectural design of VisTrails-HEC plugin.

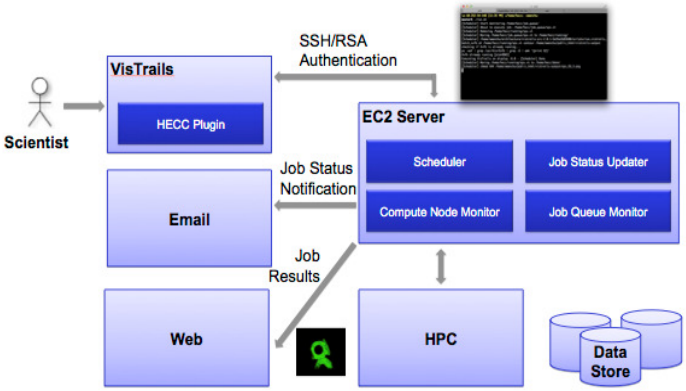

Fig. 6 Deployment view.

tier. The Scheduler coordinates the usage of HEC computing resources, by connecting to HEC's front-end and bridges servers via the SSH protocol and communicating with the Scheduler Agent to dispatch compute jobs requested by scientists. The Scheduler module gathers the backend tier's status through the Job Status Monitor and the Compute Node Monitor, and schedules the requests according to the loading of the backend system. The Job Status Monitor is responsible for reporting the status of the requests to the HEC plugin. The Job Queue Monitor contributes to the Scheduler the availability of corresponding HEC job queues.

Note that this infrastructure is not limited to VisTrails and NASA HEC. As shown in Fig. 5, the major component of the workflow-HPC connection is designed as an independent middle tier exposed as Web services. At the front end, it can interact with any workflow tool with a corresponding plugin. At the backend, it can interact with any HPC environment.

As shown in the deployment view in Fig. 6, the middle tier interacts with the backend high performance computing HPC facility and data stores, to monitor job status and job queue information. The middle-tier will reside on a web server (e.g., Amazon EC2 server). VisTrails-HEC plugin interacts with the middle-tier server through two-phase authentication process. When a job is finished, notifications will be delivered to users in email. If the middle tier is independent outside of NASA HPC systems, for example on Amazon or Heroku (see Section V for details), users may then view the execution results through web interface online.

IV. Computing Resource Planning

As mentioned earlier, Pleiades system comprises four types of compute nodes, each with different architecture and in turn, with different cost of usage. As shown in Fig. 4, HEC interface allows a user to check the availability of the four types of compute nodes and select a combination of different types of nodes for a specific job. Up to now, users usually select one type of architecture to submit a single job. However, a job comprising components that can run in

parallel may assign them to different compute nodes based on corresponding usage costs.

From the perspective of HEC, it is important to increase the overall node utilization through load balancing over all of its comprising computing resources. However, from a user's perspective, both cost and performance (response time) may have to be taken into consideration to decide how to request compute nodes. The definitions of the key performance indicators (KPIs) are listed as below.

Estimated *response time* (T) of a job refers to the time interval between when a job is submitted to a job queue and when the job is finished (i.e., a notification is sent to user). It includes the waiting time of a job in corresponding job queue and the execution time of the job. Since a job may be distributed to a combination of multiple types of compute nodes, its response time will take the maximum time period elapsed on different node types.

$$\Omega = \{\text{Sandy Bridge}, \text{Westmere}, \text{Nehalem}, \text{Harpertown}\}$$
$$T = \max_{i \in \Omega}\{tw_i + te_i\}$$

where *tw* denotes job waiting time, and *te* denotes job execution time.

The waiting time of a job on a compute node type mainly depends on its load. The lighter load a compute node type has, the more likely the job will obtain a shorter response time. Note that all data, including raw data and execution results, are hosted at NEX, and a push-code-to-data strategy is adopted. Therefore, without losing generality, code transmission time will not be considered. In addition, job sections (workflow sections) distributed to different node types may need to communicate based on their relationships defined in the workflow. Transferring results among them may incur some overhead. In this paper, we do not consider such an overhead.

Estimated cost (C) of a job refers to the sum of the cost over each type of the compute nodes, based on corresponding rate and estimated usage time.

$$C = \sum_{i \in \Omega} r_i * n_i * te_i$$

where *r* denotes the price rate of the type of compute node, *n* denotes the number of the type of node used, and *te* denotes the execution time of the job on the type of nodes.

Since users intend to minimize their response time and minimize the cost they have to pay, we define a *utility function* as a weighted sum of these two KPIs:

$$f = w_p * \frac{T - avg_p}{std_p} + w_c * \frac{C - avg_c}{std_c}$$

where w_p, w_c represent the weights of the response time and cost, respectively; avg_p and avg_c represent the average response time and cost for the same scale of jobs, respectively; std_p and std_c represent the standard deviation of response time and cost for the same scale of jobs, respectively.

Note that this utility function can be extended to include other attributes. Also note that our resource scheduling investigation here is from the perspective of HEC users; it does not conflict with the job scheduling facility embedded in the supercomputing center. For example, a user may choose to send parts of a job to Sandy Bridge nodes and the other parts to Harpertown nodes because the two types of compute nodes charge differently.

The compute node selection problem can be thus modeled as a Linear Multiple-Choice Knapsack Problem (LMCK) [11]. Given a set of items in several classes and a knapsack, where each item has a weight and profit, and the knapsack has a capacity, LMCK aims to select a certain number of items in each class to be placed in the knapsack within the capacity yet has the highest total profit.

The compute node selection problem can be formalized as a LMCK problem in the following way:

- The compute nodes in HEC represent the items in LMCK;
- The different types of Intel Xeon processors represent the classes in LMCK (each class comprises multiple items);
- LMCK aims to select zero to many compute nodes in each class;
- The objective is to minimize the response time as well as minimize the cost, under the constraint that the total response time is less than *Tm* and the total cost is less than *Cm*.

The problem is thus formulated as:

$$Min \quad \sum_{i \in \Omega} \sum_{j \le |C_i|} w_p * \frac{T - avg_p}{std_p} + w_c * \frac{C - avg_c}{std_c}$$

The LMCK problem is NP-hard. For large systems, it may be highly difficult to find the optimal solution. As the first step, we adopted Pisinger [12]'s solution to LMCK problem centered on a partitioning algorithm. The optimal solution x^* to LMCK is composed by the LP-optimal choices b_i in each class, where $x_i b_i$=1.

V. Prototype Implementations

A prototype of the proposed system has been implemented as a proof of concept. In spite of the fact that it is still a prototype, it implements the full set of usable features on the client side as a VisTrails plugin. These features include user login, job status monitoring, and job scheduling. A Scheduler Server has also been implemented to receive compute job requests and generate the PBS script. Along with the Scheduler Server, we also simulated the two-level logging process and the job execution to mimic the real situation since we now only have limited access to the NASA HEC environment. The middle tier was implemented as a web application deployed on the open Heroku platform using Ruby on Rails (http://rubyonrails.org/), providing RESTful web services.

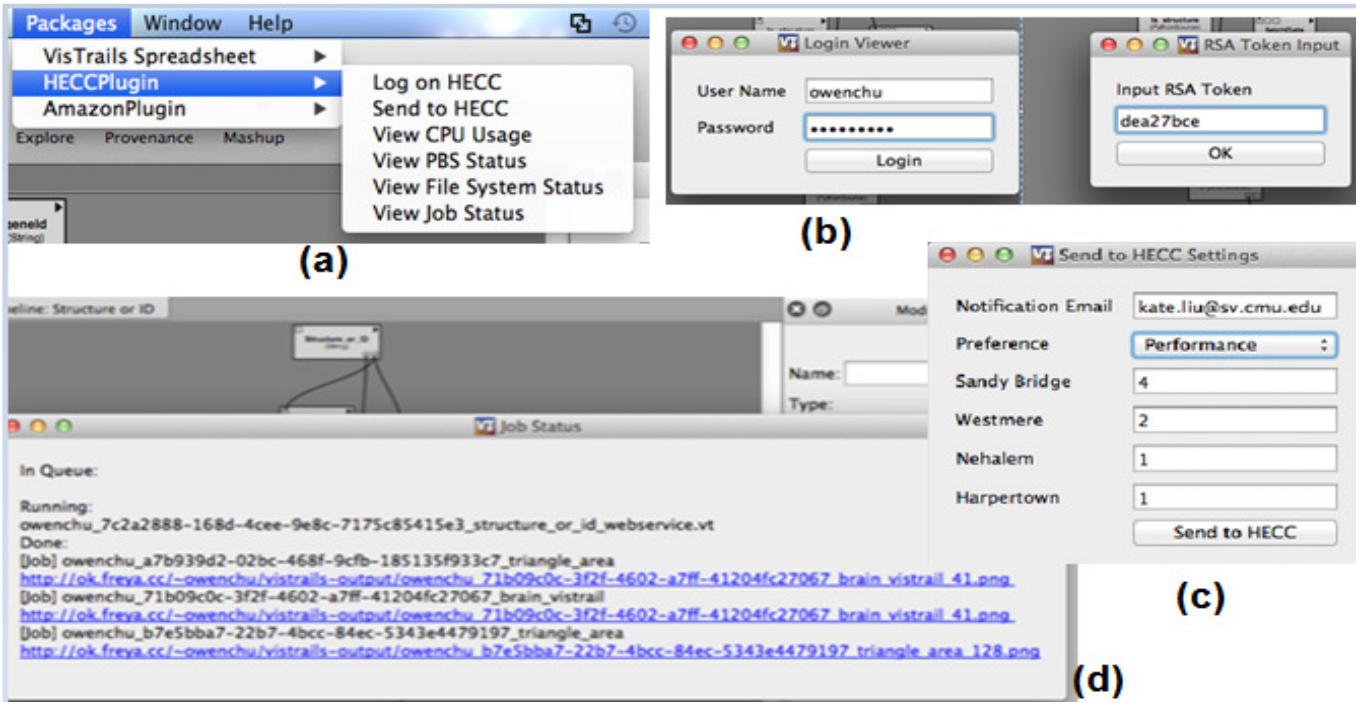

Fig. 7 Prototyping system screen shots.

Ruby on Rails is a popular technology known for helping develop web applications and services.

As described in the previous sections, VisTrails is the tool currently used by NEX scientists that provides a graphical interface for designing and managing workflows. VisTrails provides a plugin infrastructure to support extensibility for new features. VisTrails plugins are implemented in Python. The GUI framework is built using Qt [13], a cross-platform application and UI framework providing tools to help streamline the creation of applications and user interfaces for desktop, embedded and mobile platforms.

The prototype of the middle-tier is currently deployed on an Amazon EC2 Ubuntu instance. It monitors new workflow jobs sent from VisTrails users and generates PBS scripts according to user configurations.

A. *VisTrails-HEC Plugin*

One challenge of building the VisTrails-HEC plugin is the accessibility of PBS through the PFE nodes. Accounts must be granted to access these nodes. One prerequisite is to gain developer access to NASA's computing resources. When prototyping the solution, we chose to simulate the different nodes in the Pleiades system and the servers involved in the plugin. However, access of NASA's nodes is required to test the integration of the solution more comprehensively.

The implemented VisTrails plugin adds a set of functionalities to VisTrails users, as shown in Fig. 7(a). After a workflow is designed in VisTrails, a user can sign onto HEC with the "Log on HECC" menu item. A dedicated "remoteLogin" plugin has been implemented. The included pexpect python package further facilitates the program to detect and react with RSA authentication process.

Following HEC security settings, we have simulated a two-factor (SSH+RSA) logging mechanism to register a VisTrails session to HEC [6]. As shown in Fig. 7(b), when a user intends to "Send to HECC" a workflow, a phase-one window will pop up to allow a user enters user name and password. Afterwards, a phase-two window will pop up and prompt the user to enter the code that appears on her RSA token to connect to HEC. User name will be remembered by the VisTrails-HEC plugin through configuration. Such a design will allow scientists to leverage HEC resources without ever leaving the VisTrails environment. The second reason is the exploratory feature of scientific workflow development. When a scientist modifies a workflow and runs it many times, such an embedded feature will significantly simplify HEC connection efforts.

NEX users can use "View CPU Usage," "View PBS Status" and "View File System Status" to fetch the current status of HEC. Currently, we crawl the real-time Pleiades status pages (under http://www.nas.nasa.gov/monitoring/hud/realtime/) and render the corresponding information through the *QWebView* class in QT. Once again, we provide a single access point for NEX users.

The item of "Send to HECC" will provide both automatic and manual ways to select computing nodes, as shown in Fig. 7(c). Currently, the decision vector comprises only performance and cost. Two automatic options are

provided based on user preference of which factor is more important. As an example shown in Fig. 7(c), *performance* is selected to be more important, and a recommendation of selecting a combination of computing node types and nodes is presented (Sandy Bridge: 4; Westmere: 2; Nehalem: 1; Harpertown: 1). If user agrees and clicks on the "Send to HECC" button, a job carrying the workflow will be sent to HEC system. The plugin makes this possible by uploading the current VisTrails project file and a generated configuration file to the server. A user may send multiple jobs to the Scheduler Server and afterwards, use the "View Job Status" option to retrieve job statuses.

B. VisTrails-Amazon Plugin

Due to the fact that not all research groups could obtain NASA HEC access, we have decided to build another plugin to connect VisTrails to an open accessible HPC environment.

Rapidly advanced cloud computing technology has enabled Infrastructure as a Service (IaaS), which provides end users flexible and reliable access to resources that will meet dynamic computational needs. Particularly, Amazon EC2 offers Cluster Compute instance type optimized for high performance computing applications [14].

As a proof-of-concept example, we selected Amazon EC2 because it supports (limited access) free accounts. As another side effect, such a VisTrails-Amazon plugin opens up new vistas for VisTrails users without NASA HEC access to exploit high performance computing capability.

As shown in Fig. 7(a), a VisTrails-Amazon plugin provides the same collection of facilities as those of the VisTrails-HEC plugin. Unlike NASA HEC using a two-phase login process, Amazon EC2 requires a one-phase SSH login. A VisTrails running instance is installed on Amazon EC2 and workflow jobs can be thus sent to Amazon EC2 to be executed. After a compute job is done, the server automatically sends a notification email with the web link of results for users to view online. Fig. 7(d) shows when a user selects 'View Job Status' after a job has been submitted to Amazon EC2. The example shows one job is still running, while some jobs have been successfully executed and the links to their results viewable online are provided as well.

C. Further Discussions

The scalability of the middle-tier server holding all requests is a major concern, because it could be a potential architectural bottleneck. Because scientists submit processes and receive updates to and from the remote server through VisTrails, increased traffic to this remote server could impede or disrupt scientists from being able to submit their processes and receive updates. A potential solution could be to increase the number of remote servers and allow VisTrails to choose the remote server to use based on the number of current requests of each server.

The availability of a parallel solution is another concern, because there are currently no openly-available ways to automatically parallelize a VisTrails program. Without such a capability, the program can only be run as a batch job and not in parallel. A solution to this concern could be to write an independent automatic parallelization component but it would be highly resource-intensive and still mostly applicable to a subset of possible programs.

VI. Related Work

Scientific workflows are typically oriented to big data analysis thus require large computing power and mass storage capabilities [5]. However, Critchlow and Chin [8] have indicated that a gap exists between workflow design on workflow engines (tools) and workflow execution on supercomputers.

Several scientific workflow management systems have been widely used nowadays. Script [15] provides a scripting language to allow scientists to define, execute, and manage large-scale scientific workflows. An execution engine is associated to dispatch parallel computations to Grid environment, based on user specifications and data flows defined in scripts. In Kepler [16], a workflow is comprised of interconnected actors (components). Such a setting allows actors to be executed concurrently and communicated with each other through interconnected ports. Wang et al. [17] introduce a MapReduce actor into Kepler supported by a Hadoop infrastructure. Kepler users can compose and execute MapReduce applications; and computations are moved to partitioned datasets and run in parallel. Pegasus [18] provides a pegasus-mpi-cluster tool to run High Throughput Computing (HTC) scientific workflows on systems designed for HPC. Microsoft Trident [19] also can run scientific workflows on Windows HPC clusters. Complementary to these efforts that focus on embedding functions in scientific workflow tools to support HPC capability, our project focuses on building a light-weight plugin into scientific workflow tools to connect them to HPC backend.

Juve et al. [20] studied data sharing options for enhancing scientific workflow performance and cost on Amazon EC2. Mehrotra et al. [21] explored running NASA HPC applications on the Cluster Compute instance of Amazon EC2 Services, and compared its performance with running NASA HPC applications on NASA Pleiades supercomputer (i.e., HEC). They concluded that due to communication overhead, the traditional supercomputer environments (e.g., NASA HEC) exceeds the cloud computing environment in running large-scale HPC applications requiring large core counts. However, their study confirms our motivation of allowing VisTrails users without NASA HEC access to exploit high performance computing capability of Amazon EC2, especially when single core is sufficient for running a workflow.

VII. Conclusions and Future Work

In order to leverage the supercomputing capabilities, domain scientists need to be able to quickly move code and computation between their development environments (sandbox) and the supercomputing environment at NASA HEC facilities. This paper describes our extension to the VisTrails scientific workflow tool that automates such a process. Regardless of the current HEC constraints, we have presented an "ideal" architecture that supports seamless migration of scientific workflows between a development environment and a supercomputing environment (NASA HEC and Amazon EC2). By providing a single access point, a NEX user can design workflows, send to execute on high-end computing environment, and view execution results online. We believe that our work will help increase the number of scientists to adopt the NASA computing ecosystem.

In future research we plan to construct a middle tier inside of NEX to host a VisTrails running environment, empowered by our VisTrails-HEC plugin. This layer will intend to provide a system-level support to handle all VisTrails-supported HEC access and data management. We also plan to accumulate practice data to create benchmarks for the presented workflow scheduling approach in this paper.

VIII. Acknowledgement

This work is partially supported by National Aeronautics and Space Administration, under grant NASA NNX13AB38G. The authors sincerely appreciate Dr. Thomas Hinke for his constructive discussions and comments.

VIV. References

[1] NASA, "NASA Earth Exchange (NEX)", 2012, Accessed on, Available from: https://c3.nasa.gov/nex/.

[2] S.P. Callahan, J. Freire, E. Santos, C.E. Scheidegger, C.T. Silva, and H.T. Vo, "Managing the Evolution of Dataflows with VisTrails", in Proceedings of *the 22nd International Conference on Data Engineering Workshops*, 2006, pp. 71.

[3] E. Deelman and Y. Gil, "NSF Workshop on the Challenges of Scientific Workflows", (ed.), May 1-2, 2006.

[4] S.B. Davidson and J. Freire, "Provenance and Scientific Workflows: Challenges and Opportunities", in Proceedings of *ACM SIGMOD*, 2008, pp. 1345-1350.

[5] M. M. Sonntag, D. Karastoyanova, and E. Deelman, "Bridging the Gap between Business and Scientific Workflows: Humans in the Loop of Scientific Workflows ", in Proceedings of *IEEE 6th International Conference on e-Science (e-Science)*, 2010, Queensland, Australia, Dec. 7-10, pp. 206-213.

[6] NASA, "High-End Computing Capability (HECC)", 2012, Accessed on 12/25/2012, Available from: http://www.nas.nasa.gov/hecc/.

[7] NASA, "Pleiades", 2012, Accessed on 12/26/2012, Available from: http://www.nas.nasa.gov/hecc/resources/pleiades.html.

[8] T. Critchlow and G.J. Chin, "Supercomputing and Scientific Workflows Gaps and Requirements", in Proceedings of *2011 IEEE World Congress on Services (SERVICES)*, 2011, Washington DC, USA, Jul. 4-9, pp. 208-211.

[9] P. Clements, R. Kazman, and M. Klein, *Evaluating Software Architectures: Methods and Case Studies*, 2001, Addison-Wesley Professional.

[10] P. Korpipää, *Blackboard-Based Software Framework and Tool for Mobile Device Context Awareness*, Vol. 579, 2005, VTT Publications.

[11] E. Zemel, "The Linear Multiple Choice Knapsack Problem", *Operations Research*, Nov.-Dec., 1980, 28(6): pp. 1412-1423.

[12] D. Pisinger, "A Minimal Algorithm for the Multiple-Choice Knapsack Problem", *European Journal of Operational Research*, 1995, 83: pp. 394-410.

[13] Digia, "QT", 2012, Accessed on, Available from: http://qt.digia.com/product/.

[14] Amazon, "High Performance Computing (HPC) on AWS", 2012, Accessed on, Available from: http://aws.amazon.com/hpc-applications/.

[15] M. Wilde, I. Foster, K. Iskra, P. Beckman, Z. Zhang, A. Espinosa, M. Hategan, B. Clifford, and I. Raicu, "Parallel Scripting for Applications at the Petascale and Beyond", *IEEE Computer*, 2009, 42(11): pp. 50-60.

[16] B. Ludäscher, I. Altintas, C. Berkley, D. Higgins, E. Jaeger, M. Jones, E.A. Lee, J. Tao, and Y. Zhao, "Scientific Workflow Management and the Kepler System", *Concurrency and Computation: Practice & Experience*, 2006, 18(10): pp. 1039-1065.

[17] J. Wang, D. Crawl, and I. Altintas, "Kepler + Hadoop: A General Architecture Facilitating Data-Intensive Applications in Scientific Workflow Systems", in Proceedings of *the 4th Workshop on Workflows in Support of Large-Scale Science*, 2009, Portland, Oregon, USA, Nov. 15, pp.

[18] E. Deelman, G. Singh, M. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. Berriman, J. Good, A. Laity, J. Jacob, and D. Katz, "Pegasus: A Framework for Mapping Complex Scientific Workflows onto Distributed Systems", *Scientific Programming Journal*, 2005, 13: pp. 219 - 237.

[19] R. Barga, J. Jackson, N. Araujo, D. Guo, N. Gautam, and Y. Simmhan, "The Trident Scientific Workflow Workbench", in Proceedings of *4th IEEE International Conference on eScience*, 2008, Dec. 7-12, pp. 317-318.

[20] G. Juve, E. Deelman, K. Vahi, G. Mehta, B.P. Berman, B. Berriman, and P. Maechling, "Data Sharing Options for Scientific Workflows on Amazon EC2", in Proceedings of *ACM/IEEE International Conference for High Performance Computing, Networking, Storage and Analysis*, 2010, New Orleans, LA, USA, Nov. 13-19, pp. 1-9.

[21] P. Mehrotra, J. Djomehri, S. Heistand, R. Hood, H. Jin, A. Lazanoff, S. Saini, and R. Biswas, "Performance evaluation of Amazon EC2 for NASA HPC Applications", in Proceedings of *The 3rd workshop on Scientific Cloud Computing (ScienceCloud)*, 2012, Delft, Netherlands, Jun. 18-22, pp. 41-50.