Computational Science and Engineering Online

(CSE-Online): A cyber-infrastructure for scientific computing

Thanh N. Truong*, Manohar Nayak, Hung H. Huynh, Tom Cook, Priya Mahajan, LeThuy T. Tran, Jannu Bharath, Shrish Jain, Ha B. Pham, Chaiwoot Boonyasiriwat, Nhat Nguyen, Evan Andersen, Yong Kim, Suengkeol Choe, Jihoon Choi, Thomas E. Cheatham, III1, and Julio C. Facelli2

Department of Chemistry and the Interdisciplinary Computational Engineering and Science Program
1Departments of Medicinal Chemistry and of Pharmaceutics and Pharmaceutical Chemistry and of Bioengineering
2Center for High Performance Computing
University of Utah
315 South 1400 East, Rm 2020
Salt Lake City, Utah 84112

*Corresponding author email: truong@chem.utah.edu

Abstract

With the expansion of the Internet and World-Wide-Web (or the Web), research environments have changed dramatically. As a result, the need to be able to efficiently and securely access information and resources from remote computer systems is becoming even more critical. This paper describes the development of an extendable integrated Grid enabled web-based simulation environment for computational science and engineering, called Computational Science and Engineering Online (CSE-Online, http://cse-online.net). CSE-Online is based on a unique client-server software architecture that can distribute the workload between the client and server computers in such a way as to minimize the communication between the client and server thus making the environment less sensitive to the network instability. Furthermore, the new software architecture allows the user to access data and resources on one or more remote servers as well as the computing grid while having the
full capability of the web-services collaborative environment. It can be accessed anytime and anywhere from a web-browser connected to the network by either a wired or wireless connection. It has different modes of operations to support different working environments and styles. CSE-Online is evolving into a middleware that can provide a framework for accessing and managing remote data and resources including the computing grid for any domain, not necessarily within computational science and engineering.
1. INTRODUCTION

...a new age has dawned in scientific and engineering research, pushed by continuing progress in computing, information, and communication technology, and pulled by the expanding complexity, scope, and scale of today’s challenges. The capacity of this technology has crossed thresholds that now make possible a comprehensive ‘cyberinfrastructure’ on which to build new types of scientific and engineering knowledge environments and organizations and to pursue research in new ways and with increased efficacy.

—NSF Blue Ribbon Advisory Panel report on cyberinfrastructure, 2003

There have been a number of activities directed toward achieving the vision outlined above, such as the creation of ‘collaboratories’, computing ‘Grids’, e-Science, and numerous domain-specific projects aiming to create a new cyberinfrastructure. Such projects would ultimately induce a paradigm shift in the way we do research, access scientific data, and collaborate.

All of these activities recognize the fact that with the expansion of the Internet and World-Wide-Web (or the Web), research environments have changed dramatically. As a result, the need to be able to efficiently and securely access information and resources from remote computer systems is becoming even more critical. Furthermore, in many cases, and particularly for computing intensive work, a remote system is not just one computer or server, but rather a collection of computers over the internet or via a grid structure, such as the TeraGrid. The same is also true for data intensive work where data can be stored in distributed databases located on many computers at different locations.

Nearly all computers for scientific computing purposes are running Unix or Linux operating system. A typical work environment for a user to access these computers remotely is to open an X-window session on his local computer to connect directly to these servers using the secure shell (SSH) protocol. He may choose to connect to more than one server concurrently. In this case, the user has an account, owns directories on each of these servers, and communicates directly with the server OS using command line commands. Users do not see the desktop environment of the server using SSH. To visualize data, users can run an application that exports a
display to the local computer from the server using the X11 forwarding protocol. The X11 protocol can “serve out” the screen, keyboard, and mouse, etc. Unfortunately, the X11 protocol often has difficulties in exporting graphics from the server. In this technology, all events are executed on the server. Stability can be a problem because a weak link in this system is the network connection. Because the network connection is used constantly, the entire X11 session fails when the network connection fails and it requires significant network bandwidth for real time performance.

Recent advances in Web technologies has lead to a different mechanism for accessing remote data and resources. Such technologies have been used extensively in developing a new research environment called ‘collaboratories’, laboratories without walls, to expand the research capabilities far beyond the limitation of the local facilities available to the researchers and to collaborate in solving specific complex scientific problems. These collaboratories, for the most part, can be classified into two types: data sharing oriented or remote access scientific instrument oriented. The Research Collaboratory for Structural Bioinformatics is an excellent example of the data-oriented collaboratory that provides access to databases of biological structures and tools for determining and analyzing these structures. Most existing collaboratories are instrument driven and provide the capability for real-time data acquisition from remote research instruments through web-accessible servers in a seemingly transparent way and are often focused specifically on a particular complex scientific problem. The Space Physics and Aeronomy Research Collaboratory (SPARC) provides an internet-based collaborative environment for studies of space and upper atmospheric science facilitating real-time data acquisitions from a remote site in Greenland. The Materials Microcharacterization Collaboratory provides remote access to facilities that perform electron-beam microcharacterization of materials. The Environmental Molecular Science Collaboratory allows remote instrument control of the NMR and FTICR spectrometers located at the Pacific Northwest National Laboratory.

The possibility of using such web technologies to provide a new framework for scientific computing also attracted a lot of interest in the computer science community in the mid-1990s. Numerous reports illustrate the potential of web-based framework for simulations. However, realization of such potential was rather limited and the interest in this area has significantly dropped since 2000 due to the lack of real applications as pointed out
by Kuljus and Paul\textsuperscript{18}. Fortunately, interests in developments of web-based simulation environments have been revitalized recently by domain scientists, particularly in conjunction with the computing Grid. For instance, within the molecular science simulation community, there are a number of projects such as the development of Collaborative Multi-scale Chemical Science (CMCS)\textsuperscript{19}, Grid Enabled Molecular Science Through Online Networks Environments (GEMSTONE)\textsuperscript{20}, Computational Chemistry Grid (GridChem)\textsuperscript{21}, Discovery Net\textsuperscript{22}, CombChem\textsuperscript{23}, Reality Grid\textsuperscript{24}, and the Virtual Laboratory Project\textsuperscript{25}, among others.

All of these web-based collaborative or simulation environments share the common Web-services software architecture. In this architecture, users access information and request for services provided by a particular web portal from within a web browser. Users can connect to the web portal or server either via a non-secure (http) or secure (https) connection from a web page. At the server, a web application configured as an agent (known as servlet) is running to listen for and provide such connections. The web application can perform a variety of tasks ranging from the simple delivering of data to the requesting user to complicated processes that involve authentication, receiving input data from the user, sending a request to the server to run a different application, or receiving output data from the server and then delivering such data back to the user web-browser.

In the Web-services architecture, users have a relatively thin web client that communicates directly to the web application servlet and not the server operating system (OS). Thus, the authentication process is performed by the servlet and not the web server’s OS. In addition, users do not generally own accounts and directories on the server’s operating system. However, the web application may provide web accounts and virtual directories for the end user but this is significantly different from having an account directly on the server itself, for instance, files in virtual directories are not owned by the user according to the server OS. Requests for resources on the server are done by the servlet. Furthermore, all computations are done on the server computer. In other words, all the web communications with the server are channeled through the servlet which provides the web data and protects the underlying server. The user must connect to a particular web portal to access the data and services. This environment is quite useful for those who only care that the requested data or service, i.e. the end result is delivered and do not care how the data was stored or the process for generating such data. This environment is very different from opening an X-window session on the user’s local computer to connect directly to any server,
on which he has an account, to access his files and resources. In today’s scientific computing research environments most users have accounts, resource allocations, applications, and data on more than one server. Web-services environments cannot access these data and resources. Users still rely on the SSH technology to do so. In addition, and similar to the X-window technology, if the servlet crashes or network communication becomes unavailable, the user’s side application will immediately fail and/or the objects and pages for the web application will not be available.

In this study, we describe our current efforts in developing an extendable integrated Grid enabled web-based simulation environment for computational science and engineering, called Computational Science and Engineering Online (CSE-Online). CSE-Online is based on a unique client-server software architecture that can distribute the workload between the client and server computers in such a way as to minimize the communication between the client and server and thus making the environment less sensitive to the network instability. Furthermore, the new software architecture allows the user to access data and resources on one or more remote servers as well as the computing grid while having the full capability of the web-services collaborative environment. This paper is organized as follows. Section 2 describes the overall software architecture of CSE-Online and its implementation. The following sections provide further details related to its four major components, namely the Knowledge Management System, Visualization/Analysis, Collaborative environment/Communication, and the Computing Environment components, respectively. In discussion of these components, our focus is on the high-level design and functionalities. We also provide some discussion on technologies used for implementation.

2. SOFTWARE ARCHITECTURE AND IMPLEMENTATION

The main goal of this integrated, extendable, platform-independent, and secured environment is that it enables a seamless interface and data flow between a variety of different computational application tools and databases. This environment pulls together independent application tools and databases to provide a complete bottom-up environment that integrates methods ranging from quantum chemistry to reaction engineering, or to computer-aided drug design which also involves homology modeling, molecular docking, molecular dynamics
(MD) and free energy simulations while having the look-and-feel of a single desktop application. It provides a knowledge management system that allows a user to store, query, retrieve, and mine data extracted by running these application tools. The environment also provides communication tools, which work in both synchronous and asynchronous modes, to facilitate training, user support, and collaboration. It can be accessed anytime and anywhere from a web-browser connected to the network by either a wired or wireless connection while still enabling visualization/analysis of remote data in real time.

Due to the complexity inherent in the specifications of the environment and the existence of numerous relevant technologies, to help us in the design of the software architecture as well as the technology choice we rely on the following four guiding principles:

1. Having a good human experience at the interface is equally as important as solid software architecture.
2. It is critical to strike a balance between employing emerging technologies and providing users with stable and reliable tools.
3. It is important to recognize and support the common needs across disciplines, while providing a flexible framework that can be customized for each particular scientific domain.
4. Not only should individual research be empowered, collaboration across disciplines should be facilitated.

In addition, from a software engineering point of view, the architecture of this environment should also be scalable and extendable to ensure its future growth and support of new and different applications or databases.

**Overall software architecture**

The CSE-Online is based on a client-server architecture model. The client CSEO environment consists of numerous client side java applications that allow the user to access and analyze remote data or use remote resources. It is discussed separately below. The CSE-Online server side has four different separate components and their connectivity to the client environment is different depending on their roles as shown in Figure 1.

The central server component is the CSE-Online Middleware server, which is owned by CSE-Online. It hosts the CSE-Online website where users first login from a web browser to enable the CSE-Online executable file. It has a user management system, which allows the user to share files or databases with other CSE-Online
users and contains the user preference information. It also contains applications and public databases provided by CSE-Online. The user accesses data or applications on the CSE-Online server using the Enterprise Java Bean (EJB) technology. ²⁹

The second component is the private server(s) (it can be more than one remote server). Private servers, in the context of this paper, means the servers where the user has direct access and prefers to store data and/or has access to application tools. CSE-Online has no direct access to these servers. These ‘private’ servers may or may not be owned by the user. The user accesses the private server from the CSE-Online client environment using the Secure Shell (SSH) technology, which provides strong authentication and secure communications over insecure channels. This server can also contain private databases within the CSE-Online knowledge management system as discussed in a latter section. Some installation is required on this server to use this feature.

The third component is the third-party server. It consists of all databases and applications provided by the third-party. Both CSE-Online and the user have no direct access to these servers (i.e. have no directory accounts on these servers). Accessing databases or services from these third-party servers is done by using the web-services ⁵ or http technology ³⁰. CSE-Online helps the user communicate with these web services without the user needing to leave the environment.

The fourth component is the computing resources which can be a remote cluster, distributed clusters or even the computing grid such as the TeraGrid², where the user has allocated resources. The user can access these computing resources from the CSE-Online environment by making a direct connection to these computing nodes using the SSH technology. He can also elect to request such resources from the CSE-Online client environment without making a direct connection to these computing nodes by using the web-services technology if the computing nodes supported such a service.

With the CSE-Online client environment and the four server components, this software architecture offers the user three distinctive modes of operation. Specifically, the three modes are: 1) Connection to a private server mode; 2) Without connection to a private server mode; and 3) Off-line mode. The differences in these modes are mainly how CSE-Online accesses data and computing resources.
1. **Connection to a private server mode**

   Figure 2 illustrates the active components of the ‘connection to a private server’ mode. In this mode of operation, the user wants his data to be on his private server(s) or can access certain applications on this server. The File Manager, a component of the client environment described below, maps the user file directories on that server to the environment, and if enabled, allows the user to run any application on that server. The user can set up to have the ability to create CSE-Online private databases on this server. This system is ideal for users who 1) already have licenses for their applications 2) want maximum security for their data and 3) would like to maintain their own servers. The environment is stable with regard to network interruptions. Communication to the private server is done only during the read/write request. All visualization or analysis of data is done on the client side for maximum performance. A unique feature of CSE-Online is that the environment remains operational when the internet is disconnected and data is not lost during such an event by using a cache process. Attempts to reconnect to the private server are done automatically. In addition, from the CSE-Online client environment the user can also access data and applications provided by the CSE-Online server or third-party servers, and remote computing resources including the computing grid.

2. **Without connection to private server mode**

   Figure 3 shows the active components of the ‘without connection to a private server’ mode of operation. In this mode, the user prefers to have his data on his local computer rather than on a remote ‘private’ server, or does not own a private server. In such cases, the File Manager maps all local storage devices on the user computer such as the local hard disk, CDROM, Floppy, USB memory stick, etc. The user can access data and applications provided by the CSE-Online Middleware and third-party servers. He can also access the remote computing resources from the CSE-Online environment. In summary, the only difference to the user between the ‘connection to a private server’ and ‘without connection to a private server’ modes is that there is no the private server component in the latter mode.

3. **Off-line mode**
Figure 4 shows only the client desktop environment to be the active component in the off-line mode of operation. In this mode, the Internet connection is not available. The client environment behaves as a single integrated application operating on the local computer that can read/write data to the local storage devices. Since the CSE-Online client environment provides many Java-based application tools, such as data analysis and visualization, and also provides a ‘sync’ capability for data on a private server and the user’s local computer, the user can still be productive in the off-line mode.

In the official released version of CSE-Online at http://cse-online.net only the ‘connection to a private server’ mode is available. The other two modes of operation are either being implemented or at the alpha testing stage.

**Desktop-Environment**

The CSE-Online client desktop environment is based on Java and Java Webstart technologies and thus it allows the environment to be independent of the operating systems and web browsers. In other words, its end-user look-and-feel is the same in all systems. Furthermore, it is a ‘thin client’ environment, i.e. no installation is needed by the user beyond the Java Runtime Environment (JRE) needed for running any Java applications. The executable of the desktop environment is downloaded to the user local computer for the first time upon login onto the CSE-Online Middleware server. Subsequent uses on the same computer can access the environment executable from the local cache. It is important to point out that the CSE-Online desktop environment is based on an open software architecture. Such open architecture allows each tool in the desktop environment to be developed independently and thus provides the possibility for third-party client application tools to be supported within the CSE-Online environment. Furthermore, the desktop application tools are loaded at runtime and thus it allows the user to select only desktop tools that are needed for his work to be loaded into the environment. This allows the development of the desktop environment to be extendable, i.e. the environment can support unlimited number of desktop application tools. The size of the executable only depends on the number of tools that were selected by the user. The CSE-Online desktop environment resembles a standard Windows or Linux desktop environment with a toolbar listing application and utility tools that were selected by the user.
The CSE-Online desktop environment has four major components, namely the Knowledge Management System (KMS), Resource Management system (RMS), Visualization/Analysis, and Communication components as shown in Figure 5. These components are discussed separately below. In addition to these components, the CSE-Online desktop environment has a number of desktop application tools which can be separated into either the common or domain-specific tools. The common tools are tools that are useful to all domains whereas the domain-specific tools consist only of tools that are specific to a particular domain. Below is the current list of these tools.

**Common tools:**

1. **Pandora Plot:** Pandora plot is a general-purpose plotting tool. It allows users to plot 2D data and to manipulate and customize their plots (title, labels, ranges, legends, font, plot types etc).
2. **JEKS Spreadsheet**: A simple spreadsheet that has limited functionalities such as copying and pasting data from Microsoft Excel spreadsheet; calculating by rows, columns, or cells; selecting and plotting of data; and saving data to file.
3. **Notepad:** A minimal text editor with a number of useful functionalities such as search, copy, paste, delete, save, etc.
4. **Jpeek Image viewer:** An image viewing tool which can read files in the jpg, png, or bmp format.
5. **SSH Shell:** A SSH shell tool which allows users to log onto any other server from within the environment.
6. **Job Monitor:** A background job manager that lists all the current running jobs on the server and where the output file for each run is created.
7. **Generic GUI:** The Generic GUI allows the user to configure and run any server based tool that he has on his private server for which CSE-Online does not provide a specific GUI. This tool allows for simple execution as well as more complex script based submission.

**Domain-specific tools:**
Most of domain-specific desktop tools are Graphic User Interfaces (GUIs) for scientific applications residing on a remote server or computing nodes. Currently, most GUIs are for computational chemistry applications, some for reaction engineering and computational biology.

1. JME Sketcher: It is a 2D molecular editor. It outputs the molecular structure in a SMILES string format.
2. GAUSSIAN GUI: This is a GUI for the GAUSSIAN quantum chemistry program.
3. GAMESS GUI: This is a GUI for the GAMESS quantum chemistry program.
4. Thermo GUI: It provides an interface to the Thermo application tool residing on the CSE-Online server for calculating thermodynamic properties of a given chemical species.
5. Kinetics GUI: It provides an interface to the Kinetics tool residing on the CSE-Online server for calculating rate constants of elementary reactions from supplied potential energy surface taken from quantum chemistry calculations.
6. Senkin GUI: It provides an interface to the SENKIN sensitivity analysis tool for analyzing complex mechanisms. SENKIN is a part of the CHEMKIN II package.
7. Premixed GUI: It provides an interface to the application tool for modeling steady laminar one-dimensional premixed flames. The application tool is part of the CHEMKIN II package for modeling different types of reactors.
8. Mechanism Editor: This is a useful tool for editing and analyzing complex mechanisms. It is also the GUI for the public mechanism database residing on the CSE-Online server.
9. Sander GUI: It is a GUI for the Sander MD module which is a part of the AMBER package for simulation of biological systems.

Figure 6 shows a snapshot of the user desktop environment that has active windows of several applications.

3. KNOWLEDGE MANAGEMENT SYSTEM

CSE-Online provides users with three different mechanisms for managing data. One is through the common file directory system via the File Manager. The second mechanism is via a multi-level database
management system that allows users to store data in databases for query/retrieval at a later stage. Finally, CSE-
Online has an electronic documenting system called eDoc for editing text and capturing data in different formats
and stores them in different layers with links to the data viewers.

**File Manager**

The File Manager is a GUI that allows a user to manage files in his/her user directories on the local
computer or the remote private server depending on the selected mode of operation. A snapshot of the File
Manager is shown in Figure 7.

For the connection to a private server mode, when the environment is first activated, the user directory
trees on the private server are created. The user can delete, copy, move, and create files on the server as well as
upload files from the local computer to the server and vice versa. These tasks can be done with simple click-
button or drag-drop mouse functions.

The File Manager is especially valuable for a remote user because communication to the server is made
only when there is a request for adding, deleting, updating, or changing the content in the user directories and
data storage areas. In addition, the File Manager can periodically check the content of the user directories and
update the information when it is modified due to certain application processes running on the server (or even the
remote desktop environment).

An advantage of the File Manager is that the communication with the server takes place when an actual
file request is needed. If there is a network failure, the environment is not terminated since it runs on the local
computer. However, any changes to the user directory on the server will not be made until the connection is
reestablished. The File Manager can cache the change and then re-sync those changes in light of any other
changes that may have been made by others in the meantime. Thus, the environment is more stable with respect
to network failure. The File Manager also allows the user to move files across different private servers in a
transparent manner.

When no connection to any remote private server was made, as in the without connection to a private
server mode, the File Manager maps file directories on the user local storage devices.

**Multi-level database management system**
The multi-level DB management system (MDMS) is complete with a front-end Graphic User Interface (GUI), in order to provide end users with an user-friendly tool to manipulate scientific data in databases that are physically exist on a remote server. The MDMS of CSE-Online is intended to be platform-independent, web-accessible, and scalable. It has the capability of allowing the user to customize their search queries. The data retrieved from the database as a result of the user created queries is displayed in a manner that is useful to the end-users and the data can be accessible to different applications of CSE-Online for example, the Protein Databank (PDB) data can be viewed in the BioViewer tool. In fact, the retrieval data can interface with all applications in CSE-Online that can use or view such data.

MDMS has three levels of accessing data (open, limited group, and user-only access) corresponding to three kinds of databases (public, shared, and private databases, respectively). The public databases consist of data that has already been published and available to the scientific community online. Every user of the CSE-Online environment can access these public databases and have the capability to query, retrieve and analyze the data. These databases are maintained by the CSE-Online. Users can submit information to be added to these public databases. A designated gatekeeper will validate such data before it can be posted.

Private databases contain proprietary data of individual users. The user can request the capability to create private databases on the CSE-Online remote servers. It is possible to create private databases on the user private server, however, installation of the CSE-Online database system on such a server is required. The private databases will allow the users to store and manage their own data.

Any user, who has the privilege to create private databases, will have the capability to create shared databases. The user can elect any of his private databases to be a shared database. He is then able to designate specific registered CSE-Online users who can access such a database and assign privileges to these users. The shared databases reside on the CSE-Online server.

For query and retrieval, the user can select different search levels, namely, locally in the selected database or globally in the selected private databases and all accessible distributed public or shared databases which have the same structure. With a single query, the search is transparent to the user whether it is within a single private database or over a large number of distributed databases.
CSE-Online MDMS relies on the Java and Java Webstart technologies for the Graphic User Interface (GUI). For the server side components it involves the standard Enterprise JavaBeans\textsuperscript{29} with the JBoss Application Server version 4.0.2\textsuperscript{34}. The database system used is PostgreSQL\textsuperscript{35} coupled with the Object-Relational Management (ORM) tool called Hibernate\textsuperscript{36}.

**Software architecture and implementation**

Figure 8 shows the overall software architecture of the CSE-Online MDMS. The client has a ‘Query, Retrieval and Management Component’, which connects over RMI to the remote servers in the system as shown above. The remote private servers and public servers have the ‘Data Access Components’ to access data from the database and the file system while the CSEO server has an additional component called the ‘Server lookup component’ to access information of the locations of all the remote servers in the CSE-Online network. The design of the database system consists of a 3-tier architecture with the GUI tier residing on the client machine, the EJBs and Java classes which form the Data Access and Server lookup components forming the middle tier and the database and the file system forming the third data tier in the application. In addition, the CSE-Online MDMS has a component that allows users to access external online public databases from within the CSE-Online environment. The current implementation is by using http, however, we plan to convert it to Web services at a later stage in the development.

**Current functionalities**

The CSE-Online MDMS currently has three public databases accessible to every user of the CSE-Online environment. These are the Protein Data Bank (PDB) database, the Reaction Mechanism database and the Combustion Flame database. The PDB database is a mirror of the Protein Data Bank database from the Research Collaboratory for Structural Bioinformatics (RCSB)\textsuperscript{10}. The Reaction Mechanism database is a collection of mechanisms for combustion of hydrocarbons including soot formation existed in the literature. The Combustion Flame database is a collection of published experimental and theoretical data for different types of combustion flames. Currently, it has a number of premixed flame data. Data from these databases can be viewed and analyzed by tools available within the CSE-Online environment. For instance, data from the PDB database can be visualized using the BioViewer tool as shown in Figure 9, data from the Reaction Mechanism database can be
viewed and analyzed by the Mechanism Editor, and finally data from the Combustion Flame database is
interfaced with the plotting tool which displays plots of any selected data. We have also interfaced to several
external online public databases for structural information, particularly the National Cancer Institute (NCI) small
molecule database\textsuperscript{37} and the National Center for Biotechnology Information (NCBI) collection of public
databases\textsuperscript{38}, which include the Structure, 3D Domains, and PubChem databases. An illustration on the
connection to the NCI database from within the CSE-Online environment is shown in Figure 10.

CSE-Online currently allows users to create two types of private databases, namely the Chemical Properties
and Reaction Mechanism databases. The Chemical Properties database allows users to manage molecular
properties data obtained from quantum chemistry calculations. Users can insert, copy, paste and delete records in
their private databases. Users also have the option of copying and pasting records from one database to another,
but only for databases sharing the same schema. Private databases are read-write as opposed to public databases
which are read-only. There are options available for deleting multiple records from the database as well as for
deleting or renaming the entire database itself.

We have successfully prototyped the shared database component that allows users to share their private
databases to designated users. Work is in progress to make this component fully functional.

We have designed a generic GUI that allows users to create their own specific queries for each database. The
GUI allows the user to select the name of the field that he wants to search, and then apply an ‘equals’ or ‘fuzzy’
search on the search string that he enters. There is the possibility of adding additional terms to the search query by
clicking on the ‘Add Refinements’ button. Clicking on ‘Execute’ will execute the query on the required database.
For private databases, users have the option of customizing their queries by adding additional searchable fields
into the database. These fields are called user-defined fields and allow the user to extend the implemented CSE-
Online static searchable fields by inserting their own.

CSE-Online has the capability of searching across distributed databases, if all the databases share the CSE-
Online schemas by the means of the Server Lookup Component on the CSEO Server that contains the remote
server locations of all accessible public databases. Upon activating the database tool, the connections are made to
these different servers. The user can now execute a query and the data is retrieved and displayed to him
transparently. The user is not even aware that the GUI is accessing more than one database concurrently. This capability has recently been prototyped using the multiple copies of our PDB database located at different servers.

**Electronic Documenting System**

CSE-Online environment has a general-purpose online electronic documenting system called eDoc that is platform independent and has the capability of editing text as well as capturing data in different formats (flat files, images, voice, etc) then storing them in different layers with links to data viewers and original data sources. eDoc is capable of recording the progress of a scientific research project with all associated data and their pedigree in a single multi-layer electronic document.

The main feature of eDoc will be its multiple layers of data underneath a single document presented to the reader. The reader then has the ability to access all layers of data, if so desired. For example, an inserted figure can embed another layer of data, which contains the original data file and link to the application tool that was used to generate it. Existing technologies of what you see on the two-dimensional layout is what you get confine others to seeing the same point of view as of the one who inserted the figure. Such restriction would hinder exploration of different points of view and interpretations on a single object. Furthermore, most often the inserted data are extracted from the original and many relevant data are lost during the documenting process. By allowing for more layers of data to be stored, eDoc preserves all relevant data during the documenting process.

eDoc uses JTextPane which is a Java class that accepts not just text, but also embeds Java components like Buttons, Labels, and even user defined components. This powerful feature allows the ordinary text document to become a launching pad for other programs. Thus, we can create Viewer expansions (e.g. table editors, equation editors, molecule editors) for the ordinary text editor.

In line with the CSE-Online overall software architecture, eDoc also has an open software architecture that allows different data Viewers to be developed independently by third-parties. Such Viewers can also be loaded at run time.

Each data Viewer has the MVC (Model-View-Control) architecture as described below in the Visualization/Analysis component section. eDoc separates the M from the VC and stores it in the eDoc document. The associated VC component of the Viewer can be called over the web. When a user selects to view
an embedded data, the M is sent to the VC at the time of request. The M is not only data, but is also functions
that operate on that data, specifically functions needed by eDoc and other relevant information of the Viewer such
as version number, etc.

Prototype

We have successfully prototyped the eDoc system. Users can edit text and insert figures. eDoc
automatically links the figure with its viewer and original data files. This capability is illustrated in the snapshot
shown in Figure 11. This snapshot shows an inserted plot opened by the plotting tool and the selected data used
for plotting such a graph opened by the spreadsheet application. This illustrates the main feature of the eDoc
system. It is still under active development and more work is needed for supporting different scientific data types
and viewers.

4. VISUALIZATION/ANALYSIS COMPONENT

Since CSE-Online environment is web-based, its visualization component must also be based on the Java
platform independent programming language. Although there are numerous visualization tools for different areas
of research, most of them are based on the platform-dependent OpenGL language and thus cannot be turned into a
web-based solution. Furthermore, mature technologies for Web-based 3D graphics are only available recently.
For these reasons, we developed the graphic component in the CSE-Online environment from scratch. The first
of the web based 3D graphics solutions was VRML (Virtual Markup Modeling Language) 39, which was more of
a modeling language. VRML could not satisfy the requirements of 3D graphics engineers, and failed due to
various technical issues. Java3D was the first popular 3D graphics library for Java 40. However, it suffers severe
performance issues due to its heavy usage of object oriented design. The more recent Java OpenGL (JOGL) 41
language is able to overcome such shortcomings. JOGL is actually a cross language interface or binding,
providing interface to native OpenGL and CG libraries using Java Native Interface (JNI). JOGL is not part of the
Java Standard library. JOGL uses hardware acceleration and could achieve the performance of C++ OpenGL
applications, and provides all the features supported by OpenGL. It is expected to become the de facto graphics
library for the Java platform and usher in the development of graphics applications and games in Java. For these
reasons, all 3D graphic applications in CSE-Online are based on JOGL. Though, initially these applications were written in Java3D before JOGL was available. We learned an invaluable lesson on using emerging technology before it became an industry standard, when Sun no longer supported Java3D. The 3D graphic applications are part of the CSEO framework and are deployed over internet onto the client using Java Web Start.

3D Graphic Application Software Architecture

3D graphic’s applications in CSEO are developed around the Model-View-Control (MVC) architecture using the Object Oriented Design as shown in Figure 12. The Model component consists of application data - comprised of model data such as atoms, molecules, proteins and visualization data. The View component consists of the 3D graphics objects, which handles the rendering of the graphics objects. And the Control refers to the user interaction through the GUI and the mouse.

The Object Oriented Design of the software allows sharing the Model and View component classes among all 3D graphic applications in CSEO, in spite of JOGL not being inherently object oriented. The common classes also results in interoperability among various graphic applications, apart from decreasing the development effort. The graphics APIs consists of common classes for the Graphics-Canvas/Renderer and Graphic Objects. Individual applications consist of classes derived from the common classes. The common base classes assure compatibility among various applications, and enforce standards. The Graphic Objects consists of functions for defining the graphic scene and picking process, a new graphics class is derived from the generic Graphic Object and its scene/picking/other functions are over-ridden. These derived graphics objects are delegated to the graphics-canvas, which manages the scenes. The Graphic Objects use Display Lists (OpenGL feature) for optimizing OpenGL rendering of scenes. The mouse handling functions in the Graphics-Canvas is over-ridden to implement any application specific mouse interaction.

Current 3D graphic applications

MolBuilder: This tool allows users to build a 3D molecule from atoms and fragments. It can also generate 3D structure from 1D SMILES 42 and InChI 43 notations as well as can read in various input formats for 3D data and
allows for modification of the geometry. MolBuilder generates structural data that can be imported to GUIs of all relevant applications such as for quantum chemistry programs.

PsiViewer: This 3D graphic tool allows for visualization and analysis of results from quantum chemistry calculations. In addition to viewing molecular species in different modes such as sticks, ball&stick, overlapping spheres, it can also plot iso-surface of electron density and molecular orbitals, density of states, IR spectra, animate vibrational modes, as well as monitor an optimization process.

BioViewer: This 3D graphic tool is for viewing biological systems such as proteins, DNA, and RNA. BioViewer accepts PDB files as input and has a link to the PDB database. It can display the structure in various forms like ball&stick, stick only, space filling, ribbon, and C-backbone. It also allows for selection of a subset for display in a different mode.

CrystalViewer: This 3D graphic tool allows for visualization and analysis of crystal structure. It can generate all equivalent positions of a given crystal symmetry from the unique ones of the unit cell. Currently, it can read structural data in the Cambridge Structure Database (CSD) 44 format and view the crystal structure in different modes and perspective views.

5. COMMUNICATION COMPONENT

CSE-Online provides capabilities for both synchronous and asynchronous modes of communication. For asynchronous communication, an email Handler tool based on the Pooka tool 45 that has the POP and IMAP support for incoming mails. It can also handle sending mails using the SMTP protocol. In addition, the communication component also has a User Feedback system that allows a user to communicate with the developers or other users on issues associated with a particular application tool. For synchronous communication, we have developed a CSEO Messenger that allows users to have both text and audio chat with other users.

User Feedback System

The User Feedback System allows users to take a snapshot of any particular tool in the CSE-Online environment, make a note on the picture, save all associate data needed to recreate the same state of the tool, and write comments to send to others by email or to send to the development team. This provides an effective
mechanism for users to communicate directly with the development team to give feedback or bug reports, and suggestions for further improvements. Figure 13 aside shows a snapshot of the Sander GUI with a note made by the user to request changing a parameter on the GUI. It is interesting to note that for most if not all scientific software developments, no such automated feedback mechanism exists.

**CSEO Messenger**

CSEO Messenger will enable users to communicate to each other via text, audio or video regardless their geometric positions. Currently, it provides instant text and audio exchange among users on the internet. CSEO communication main design principles include platform independence, real-time transmission and system tolerance. The software architecture of the CSEO Messenger is shown in Figure 14. The system is built on the well-known Java Media Framework (JMF). JMF provides a unified architecture and messaging protocol for managing the acquisition, processing, and delivery of time-based media data. By exploiting the advantages of the Java platform, JMF provides a common cross-platform Java API for accessing underlying media frameworks. Moreover, JMF satisfies our real-time demand as it is built on Real Time Protocol (RTP). Figure xx shows the software architecture of the system. We provide both the Peer-to-Peer (P2P) and via a Relay Server modes of communication. This is necessary because while the P2P mode can reduce the load of the server and the network by direct connection between the peers and is used as the primary operation mode, P2P connections between the peers over the internet are not always possible. In our system, we used the “hole punching” technique to establish the P2P User Datagram Protocol (UDP) -connection between two peers. Currently, about 82% of Network Address Translators (NATs) support such a hole punching for UDP and 64% for Transmission Control Protocol (TCP) connection. The remaining NATs that do not support such a hole punching require the use of a Relay Sever as an alternative mode of operation. In this Relay Server mode, TCP protocol is used to transfer of text and control data to ensure the reliability whereas RTP/UDP protocol is used for audio/video data. The use of a Relay Server is also essential for future expansion of the CSEO Messenger to allow for cyber-conference to facilitate collaborations among users.
The future work of CSEO communication will focus on building virtual classroom, shared windows and remote control so that users can exchange not only text and audio but also images, video and especially remote manipulation of the data or even the application tool.

6. RESOURCE MANAGEMENT SYSTEM

CSE-Online currently supports three modes of running an application: 1) on the login private server; 2) on another remote Unix/Linux computer or cluster; and 3) on a Globus-enabled computing grid such as the TeraGrid. Our current resource management system consists of the Queue manager which is the front-end to the computing environment accessible to the user from within the CSE-Online environment. This GUI allows users to work with any PBS enabled server using a user-friendly and intuitive interface rather than having to use the command line tools provided by PBS. The tool allows for submission, deletion, listing of jobs and other PEB functionalities. Users can transfer files to and from the PBS server to their local machines or the server they were initially logged on to through the CSEO login. A complete list of PBS functionalities provided by the queue manager is as below.

1. Submission of jobs
2. Deletion of jobs.
3. Viewing details of job.
4. Order/Move/Hold or Release a job.
5. View output files.

The queue manager tries to automatically detect whether the server that you are connected to is PBS enabled. If it cannot find such a service the user can use the preferences option to set the location of the PBS commands. The tool also allows for submission of jobs to any queue on the PBS server if it has more that one queue.

The queue manager has also been expanded so that users can submit jobs to the TeraGrid. Users can login to any of the interactive TeraGrid nodes and submit jobs. The queue manager uses the cog-toolkit from Globus technologies to submit jobs to any Globus enabled grid machine. This toolkit makes use of the Java web-
services architecture to submit jobs to grid (Globus enabled) machines from the local computer. Since this toolkit was written with Java interfaces, it can be incorporated into the CSE-Online desktop environment.

7. CONCLUSION AND PERSPECTIVE OUTLOOK

The CSE-Online provides a unique and ubiquitous desktop environment for research, education, and collaboration in computational science and engineering. It can be accessed from a web-browser regardless of geophysical location or time zone and does not require any installation beyond the Java Runtime Environment on the user local computer. It is an extendable, platform-independent, and secured environment that enables a seamless interface and data flow between a variety of different computational application tools and databases. It provides a complete bottom-up environment that integrates methods from different disciplines while having the look-and-feel of a single desktop application. It has different modes of operations such as the connection to a private server, without connection to a private server, and off-line modes to support different working environments and styles. It allows the user to access remote data and computing resources in a transparent manner. Its knowledge management system provides different mechanisms for the user to manage scientific data in the form of files or databases as well as share certain data with other designated users. The environment also provides different communication tools, in both synchronous and asynchronous modes, to facilitate training, user support, and collaboration. Its 3D graphic tools can provide real time performance for visualization/analysis of data from remote sources.

CSE-Online open software architecture allows third-party developments of different desktop application tools. Furthermore, the desktop applications are loaded at runtime, thereby allowing the user to select only applications that are needed to build the executable. From the software engineering point of view, this architecture allows the development to be scalable and the desktop environment to support unlimited number of applications while the memory requirement of the environment depends only on those selected by the user.

CSE-Online at the beginning is only an online desktop environment for computational science and engineering, however, it is evolving into a middleware that can provide a framework for accessing and managing
remote data and resources including the computing grid for any domain, not necessarily within computational science and engineering.

ACKNOWLEDGEMENT

This project is supported by the National Science Foundation under the Information Technology Research Initiative. We are grateful to our Advisory Board members, Professors Jack Simons, Monte Pettitt, Adel Sarofim, Gary Lindstrom, Martin Berzins, and Chris Johnson, and Drs. Jeff Nichols and Max Tirtowidjojo for their guidance.
References


(28) Computational Science and Engineering Online (CSE-Online), [http://cse-online.net](http://cse-online.net).


(30) Fielding, R.; Gettys, J.; Mogul, J.; Frystyk, H.; Berners-Lee, T. Hypertext Transfer Protocol -- HTTP/1.1, RFC 2068, **1997**.


(33) JME Molecular Editor, [http://www.molinspiration.com/jme/](http://www.molinspiration.com/jme/).


(35) Postgres SQL, [http://www.pervasive-postgres.com](http://www.pervasive-postgres.com).

(37) National Cancer Institute small molecule database,


(39) Virtual Reality Modeling Language (VRML), http://www.w3.org/MarkUp/VRML/.


(41) Java OpenGL (JOGL), http://jogl.dev.java.net.

(42) Simplified Molecular Input Line Entry Specification (SMILES),

(43) Stein, S. E.; Heller, S. R.; Tchekhovskoi, D. In An Open Standard for Chemical Structure
Representation: The IUPAC Chemical Identifier, in the Proceedings of the 2003 International Chemical


(47) Ford, B.; Srisuresh, P.; Kegel, D. In Peer-to-Peer Communication Across Network Address Translators,
in the Proceedings of The 2005 USENIX Annual Technical Conference (USENIX ’05), Anaheim, 
California, 2005; Anaheim, California, 2005; p. 179-192.

Figure captions

Fig. 1: A plot of the overall software architecture of CSE-Online.

Fig. 2: A plot to show the active components of the ‘connection to a private server’ operational mode.

Fig. 3: A plot to show the active components of the ‘without connection to a private server’ operational mode.

Fig. 4: A plot to show that only the client environment is the active component in the off-line mode.

Fig. 5: A plot of the software architecture of the user desktop environment. Different components of the environment and their connections to the server components are also shown.

Fig. 6: A snapshot of the user desktop environment that shows active windows of several applications.

Fig. 7: A snapshot of the File Manager.

Fig. 8: A plot of the software architecture of the Multi-level Database Management System (MDMS).

Fig. 9: A snapshot of the MDMS connecting to the Protein Data Bank and have the retrieved data viewed by the BioViewer tool.

Fig. 10: A snapshot to show the capability of MDMS to connect to the NCI public database. The selected data can be retrieved and viewed by any appropriate tools available within the CSE-Online environment.

Fig. 11: A snapshot of the prototype of the eDoc system displaying an electronic document with embedded data.

Fig. 12: A plot of the software architecture of the 3D graphics application tools.

Fig. 13: A snapshot of the User Feedback System.

Fig. 14: A plot of the software architecture of the CSEO Messenger for text and audio/video communication between two users.
FIGURE 1
FIGURE 3
FIGURE 5

This figure illustrates the architecture of a desktop environment and its interaction with a Java virtual machine/local CPU on the client side. It also shows the communication and data transfer processes between the client and server sides. The diagram includes components such as file managers, programs, databases, resource managers, and communication tools, along with various communication protocols like SSH and Web services. The private server and computing nodes are also depicted, showing the interaction with remote computing nodes and grid servers.
FIGURE 10
FIGURE 11

The Electronic Notebook

An electronic notebook can incorporate text, as seen in this page, an index of the overall document, seen on the left, and a large collection of questionable metadata, seen on the right.

An electronic notebook can also contain data. This can be seen with the energy plot shown in the right-hand window.
FIGURE 12

Diagram showing the components of an application:
- View
- Control
- Model
- DB
- Server

Application structure:
- View connected to Control
- Control connected to Model
- Model connected to DB
- Model connected to Server
FIGURE 13