The Extensible Computational Chemistry Environment: A Problem Solving Environment for High Performance Theoretical Chemistry

Gary Black, Karen Schuchardt, Debbie Gracio, Bruce Palmer

Pacific Northwest National Laboratory
Richland, Washington 99352

Abstract. The Extensible Computational Chemistry Environment (Ecce) is a suite of distributed applications that are integrated as a comprehensive problem solving environment for computational chemistry. Ecce provides scientists with an easily used graphical user interface to the tasks of setting up complex molecular modeling calculations, distributed use of high performance computers, and scientific visualization and analysis. Ecce's flexible, standards-based architecture is an extensible framework that represents a significant milestone in production systems, both in the field of computational chemistry and problem solving environment research. Its base problem solving architecture components and concepts are applicable to problem solving environments beyond the computational chemistry domain.

1 Introduction

Addressing the complex environmental issues facing the nation today requires new scientific understanding of the fundamental chemical, physical, and biological processes underlying these issues. We must understand these phenomena at the basic molecular level. Modeling and simulation techniques have advanced to an accuracy that is quickly approaching that of experimental processes. An integration of experimentation with modeling and simulation will support scientists' understanding of the behavior associated with these systems to solve grand challenge problems.

The Extensible Computational Chemistry Environment (Ecce), developed at the William R. Wiley Environmental Molecular Sciences Laboratory (EMSL) at Pacific Northwest National Laboratory (PNNL), is the first comprehensive, integrated, problem solving environment developed to support computational chemists. Ecce is a seamlessly integrated suite of distributed client/server applications. Ecce enables scientists to easily use computational chemistry software to perform complex molecular modeling and analysis tasks by accessing distributed, high-performance computers from their desktop workstations. Ecce includes a sophisticated graphical user interface, scientific visualization tools, and the underlying data management framework, providing scientists with the tools to efficiently set up calculations, store and retrieve results, and analyze the rapidly growing volumes of data produced by computational chemistry studies.

Ecce includes support for building molecular models; a graphical user interface to a broad range of electronic structure theory types; remote submission of calculations to UNIX and Linux workstations, clusters, and supercomputers; three-dimensional visualization and graphical display of molecular data properties; and extensive web-based help. Ecce applications can be divided into the following categories based on usage:

Organizational: The Ecce Gateway provides central access to the core tools within the Ecce application suite as well as access to online help and support options. The Calculation Manager allows researchers to organize and manipulate computational chemistry studies in Ecce. This application provides an at-a-glance overview of the status of every calculation, and easy access to key setup parameters and run statistics.

Setup: The Molecule Builder enables researchers to build, visualize, modify, and manipulate 3D images of chemical systems to be used in Ecce calculations. Chemical systems may be imported/exported in standard file formats such as XYZ and PDB. The Basis Set Tool supports over 230 predefined Gaussian basis sets for the user to select from or create new basis sets for use in *ab initio* electronic structure calculations. For a given chemical structure, a list of available basis sets is displayed for selection. The Calculation Editor provides the user with application-specific options needed to perform a calculation. As selections are made and parameters set, the editor automatically enforces valid inputs. The editor provides the final translation to the required input format for each computational code supported.

Job Launching: The Launcher submits the calculation to a compute resource after performing validation checks against user input and generating a job submission script. The Machine Browser displays reference information about compute resources known to Ecce and performs queries for overall machine, job, and disk status to aid selection of an appropriate resource to run a job.

Monitoring/Analysis: The Calculation Viewer provides access to current information for a single calculation during and after execution. The viewer supports computation of molecular orbitals, animation of normal modes, display of geometry optimization, Mulliken charges, and much more. The jobstore and jobmonitor applications are non-graphical interface processes that link the workstation running Ecce applications and the compute resource for monitoring job status and parsing output properties.

This paper provides an overview of the Ecce architecture and highlights many of the key technologies used to enable flexibility within an ever-expanding suite of software tools. Collaborations with outside projects and future directions are also summarized.

2 Ecce Architecture

The Ecce architecture seeks to achieve a number of often competing goals. These include creating a loosely coupled, component-based design; providing secure extensible access to any computational resource; leveraging existing networking services (rather than requiring the installation and administration of new ones); supporting the registration of additional computational codes with minimal programming and in a way that takes advantage of the rest of the architecture; and designing for the flexibility to incorporate new technologies and techniques. In addition, since the initial development of Ecce there has been a movement toward non-proprietary and standards-based software bolstered by many freely available and open source projects. Ecce's architecture is evolving to assimilate these standards and projects. A major new goal of Ecce is to remove remaining dependencies on proprietary products so that Ecce can be extended by a much wider developer base.

Fig. 1 shows a high-level operational view of the Ecce architecture. A suite of desktop applications assist the researcher with calculation setup, job execution, and results analysis. These components, acting independently, each augment a web-enabled data store with data and metadata, communicating indirectly via a central messaging system. Standard network services are used to submit jobs to a wide variety of queuing systems as well as workstations. The chemistry codes currently supported are NWChem, Gaussian 98, and Amica. Ecce is written primarily in C++ but also makes appropriate use of Perl, Java, and Python, as will be discussed. The following sections describe the core components of the Ecce architecture, illustrating why Ecce is a significant milestone in production systems, both in the field of computational chemistry and problem solving environment research.

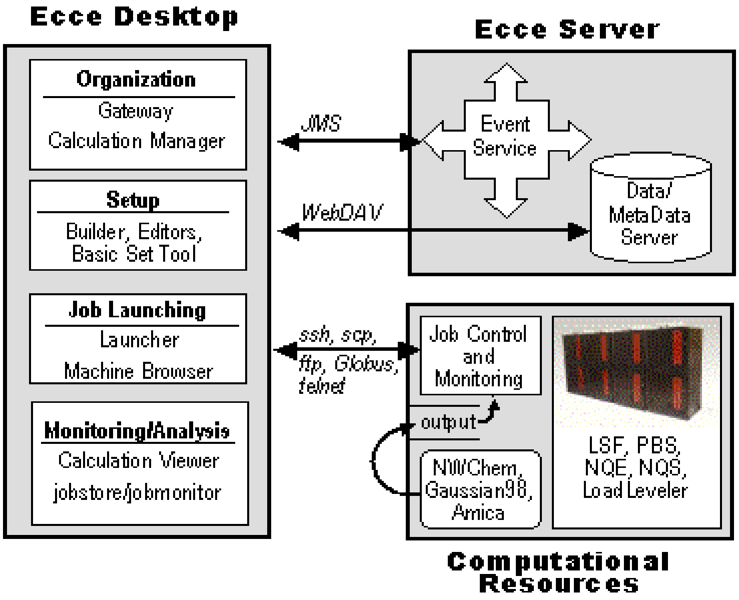


Fig. 1. Operational view of Ecce architecture

2.1 User Interface and Visualization

Two commercial tools for developing Motif applications are licensed by the Ecce project. They are TeleUSE from Aonix Corporation for laying out interfaces and generating source code to create them, and XRT PDS from Sitraka for advanced user interface widgets. Ecce has developed a base set of reusable interface objects (within TeleUSE these are referred to as templates) that can be used to more quickly create applications conforming to the standard Ecce look and feel. These templates are instantiated as C++ objects, hiding much of the underlying complexity of traditional Motif GUI development. The XRT PDS widget set provides objects such as spreadsheet-like tabular data entry and display, hierarchical folder/document tree views, and two-dimensional XY graphs.

Visualization is based on the Open Inventor C++ class library developed by SGI now available as an open source project. Open Inventor is implemented on top of OpenGL, the industry standard for high-performance visualization, and uses a scene graph paradigm to considerably reduce the complexity of visualization development. Molecular Inventor, an API that sits on top of Open Inventor, supports display and manipulation of atoms, bonds, and iso-surfaces.

2.3 Data Management

The data management component of Ecce is a ground-breaking achievement that addresses some of the inherent problems of managing scientific data with traditional database approaches such as relational database management systems or custom file-based implementations. Ecce uses the Web Distributed Authoring and Versioning protocol (WebDAV) [1], a set of extensions to the HTTP protocol that supports document retrieval, storage, and editing, as well as the annotation of the data with arbitrary metadata. Documents are used to store input setup information (the molecule, basis set, input files) and results, while metadata is used to annotate data to make it more meaningful to query engines or other non-Ecce applications. Documents can be in any format. Ecce uses a combination of formats native to the computational codes and XML for its flexibility in describing scientific data and the wide availability of tools. Because of the WebDAV-based architecture, Ecce data sets are directly available to users through their web browsers. Since WebDAV is a standard, many alternative implementations are available, allowing scalability decisions to be made at deployment time. As currently distributed, Ecce uses an Apache 2.0 server with the mod_dav module. A more complete discussion of the Ecce data management architecture can be found in [2].

2.4 Messaging

Ecce applications are able to work together while achieving the architectural goal of loose coupling, through the messaging or event layer. The event architecture allows individual applications to notify subscribers of its actions without direct knowledge of other applications. It also provides basic capability for collaborative research and supports the addition of agents without impacting existing functionality. Ecce has adopted the Java Messaging System (JMS) standard for its basic messaging engine. JMS is a subscription-based, publish-subscribe service where applications declare an interest in receiving notifications matching certain patterns, and the JMS server filters and routes these messages. JMS is a standard with multiple implementations. It runs on any platform and non-Java bindings are available. Java Open Reliable Asynchronous Messaging (JORAM) was selected as our JMS server.

2.5 Computational Code Registration

Ecce provides a framework for registering new computational codes through structured script writing rather than reworking core C++ applications. Registering a code with Ecce consists of developing three components: 1) user interface detail dialogs with Python; 2) application input file generation with Perl; and 3) output log file parsing with Perl to extract properties for analysis. The design of code registration allows it to be done by a scientist with expertise in the computational code.

Ecce uses a set of Python classes that extend the freely available Qt user interface toolkit, PyQt. The Ecce Python classes define the base interface objects such as numeric input with range validation, option menus, and toggles used to create “detail dialogs.” Detail dialogs are code-specific dialogs to set input options such as convergence criteria or algorithms to apply.

To accomplish input file generation, a Perl script (or other program) is written that is responsible for turning user inputs (molecule, basis set, options specified in detail dialogs) into a valid input file.

Output file parsing is the final aspect of code registration. Both server- and client-side parsing is done. On the computational server, chunks of raw data are extracted with keyword matches as the output log file is being generated, providing a filtering capability. This data is sent back to the client machine where the Ecce application software is running. A parser on the client side takes this raw property data and translates it into a standard XML file format defined by Ecce based on the dimensionality of the data. The client-side parsing is handled by writing separate Perl scripts, typically short, for each property. The Ecce Calculation Viewer provides default displays and visualizations based on data dimensionality.

2.6 Remote Communication

All communication with distributed compute resources is handled through an Ecce C++ class. This class is a high-level robust implementation for managing remote command shells, single commands, and file transfer on UNIX/Linux platforms. Specifically, the RCommand class is used for copying files to and from computational servers, submitting jobs, collecting job status and ongoing output data, and handling various other file and job management tasks. RCommand wrappers the industry standard remote shell and remote copy commands, presently including secure shell and secure copy (ssh/scp), telnet and ftp, rsh and rcp, and the Globus Toolkit 2.2 [3]. The RCommand class treats these remote shells as “black boxes”—it is not intrinsically linked or otherwise dependent on libraries, services, daemon processes, or socket-level protocols supporting remote access. Use of either secure shell and secure copy or the Globus Toolkit adds all the authentication and encryption security normally afforded these commands to any software using the RCommand class. RCommand is unique as a fast, lightweight, standards-based mechanism for building remote access into UNIX software and allows Ecce to deploy to virtually any infrastructure without imposing new administrative tasks.

2.7 Security

The security model within Ecce is vital to maintaining the confidentiality of calculation data, and even more importantly, preventing access to remote systems that are configured to run jobs. Ecce uses a single-point login, chosen by the user and referred to as a passphrase. This passphrase is used as the decryption key for all other passwords that are stored as encrypted values using the Blowfish encryption algorithm [4], thus achieving the appearance of single signon from the user’s perspective. The passphrase is maintained as an encrypted value unique to every Ecce session, stored only in the memory of Ecce applications. To maintain this level of security, it is strongly recommended that some form of secure remote communications shell be used—such as ssh, Globus, or Kerberized rsh—though Ecce maintains support for traditional protocols such as telnet and rsh.

2.8 Compute Resource Registration

Ecce supports remote submission of jobs to UNIX and Linux workstations, clusters, and supercomputers with built-in support for LoadLeveler, Maui Scheduler, NQE, NQS, PBS, and LSF, as well as workstations. Ecce also provides a mechanism to integrate new queue management systems through script files. Experience shows that each site uses queue management systems in unique ways to support local features and requirements. Through configuration files, Ecce allows complete customization of job script generation.

A GUI application, which can be used by an Ecce administrator for shared resources or by individuals for their own resources, is the primary means of making compute resources known to Ecce. The machine registration application collects reference information about the type of machine, number of processors, paths to computational codes, and supported remote communications shells. Queued machines additionally require resource limits for each queue such as time, memory, and processors.

2.9 Job Launching and Monitoring

Launching a job to a compute resource is a straightforward process, made so by its reliance on the Ecce architecture components for remote communication and compute resource registration. Using the Job Launcher application, a user selects from available machines, overrides default request values such as number of processors, and presses the “Launch” button. Ecce creates a remote connection to the resource, performs various checks, transfers files, invokes the job submit script, and initiates the applications that provide job monitoring functionality.

The term job monitoring refers both to monitoring the status of a job in the queue management system (idle, running, complete) or process table for workstations and to providing near real-time analysis of results while the code is running. Ecce extracts, parses, stores, and visualizes any output data immediately as it is generated through passive file monitoring techniques, allowing the user to monitor progress and take corrective action. The client-side monitoring process, jobstore, is a C++ application that is responsible for dynamically installing the compute server monitoring script, jobmonitor, re-establishing connections when faults occur, and posting all output data and files to the data server.

3 Ecce Collaborations

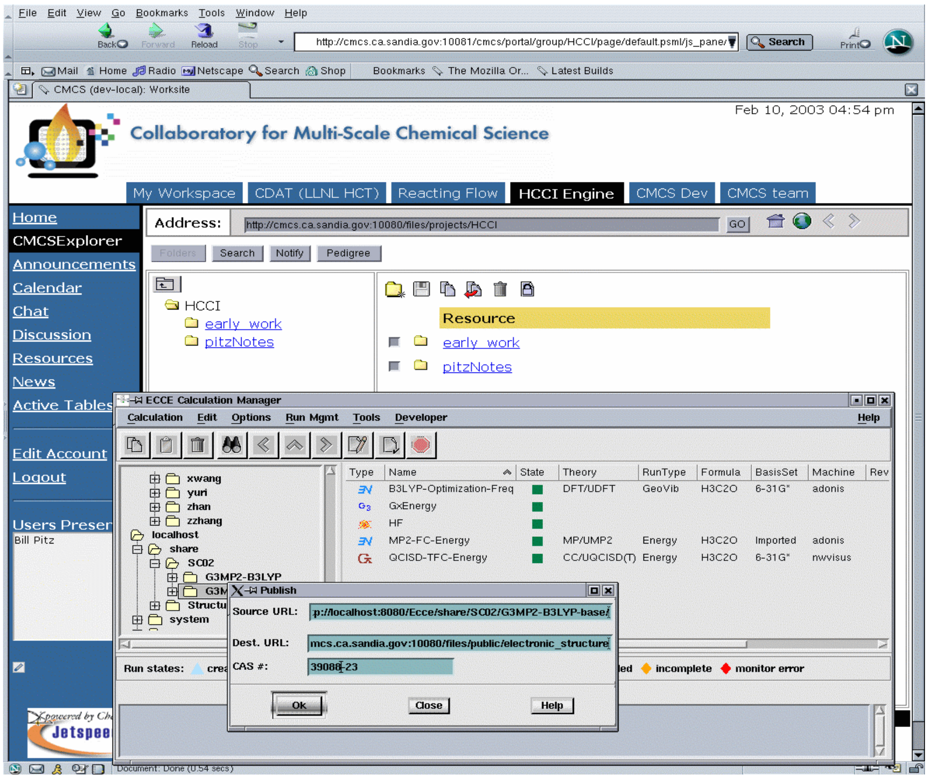
The Ecce project participates in number of collaborations. The most extensive collaboration is with the NWChem project with current emphasis on adding support for molecular dynamics (MD) calculations. Efforts to date have focused on construction and validation of large proteins. The Ecce Molecule Builder has been optimized to support systems with hundreds of thousands of atoms. The Builder’s MD Toolkit supports manipulation of residues and validates structures against known residue datasets, assigning atom types and partial charges. Tools for editing force field parameters and constructing the topology are nearing an initial release. Future plans call for an MD Calculation Editor for input file setup, job execution, and archiving trajectory files. Support for viewing trajectory files and creating movies exist today.

A recent successful collaboration with the University of Utah resulted in the integration of Amica, a code for very high accuracy electronic structure energy calculations, into the Ecce framework. The effort entailed developing custom detail dialogs, input file generation, job submit script generation, and the extraction of selected properties for viewing in the Calculation Viewer. In addition, capabilities added to the Basis Set Tool give researchers significant control over construction of custom basis sets using the extensive EMSL reference basis set library.

An ongoing collaboration exists with the Collaboratory for Multi-scale Chemical Sciences (CMCS) project. The objective of CMCS is to provide tools that enhance chemical science research among multidisciplinary teams of researchers. For example, combustion researchers require expertise and data from thermochemists, kineticists, and molecular modelers. CMCS, using data middleware provided by the

Scientific Annotation Middleware (SAM) project, is developing an adaptive informatics infrastructure and a web portal environment that integrates chemistry applications, data resources, and collaboration tools to achieve this objective. The initial integration of Ecce with CMCS focuses on integration at the data level. Consider a scenario in which a researcher developing chemical mechanisms has trouble matching experimental data. A sensitivity analysis indicates that the result is sensitive to the thermochemistry of a particular chemical species. The researcher uses CMCS to seek out data with lower levels of uncertainty. If the data does not exist, CMCS connects the researcher with a molecular chemist who provides the desired results by using Ecce.

CMCS is defining standards for annotating data with pedigree, allowing researchers to search, categorize, and trace data across scientific disciplines and back to its original source. Fig. 2 illustrates a capability to export from Ecce to CMCS with CMCS pedigree information attached. In this example, a molecular chemist computed a heat of formation for the scenario described above. Calculating this one value required several sub-calculations. When the data is exported to CMCS, the complete



http://cmcs.ca.sandia.gov:10081/cmcs/portal/group/HCCI/page/default.psml/js_pane/

Feb 10, 2003 04:54 pm

Address: http://cmcs.ca.sandia.gov:10080/files/projects/HCCI

Type	Name	State	Theory	RunType	Formula	BasisSet	Machine	Rev
ƎN	B3LYP-Optimization-Freq	🟩	DFT/UDFT	GeoVib	H3C2O	6-31G*	adonis	
G3	GxEnergy	🟩						
	HF	🟩						
ƎN	MP2-FC-Energy	🟩	MP/UMP2	Energy	H3C2O	Imported	adonis	
G	QCISD-TFC-Energy	🟩	CC/UQCISD(T)	Energy	H3C2O	6-31G*	nwvisus	

Source URL: p://localhost:8080/Ecce/share/SC02/G3MP2-B3LYP-base/

Dest. URL: mcs.ca.sandia.gov:10080/files/public/electronic_structure

CAS #: 39088-23

Fig. 2. Exporting results from Ecce to the CMCS Community Portal

sequence with all data is provided and full pedigree information is added to allow other researchers to examine the steps used to create the value. This prototype scenario was demonstrated at the IEEE/ACM Supercomputing 2002 conference.

Finally, since CMCS also uses the WebDAV protocol for its data layer, Ecce can be run directly against a CMCS server. Users have immediate web browser access to their data via translators and viewers integrated within the CMCS portal environment.

4 Future Directions

Future development in Ecce will focus both on extending the chemistry domain functionality and improving and enhancing the underlying architecture. New domain functionality will continue to focus on molecular dynamics support and the development of a workflow-oriented interaction model that allows researchers to express and execute higher order operations that inherently require many computational steps. Examples include evaluations of the Basis Set Superposition Error (BSSE) and fragment guess calculations.

Extensions to the architecture include the replacement of the Motif development environment with a multiplatform, Python-extended toolkit such as Qt or GTK, integration with an archive for larger data requirements of molecular dynamics, integration with new Open Grid Services Architecture (OGSA) [5], and continued adoption of CMCS and SAM technologies to support collaboration and web-based access to Ecce services.

Several Ecce applications are useful independent of the Ecce suite and will be distributed individually. These include the Builder, Basis Set Tool, and Calculation Viewer. The Builder will be available as a standalone distribution in spring 2003.

5 Summary

Ecce has matured from a test bed of prototype, standalone computational chemistry software in the mid 1990’s to an integrated suite of tools heavily reliant on proprietary technology in the late 1990’s to a robust, flexible, standards-based, globally available framework for ever-expanding computational chemistry capability development today. As Ecce follows this evolution, projects outside the domain of computational chemistry will be able to utilize the base problem-solving architecture components and concepts. With early collaborations already demonstrating results and with Ecce continuing to push new boundaries with support for user-controlled workflow, complete script-driven application development, Grid interoperability, and web integration, the promise for this pioneering problem solving environment continues to grow.

Ecce currently runs on Linux, Sun, and SGI workstations. The software suite is available to researchers through our website at http://ecce.emsl.pnl.gov. To download the software, an EMSL software user agreement must be signed by the requesting institution.

6 Acknowledgements

Pacific Northwest National Laboratory is operated by Battelle for the U.S. Department of Energy under Contract DE-AC06-76RLO 1830. The William R. Wiley Environmental Molecular Sciences Laboratory, located at PNNL in Richland, Washington, is a national scientific user facility sponsored by the U.S. Department of Energy's Office of Biological and Environmental Research.

We would like to acknowledge the valuable discussions and interactions with the development team of the computational chemistry code NWChem.

We would like to acknowledge our ongoing collaboration with the Collaboratory for Multi-scale Chemical Science project and the Scientific Annotation Middleware project, both funded by the DOE Scientific Discovery through Advanced Computing Initiative. The CMCS website is located at http://cmcs.ca.sandia.gov. The SAM website is located at http://www.emsl.pnl.gov:2080/docs/collab/sam.

We would like to thank Dr. Robert Gdanitz formerly at University of Utah, now at North Carolina Agricultural and Technical State University, for his help in the integration of Amica as a computational code that Ecce supports. The Amica website is located at http://gdanitz.hec.utah.edu/amica.

References

1. Web Distributed Authoring and Versioning Protocol, http://www.ietf.org/rfc/rfc2518.txt.
2. Schuchardt, K.L., Myers, J.D., Stephan, E.G.: A Web-based Data Architecture for Problem-Solving Environments: Application of Distributed Authoring and Versioning to the Extensible Computational Chemistry Environment. Cluster Computing 5(3) (2002) 287-296.
3. The Globus Project, http://www.globus.org/.
4. Schneier, B.: Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish). Fast Software Encryption, Cambridge Security Workshop Proceedings (December 1993). Springer-Verlag (1994) 191-204.
5. Tuecke, S., Czajkowski, K., Foster, I., Frey, J., Graham, S., Kesselman, C.: Grid Service Specification. Open Grid Service Infrastructure Working Group, Global Grid Forum, Draft 2, 7/17/2002.