

## Secure Cloud Computing in the Chemistry Laboratory: A Budget-Friendly Approach to Computational Work

Jon R. Serra  
Department of Natural Sciences  
The University of Pittsburgh at Titusville  
504 East Main Street  
Titusville, PA 16354  
814.827.4435  
serra@pitt.edu

### Abstract

The design and implementation of a web-based, cloud computing system at a small undergraduate college will be discussed. This cloud uses freely available and low cost software to create a robust, user-friendly environment for high-level chemical structure modeling free of licensing limitations. The web-based implementation allows users to access the system from any computer connected to the Internet by entering a common web-portal into a server machine that then relays computational chemical modeling calculations to remote compute nodes via a local area network. The configuration of the web-based portal, network design and limitations, and some freely available computational chemistry packages will be presented from the system administrator's perspective. Portal usage and some common computational chemistry exercises are presented from the end user's perspective.

### Background

The proliferation of computers and molecular modeling software in the organic chemistry classroom and laboratory has exploded over the past several years.<sup>1-2</sup> Many organic chemistry textbook publishers have begun to incorporate proprietary and popular molecular modeling software packages with their lecture textbooks and laboratory manuals.<sup>1,3</sup> Several stand alone courses built around software packages have also been developed by academicians, textbook publishers and software manufacturers.<sup>2</sup> The Internet is riddled with tutorials and guides to perform basic and complex computational experiments. These tutorials, however, can be very specific, proprietary and even out of date at times. With so many choices, selecting a computational chemistry software package for an organic chemistry course can be a daunting task.

Very broadly defined, computational chemistry uses computer software to elucidate a chemical problem.<sup>4</sup> A branch of computational chemistry, molecular modeling, uses software packages to compute atomic coordinates by finding a lowest energy conformation or representation of a molecule or ion.<sup>4</sup> Once a conformation is discovered, many complex chemical and physical properties of molecules can be calculated. These include: heat of formation, electrostatic potential, highest occupied molecular orbitals and lowest occupied molecular orbitals, amongst many others.

Several very popular commercial software packages exist to perform these types of studies. Quite possibly the most popular software package used in research and industry is the Gaussian software package.<sup>5</sup> This package allows for very robust calculations on a wide variety of molecules and ions to be determined via a command line or graphical user interface. As the industry standard, Gaussian remains the computational software package of choice for many organic chemistry courses at major research universities. Since many research universities have encompassing site licensing agreements for Gaussian this is a natural fit. But what software is right for a small college organic chemistry classroom? To answer this, one must consider four questions: (1) Is this software package affordable? (2) Is this software package user friendly? (3) How many students will be using this software and where? and (4) How easy is this software package to maintain? To begin, consider two of the most popular computational chemistry packages available, Gaussian and Wave Function's, Spartan Pro.

Gaussian can be prohibitively expensive for many small college organic chemistry instructors. A single academic license can cost thousands of dollars and a multi-user or multi computer site license can be much more. The standard Gaussian package does not include a graphical user interface that must be purchased separately, increasing the cost. A single license of Gaussian will allow installation on only one computer making it very difficult for multiple students to use the program simultaneously. However, if cost is not a factor, then a site license option exists that allows for installations of the software on multiple computers. Gaussian is very easy to maintain as many service agreements are available and upgrades can be as simple as pushing the "check for updates" button as with many other software packages. So, if cost is not an issue, Gaussian can be a very attractive alternative.

Spartan Pro<sup>6</sup>, available from Wave Function, can also be a very attractive candidate. A single user license, which allows for one complete installation on a single physical machine can cost over \$1000 with a multi-user, multi-processor option increasing the cost significantly. A very attractive feature of Spartan is that many textbook publishers offer a limited student version of Spartan as a bundled supplement to their textbook. However, this usually increases the cost of the textbook and therefore the cost to the students. One of the greatest features of Spartan is its graphical user interface. Students can easily draw molecules and perform computational tasks with very little guidance. However, students must physically access the computer running Spartan. If many students are accessing the software simultaneously, then each student would need a computer running Spartan. Like Gaussian, Spartan is also very easy to upgrade and maintain. If Spartan is bundled with the chosen organic chemistry textbook, or if the software can be easily installed in a campus computer lab, then Spartan may be a viable alternative.

If cost, ease of use for students and student accessibility are paramount issues, then a web-based cloud-computing platform may be an attractive alternative. The platform outlined here uses several open-source and shareware software packages over a WAN/LAN system to provide students a web-based graphical user interface to draw molecules and perform many computational task at little or no cost to the instructor or institution.

### **Platform Design**

As a student, one accesses the platform via a cloud-computing web-based interface using a unique username and password. Then molecules can be drawn and edited using a Java based editor and parameters can be configured for the computational task chosen and calculations can

be submitted to the server all via the web-based interface. Once submitted, students can check the status of their computational jobs and perform data analysis over the Internet.

Once a student submits a calculation from the cloud interface, the server then creates an input file for the appropriate computational software package and adds the job to a scheduling/queuing system. The job is then submitted to an available remote LAN node for completion. The server then has the ability to check the status of every remote node and reports that status back to the server. Once a remote node completes a computational task, the appropriate output files are copied back to the server for storage and data analysis.

The cloud design includes four steps (1) WAN/LAN server configuration, (2) node configuration,(3) computational software configuration and installation, and (4) web-based interface configuration. This must be performed as needed on either the server or remote computer nodes or both.

To configure the server, it must first be equipped with dual network interface cards, one for the WAN and one for the LAN. Also, a valid external IP address must be provided that allows for WAN http and https access. WAN SSH access can be very advantageous to allow for remote administration. For the implementation described here, Scientific Linux Version 5.2<sup>7</sup> was chosen as the operating system on a Gateway E-Series Intel Pentium IV, 2.0 GHz computer with a 250 GB hard drive and 2 GB of DDR RAM. Scientific Linux is freely available and can be easily configured to run all necessary software. The Gateway E Series computer was a campus surplus computer replaced from a campus laboratory. Scientific Linux version 5.2 was also installed on each remote LAN node.

Once the operating system was installed, the machine was configured to act as a LAN server using the open source application Firestarter<sup>8</sup>. The basic configuration was used, with access only provided to the remote machines via IP address verification. Also, WAN access was restricted to allow only HTTP, HTTPS and SSH incoming WAN connections, and SSH was configured to allow access to only specific IP and MAC addresses. Students access the cloud via HTTP and HTTPS, SSH is only used to access the server for administration and maintenance.

Next the computational chemistry software was installed on the server. For this implementation, the General Atomic and Molecular Structure System (GAMESS)<sup>9</sup> was configured and installed. The GAMESS software package is made available to interested academicians as precompiled platform dependent binary executable files or as source code to be compiled later. Since GAMESS has been developed for years as a research tool, it can be very cumbersome to use. This is overcome by the use of WebMO, a shareware software package that provides the web-based graphical user interface and scheduler/queuing system used in this implementation. The WebMO software package is free available to perform most of the tasks outlined here, but WebMO Pro was purchased that adds many analysis tools as well as the scheduler/queuing system to manage jobs on remote LAN compute nodes. The WebMO Pro software package was installed on the server by following the standard installation instructions found a on their website. A WebMO user and group was created on the server that will be used by WebMO Pro to perform necessary calculations on both the server and remote compute nodes.

The remote compute nodes access the server via a LAN. Currently, this installation contains 8 computers that range from Pentium 3, 350 MHz with 250 MB of SD RAM to Pentium IV, 1 GHz

machines with 1 GB of DDR RAM. All of the remote LAN computers were donated to this project from campus surplus. Scientific Linux version 5.2 was installed on each of the nodes, configured to access the server via SSH and a WebMO user and group was created that was identical to that on the server. The WebMO software requires passwordless exchange of information between the server and nodes for the WebMO user. This was achieved using the RSA protocol within SSH. Finally, the GAMESS computational chemistry software package was installed on each node and the WebMO program on the server was updated to include each LAN remote compute node.

### **WebMO Configuration**

The WebMO<sup>10</sup> installation and configuration is very straight forward, and the directions from the developer are readily available from their website. Briefly, the server software is installed on the server with the supplied Perl script installer. Builds exist for Windows, Macintosh and Linux operating systems; however, in this installation, the Linux version was implemented. WebMO installation is achieved in three steps. First, create a WebMO user and group on the server and each node. It helps to assign the same UID and GID on each machine. Second, install the WebMO software on the server. Finally, configure WebMO to access each node and computational software package available as well as adding users to the WebMO system. Once this is completed, a WebMO administrator account is created within the WebMO software for access via a web portal. This can be different than the Linux account where WebMO will be accessed and for security reasons should be different than the administrator for the machine's operating system.

One point to note, WebMO utilizes a common work area for all jobs submitted. The default for Linux is to use /tmp for this purpose. This can have a disadvantage as Scientific Linux periodically removes files from /tmp that have not been updated or accessed for a specified period of time. Since some calculations can take long periods of time without accessing some or all of the necessary files in /tmp, the operating system could potentially delete a file needed by WebMO causing the WebMO job to crash.

Once the WebMO software is installed on the server, the computational software must be installed on each computer that WebMO will access for job submission. The GAMESS software is available at no cost from the Gordon Research Group at Ames Lab – Iowa State University. Simply follow the guidelines on their web page to gain access to the program. The GAMESS application is available as precompiled binary files for most major operating systems, including 64 and 32 bit versions. For this work, the precompiled binary files of GAMESS for 32 bit Linux was obtained and simply copied to /usr/local in a newly created GAMESS directory.

Next, WebMO is configured to access each node, computational chemistry package, and users added from a web based interface using the WebMO administrator's password. This is a very straight forward process that requires the IP address of each node to be assigned computational jobs, the location of the scratch directory for each node, and the path to the computational chemistry software package. Users can now be added and the entire system tested.

## Example Usages

Many classroom examples could be provided<sup>11</sup>, but fundamentally, most are based on (1) molecular energy calculations, (2) molecular equilibrium geometry optimization and (3) graphically displaying chemical properties.

### (1) Molecular Energy

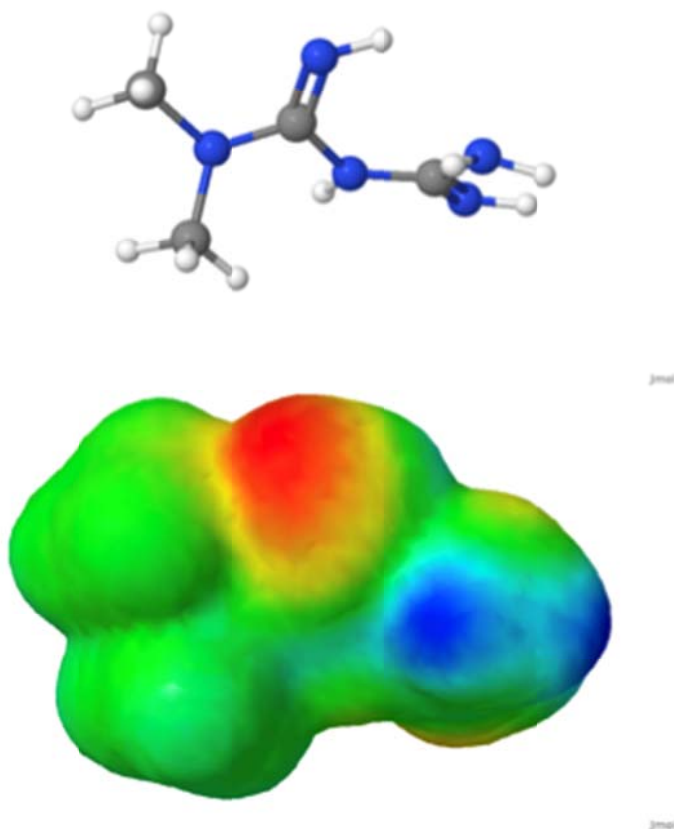
Molecular energy calculations are perhaps the most fundamental application of computational chemistry. In this process, a student would draw a representation of a molecule on the computer using the ball and stick approach. Then, once a reasonable representation has been drawn, the molecule is submitted to the computational chemistry engine (GAMESS in this example) to calculate the energy of the molecule drawn by the student. From this energy calculation, students can then calculate several relevant chemical and physical properties that include dipole moment, atomic charges, bond orders, and molecular orbital analysis.

### (2) Molecular Geometry Optimization

Molecular geometry optimization is a refinement to the molecular energy calculation outlined previously. Here the student-drawn representation of a molecule is submitted to a computational engine (GAMESS in this example) to determine the minimum energy of a molecule by an iterative process. The iterative process performs several molecular energy calculations in sequence by perturbing bond distances and angles and comparing the molecular energy of each. The lowest energy wins. Once a threshold is reached, called the energy minimum, the calculation is finished and chemical and physical properties similar to those outlined above can be calculated. The advantage to this procedure is that mathematical equations are used to determine the energy of the molecule and not the student's original structure, which could be a very poor representation. The major disadvantage is the time associated with this type of calculation. A typical optimization can take many individual energy calculations, as described in the previous section, and can therefore take much longer to complete.

### (3) Graphically Displayed Three-Dimension Potential Energy Maps

Once the energy of a molecule is computed, students can then represent the 3-dimensional structure using potential maps. The potential energy of Metformin, a common diabetic drug, is shown in **Figure 1**. Here the charges of Metformin are overlaid on the molecule to illustrate charged regions. Typically, the highest charged portion of molecules are colored red, and the lowest charges are colored blue. This allows students to visually interpret areas of the molecule where chemistry is most likely to occur.



**Figure 1:** Top, shows the RHF 6-31G(d) representation of Metformin. Bottom, shows the charge potential energy map of the same structure of Metformin.

### Conclusions

Several software packages exist to calculate molecular representations. The WebMO application has proven to be a valuable tool that balances ease of use versus cost to the instructor. While there is a good deal of configuration and maintenance on the part of the instructor, it provides a consistent, user-friendly graphical, web-based interface for students to access research level computational chemistry packages. Using this implementation, organic chemistry students are able to visually interpret fundamental chemical properties such as atomic charges and molecular dipole moments that are commonly used to identify chemical reactions and mechanisms.

### Acknowledgements

The author would like to express his gratitude to the WebMO development team, especially to William F. Polik and Jordan R. Schmidt, for their help and assistance to install and configure the WebMO Pro software; as well as the University of Pittsburgh at Titusville for funding and donation of computers to this project.

References

1. Wade, L.G. “*Organic Chemistry*”, 7<sup>th</sup> Ed. Prentice Hall, 2010
2. Hehre, W.J., Shusterman, A.J., Huang, W.W. “*A Laboratory Book of Computational Chemistry*”, 1<sup>st</sup> Ed. Wave Function Inc. 1998
3. Wade, L.G., Hehre, W.J., Shusterman, A.J., and Nelson, J.E. “*Molecular Modeling Workbook*”, 6th Ed. Prentice Hall, 2006
4. Leach, A.R. “*Molecular Modeling Principles and Applications*”, 2<sup>nd</sup> Ed. Prentice Hall, 2001.
5. Gaussian 09, Revision A.1, M. J. Frisch, G. W. Trucks, H. B. Schlegel, G. E. Scuseria, M. A. Robb, J. R. Cheeseman, G. Scalmani, V. Barone, B. Mennucci, G. A. Petersson, H. Nakatsuji, M. Caricato, X. Li, H. P. Hratchian, A. F. Izmaylov, J. Bloino, G. Zheng, J. L. Sonnenberg, M. Hada, M. Ehara, K. Toyota, R. Fukuda, J. Hasegawa, M. Ishida, T. Nakajima, Y. Honda, O. Kitao, H. Nakai, T. Vreven, J. A. Montgomery, Jr., J. E. Peralta, F. Ogliaro, M. Bearpark, J. J. Heyd, E. Brothers, K. N. Kudin, V. N. Staroverov, R. Kobayashi, J. Normand, K. Raghavachari, A. Rendell, J. C. Burant, S. S. Iyengar, J. Tomasi, M. Cossi, N. Rega, J. M. Millam, M. Klene, J. E. Knox, J. B. Cross, V. Bakken, C. Adamo, J. Jaramillo, R. Gomperts, R. E. Stratmann, O. Yazyev, A. J. Austin, R. Cammi, C. Pomelli, J. W. Ochterski, R. L. Martin, K. Morokuma, V. G. Zakrzewski, G. A. Voth, P. Salvador, J. J. Dannenberg, S. Dapprich, A. D. Daniels, Ö. Farkas, J. B. Foresman, J. V. Ortiz, J. Cioslowski, and D. J. Fox, Gaussian, Inc., Wallingford CT, 2009
6. David Young, “*Computational Chemistry*”, Wiley-Interscience, 2001. Appendix A. A.1.6 pg 330, SPARTAN
7. Limi, Alexander, Runyan, Alan, and Anderson, Vidar. “Scientific Linux – Welcome to Scientific Linux (SL).” *Scientific Linux*. 2010. Web. <https://www.scientificlinux.org>
8. Junnonen, Tomas. “Firestarter.” *Firestarter*. 2010. Web. <http://www.fs-security.com>
9. GAMESS Version 21 Nov. 1995. Schmidt, M.W., Baldrige, K.K., Boatz, J.A., Elbert, S.T., Gordon, M.S., Jensen, J.J., Koseki, S., Matsunaga, N., Nguyen, K.A., Su, S., Windus, T.L., Dupuis, M., and Montgomery, J.A. *J.Comput.Chem.* 14, 1347-1363 (1993)
10. Schmidt, J.R.; Polik, W. *WebMO Pro*, version 8.0; WebMO LLC: Holland, MI, USA, 2010; available from <http://www.WebMO.net>
11. Polik, W.F., Schmidt, J.R. *WebMO User’s Guide*, WebMO LLC: Holand, MI, USA, 2010 available from <http://www.WebMO.net>