J Comput Aided Mol Des (2008) 22:119–129
DOI 10.1007/s10822-008-9183-x

GridMol: a grid application for molecular modeling and visualization

Yanhua Sun · Bin Shen · Zhonghua Lu · Zhong Jin · Xuebin Chi

Received: 15 June 2007 / Accepted: 16 January 2008 / Published online: 30 January 2008
© Springer Science+Business Media B.V. 2008

Abstract In this paper we present GridMol, an extensible tool for building a high performance computational chemistry platform in the grid environment. GridMol provides computational chemists one-stop service for molecular modeling, scientific computing and molecular information visualization. GridMol is not only a visualization and modeling tool but also simplifies control of remote Grid software that can access high performance computing resources. GridMol has been successfully integrated into China National Grid, the most powerful Chinese Grid Computing platform. In Section "Grid computing" of this paper, a computing example is given to show the availability and efficiency of GridMol. GridMol is coded using Java and Java3D for portability and cross-platform compatibility (Windows, Linux, MacOS X and UNIX). GridMol can run not only as a stand-alone application, but also as an applet through web browsers. In this paper, we will present the techniques for molecular visualization, molecular modeling and grid computing. GridMol is available free of charge under the GNU Public License (GPL) from our website: http://www.sccas.cn/~syh/GridMol/index.html *Contact:* syh@sccas.cn

Keywords Grid computing · GridMol · Java 3D^{TM} · Molecular visualization · Molecular modeling · One-stop service · Protein secondary structure

Y. Sun (✉) · B. Shen · Z. Lu · Z. Jin · X. Chi (✉)
Super Computing Center, Computer Network Information Center, Chinese Academy of Sciences, Beijing 100080, China
e-mail: syh@sccas.cn

X. Chi
e-mail: chi@sccas.cn

Introduction

With the development of computers, more and more chemists are using computers for molecular visualization, modeling and computing. Several molecular visualization programs have been developed for personal computers (e.g. RasMol [1], VMD [2], MolMol [3]) and several distributed computing programs (e.g. Gaussian [4], GAMESS [5], GROMACS [6]) for remote high performance computers.

The traditional way to use the aforementioned programs is first to model and visualize molecular structures on local personal computers, then use Secure Shell (SSH), a program and a protocol used for securely transferring information between computers, to access remote high performance computers, and at last download the computing results to local computers for analysis.

Interaction with remote high performance computers has been an important research area. Among many solutions, Grid computing provides a good way to integrate and share distributed resources. All over the world, there have been projects to build computational chemistry platforms through grid computing techniques. CSE-online [7], supported by the National Science Foundation, provides an extensible, integrated, web-accessible simulation environment for computational science and engineering. Computational Chemistry Grid (GridChem) [8] is aiming to build an integrated cyber infrastructure for computational chemistry. There are also some other projects to do similar things such as Reality Grid [9], Grid Enabled Molecular Science Through Online Networked Environments (GEMSTONE) [10] and others.

All of these environments provide many tools for molecular visualization and modeling as well as access to supported grid resources for computing. While GridMol is primarily developed as a visualization and modeling tool

with the same functionalities as other stand-alone visualization tools, it also uniquely provides a graphical interface to submit computing jobs through grid computing middleware.

We will first describe the key techniques for molecular visualization and modeling. Subsequently, we will discuss in detail grid computing job submission in GridMol and provide an example of Gaussian optimization computing as well.

The remainder of this paper is organized as follows: system overview; techniques for molecular visualization and modeling; the performance of GridMol by comparison with JMV [11]; how GridMol communicates with remote high performance computers to do chemical computing; summary and future research directions.

System overview

GridMol is developed for easier use of high performance computational resources. Its main functions include molecular visualization, molecular modeling and grid computing job submission.

The program is distributed through a Grid Portal which requires login. When the user has successfully logged into the Grid Portal, GridMol is available for modeling new molecules, opening existing files, modifying existing molecules, and many other tasks. In Fig. 1, a computational job is submitted to high performance computers through grid middleware, which is responsible for job scheduling. After job sumission, users can query the status of the job that is running on remote computers. After the job is finished, GridMol can be used to analyze the results such as molecular visualization. In this process, users directly interact only with GridMol, the interface for grid computing. Users do not need to know how to submit their jobs and which high performance computer to take the task of computing.

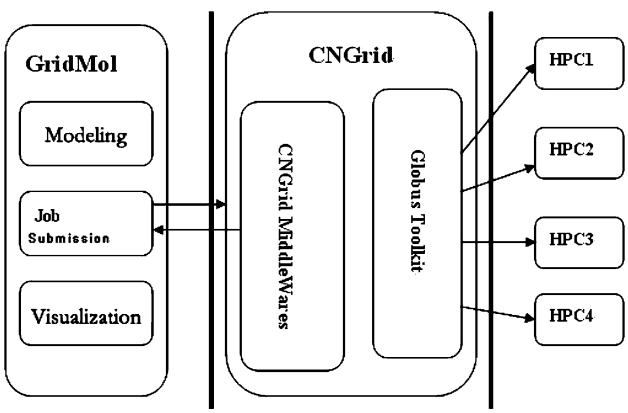

Fig. 1 System overview

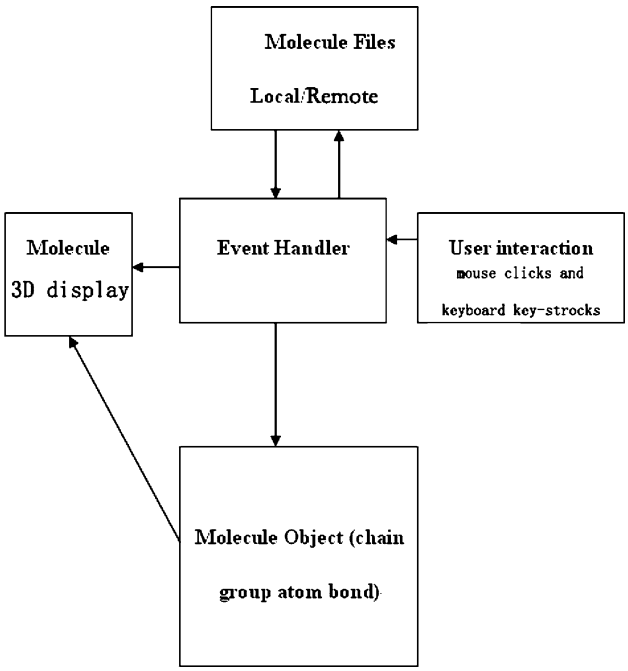

Fig. 2 GridMol design

Figure 1 diagrams the main components of GridMol and how GridMol interacts with users and communicates with remote high performance computing resources.

GridMol is an event-driven program; therefore application of the Model-View-Control design model is suitable. Figure 2 shows the design model of GridMol.

The term "model" refers to the structure of specific physical molecules. The structure of a molecule is composed of one or more chains. Each chain has one or more groups composed of atoms, which are the most basic element of a molecule. Therefore, our molecular model is defined as follows:

$$\text{Molecule} = \{\text{chains}\};\ \text{Chain} = \{\text{Groups}, \text{bonds}\};\ \text{Group} = \{\text{Atoms}\}$$

The term "view" stands for graphical user interface including two-dimensional and three-dimensional displays. Two-dimensional views provide interfaces to control the three-dimensional display of the molecule.

The term "control" encompasses any kind of event, including mouse clicks and keystrokes, which all must be monitored and interpreted to respond to user commands.

Molecular visualization

Molecular visualization has become an important research topic, especially in light of the accomplishment of the Human Genome Project (Burley et al. 1999). In this section, we will briefly discuss methods for molecular

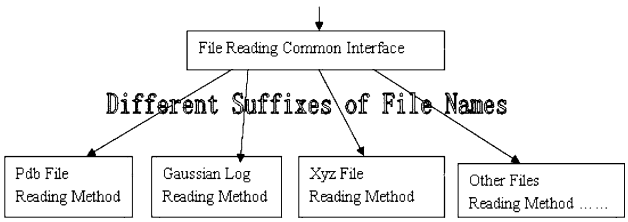

Fig. 3 GridMol scalable file reading design

visualization. While small molecules are easy to visualize, large molecules of thousands of atoms or more are more difficult. Thus, we mainly focus on how to visualize large molecules. At the end of Section "Protein visualization", we will give results obtained via GridMol.

GridMol reads files of many different formats (see Fig. 3) through factory design patterns, enabling easy addition of novel formats. GridMol can currently read files of types of pdb, Gaussian Com, Gaussian Log, Gaussian Gjf and xyz format. Additional file formats can be read by adding a new reading pattern.

Three-dimensional representation of molecules

There are many kinds of representations for the same molecule. Different representations highlight different aspects of the molecule, each having unique advantages and disadvantages. GridMol has different display models, listed below (Note: all the pictures are generated by GridMol).

Line Model (Fig. 4). Bonds are shown as lines while atoms are not displayed. This model clearly represents those connections between all atoms.

Ball-and-stick model (See Fig. 5). All atoms are shown as spheres of different size and all bonds are as cylinders of different length. This model is clear for understanding molecule structure. However, it leads to slow rendering and thus user interaction due to larger memory load than other models, such as the line and tube models.

Space-filling model (Fig. 6). All atoms are shown as spheres of different size and different color according to their van der Waals radius and atom types while bonds are

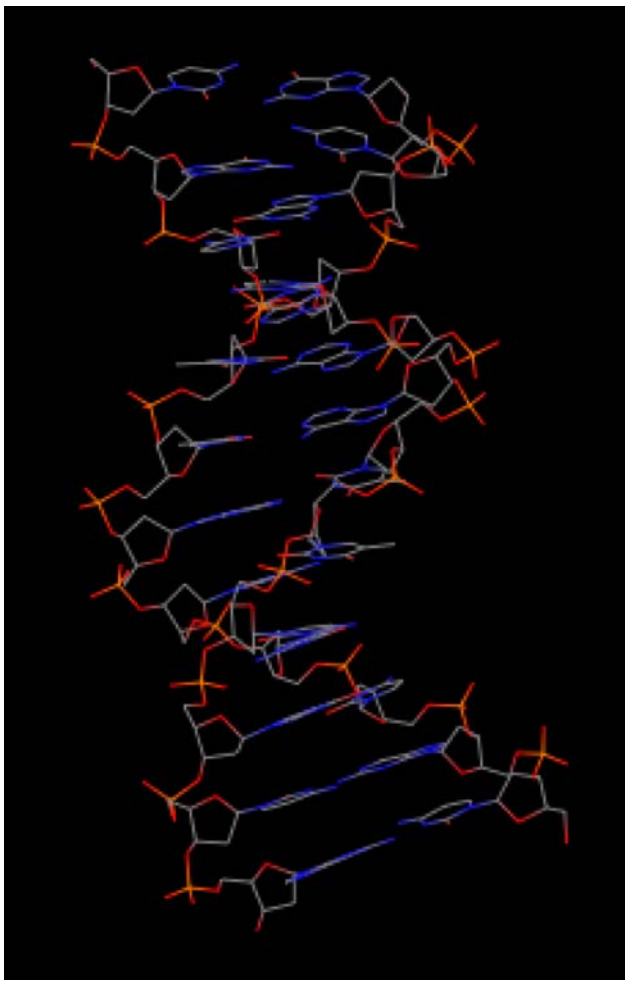

Fig. 4 Line model (PDB ID: DICK)

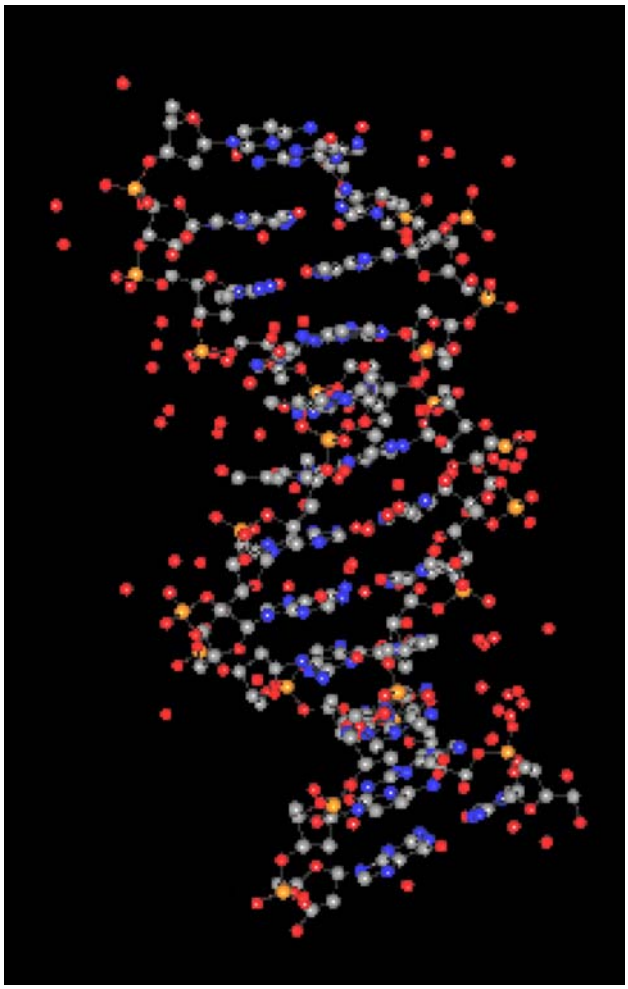

Fig. 5 Ball-and-stick model (PDB ID: DICK)

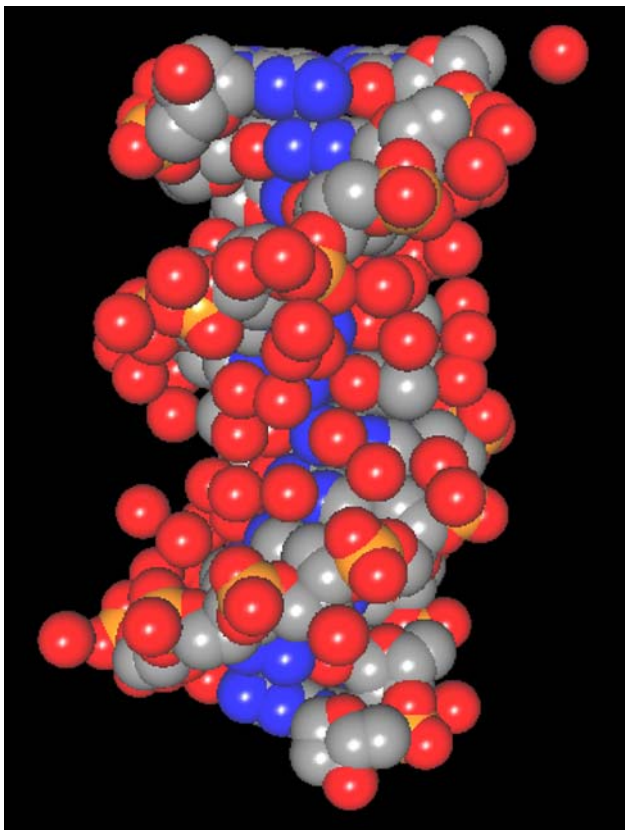

Fig. 6 Space-filling model (PDB ID: DICK)

not displayed. This model provides a good way to understand the volume of molecule.

Tube model. For large molecules such as protein, DNA, or RNA, only the backbone is displayed smoothly with cylinders. Figure 7 shows the tube representation of protein (PDB ID: 2HHB).

Ribbon model. In this model the backbone atoms (the Cα atoms of the protein's backbone or the phosphor atoms of nucleic acids) and additional information (the oxygen atoms of the protein's backbone or the phosphate oxygen atoms of nucleic acids) are used to draw directed ribbons to represent large molecules. See Fig. 8 (PDB ID: 2HHB).

Cartoon model. This model is used to display protein's secondary structure including helix, sheet and coil. We will explain what protein's secondary structure is and how to determine it from atom coordinates in section see Figs. 9 and 10 (PDB ID: 1DIH).

Protein secondary structure determination and visualization

Determining a protein's secondary structure is required for cartoon model visualization. There are many algorithms to determine the secondary structure of proteins from atomic coordinates. Among the algorithms, the DSSP [12] method

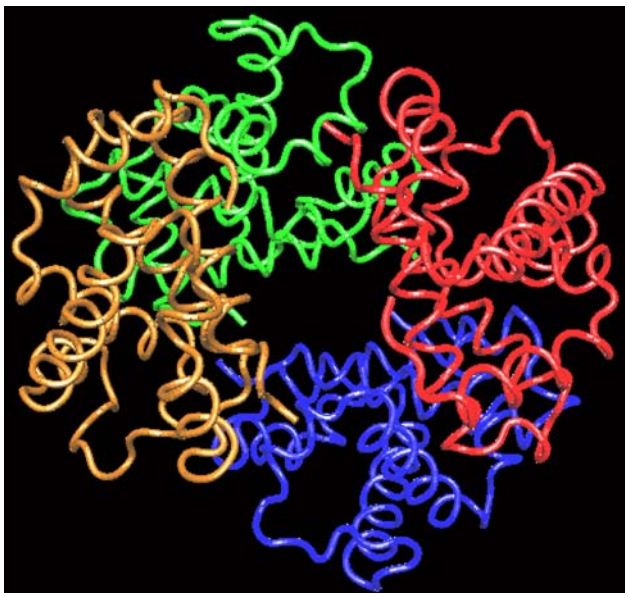

Fig. 7 Tube model (PDB ID: 2HHB)

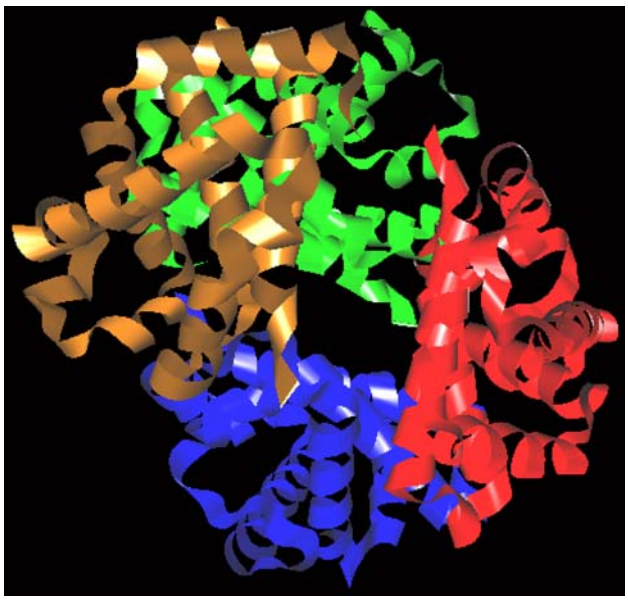

Fig. 8 Ribbon model (PDB ID: 2HHB)

is efficient and relatively strict. We adopt the main idea of DSSP to determine a protein's secondary structure and implement this method for protein visualization.

We will define several terms before presenting our algorithm.

Hydrogen bond

In chemistry, a Hydrogen bond (i, j) between oxygen atom (O) in the residue i and hydrogen atom (H) in residue j is formed when the electrostatic interaction energy (E) is less

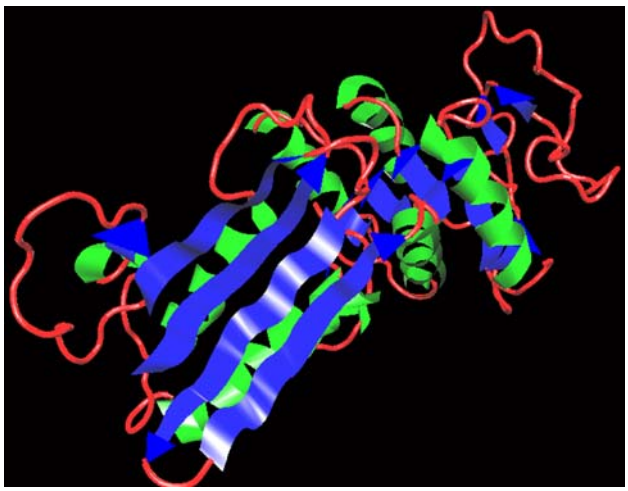

Fig. 9 Cartoon model (PDB ID: 1DIH)

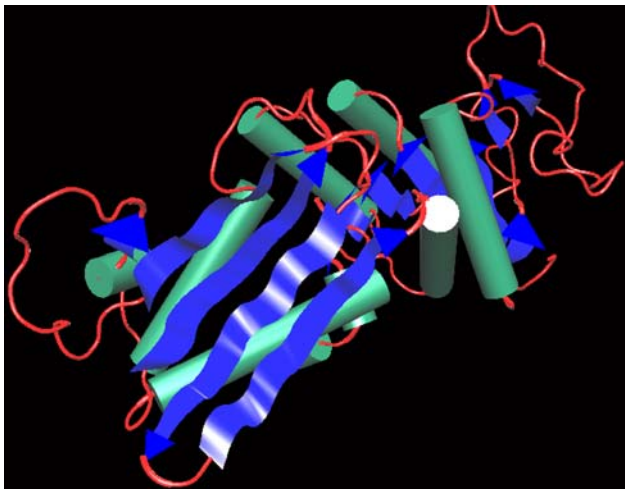

Fig. 10 New cartoon model (PDB ID: 1DIH)

than a cutoff of −0.5 kcal/mol [12]. The electrostatic energy is calculated using the equation:

$$E = q_1 q_2 (1/r(ON) + 1/r(CH) - 1/r(OH) - 1/r(CN) * f \quad (1)$$

With $q_1 = 0.42e$ and $q_1 = 0.20e$, r(AB) the interatomic distance from A to B, f = 332.

More details can be found in DSSP [12].

N-Turn

An N-Turn is a hydrogen bond that exists between residue i and proceeding residue, $i + n$. Research [12] shows that only 3-Turn, 4-Turn and 5-Turn type bonds are possible.

Bridge

Two kinds of bridges are defined as follows:

$$\begin{aligned}&\text{Parallel Bridge}(i,\ j) =: \\ &\quad [\text{Hbond}(i-1,\ j) \text{ and } \text{Hbond}(j,\ i+1)] \\ &\quad \text{or } [\text{Hbond}(j-1,\ i) \text{ and } \text{Hbond}(i,\ j+1)]\end{aligned} \quad (2)$$

$$\begin{aligned}&\text{Antiparallel Bridge}(i,\ j) =: \\ &\quad [\text{Hbond}(i,\ j) \text{ and } \text{Hbond}(j,\ i)] \text{ or } [\text{Hbond}(i-1,\ j+1) \\ &\quad \text{and } \text{Hbond}(j-1,\ i+1)]\end{aligned} \quad (3)$$

Algorithm to determine protein secondary structure

Step 1 Search the molecule for all backbone atoms of Cα, C, O, N and H in each residue. If there is no H bonded with N, we apply a Hydrogen adding algorithm to calculate all Hydrogen atoms.

Step 2 Calculate whether a Hydrogen bond is formed between any two residues and then store all the Hydrogen bonds for future use.

Step 3 For each residue, check whether the H bond type is 5-Turn, 4-Turn, or 3-Turn. Two or more consecutive turns of the same type form one fragment of helix. Search all the residues to find all helices.

Step 4 According to formulas of (2) and (3) above, find all parallel and anti-parallel bridges. Two or more consecutive bridges of the same type form one sheet. Find all sheets.

Results

We compare our assignments with those of the crystallographers for five proteins 1CC8, 105M, 1AF5, 1AOV, 1D02 at respectively resolution of 1.02, 2.02, 3.0, 5.0, and 4.0 Å. Tables 1 and 2 are comparisons of the original secondary structure and that calculated by GridMol's method. For all the segments of the proteins, the difference is less than 10%. Our implementation is relatively strict for either low-resolution proteins or high-resolution proteins.

Protein visualization

A smooth curve is required to display a protein backbone. We use the Catmull-Rom [13] spline curve because the curve passes through all control points. We adopt the Catmull-Rom spline interpolation algorithm to draw the

Table 1 Comparison of secondary structure assignments for three proteins of different resolutions

Structure	Original author	Our method
1CC8 (resolution 1.02 Å)		
E1	5–10	5–10
H1	16–29	16–29
E2	33–39	35–39
H2	44–49	44–49
E3	53–63	53–63
E4	67–73	69–71
105M (resolution 2.02 Å)		
H1	3–18	3–35
H2	20–35	
H3	36–42	36–42
H4	51–57	51–57
H5	58–77	58–78
H6	86–94	82–95
H7	100–118	100–118
H8	124–149	124–149
1AF5 (resolution 3.0 Å)		
H1	10–19	10–19
E1	22–28	22–28
E2	38–44	38–44
H2	49–51	49–51
H3	53–62	53–62
E3	66–70	66–70
E4	73–77	74–77
H4	81–94	81–94
H5	101–108	101–108
H6	128–131	128–131

H, helix; E, sheet. Columns 2 and 3 mean the amino acid sequence number

curve according to the control points, which stand for C_α atoms in each residue. Figure 11 shows that the spline curve between P_1, P_2 is determined by control points of P_0, P_1, P_2 *and* P_3.

For any point P on the curve, we derive the coordinate of P by the Eq. 4:

$$P(t) = 0.5 \times [t^3, t^2, t, ^1] \times \text{Matrix}[4][4] \times [P_0, P_1, P_2, P_3] \quad (4)$$

Table 2 Comparison of secondary structure assignments for two proteins of different resolutions

PDB ID	Number of helix		Number of sheet	
	Original	Our method	Original	Our method
1AOV (resolution 5.0 Å)	27	28	25	28
1D02 (resolution 4.0 Å)	85	88	36	42

$$\text{Matrix}[4][4] = \begin{bmatrix} -1.0/S & -1.0/S+2.0 & 1.0/S-2.0 & 1.0/S \\ 2.0/S & 1.0/S & -2.0/S+3.0 & -1.0/S \\ -1.0/S & 0.0 & 1.0/S & 0.0 \\ 0.0 & 1.0 & 0.0 & 0.0 \end{bmatrix} \quad (5a)$$

With S = 1.25, which can be modified to calculate different effects.

GridMol can read molecular files from the local disk or the network and display those files in different models. Figures 12 and 13 show the protein models produced by our program, GridMol, and JMV, which is another classic Java/Java3D Molecular viewer. Our program produces similar display results. GridMol has better performance than JMV (see Section "Implementation and performance").

Molecular modeling

Molecular modeling is the study of the geometry and properties of molecules by computer-aided techniques. It has become a growing area in chemistry due chemists' need to visualize molecules, study their structure, and compare structures and properties of different molecules. In this section, we first present the main functions of molecular modeling and then explain the algorithm used to modify dihedral angles.

Modeling functions

GridMol has all the basic functions for molecular modeling, including modifying bond length and angle, changing dihedral angle, adding or deleting atoms, and adding radicals. In this section, we mainly discuss the techniques for adding radicals easily. Algorithms employed in GridMol for the described functions will be discussed in Section "Algorithm to modify dihedral angles".

GridMol contains many kinds of radicals: single atoms, simple organic radicals, and amino acid radicals. Molecular

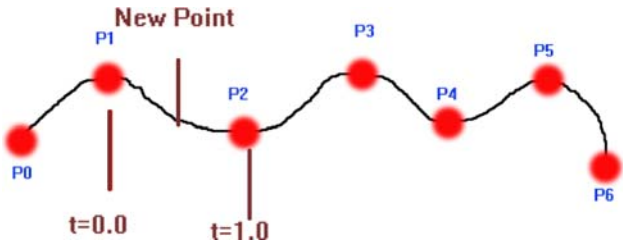

Fig. 11 Catmull-Rom spline

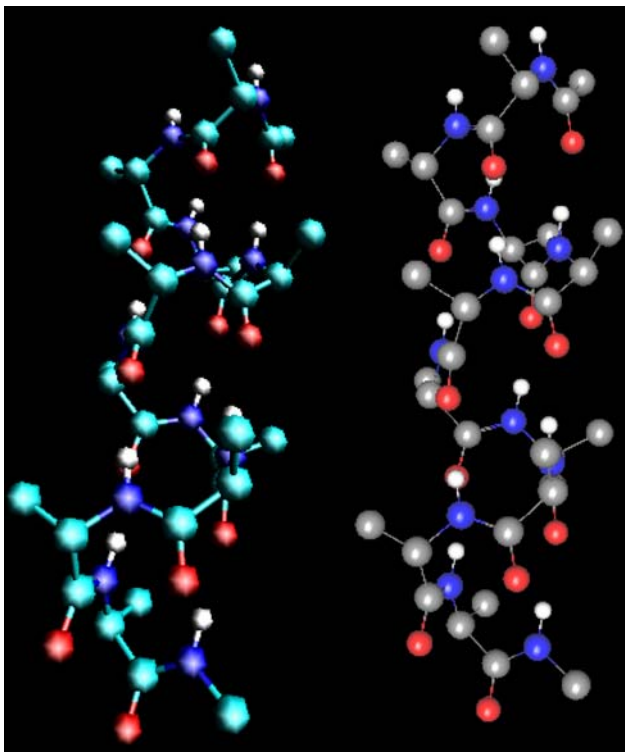

Fig. 12 Ball-stick model for 1CRN, JMV (*left*) and GridMol (*right*)

mechanics force field (TAFF) is used in GridMol to derive parameters of an more accurate molecular model.

We design radical information as configurable files, enabling easy addition of more radical types to GridMol.

Configurable files are organized as follows:

$$\text{Radical Index File} = \{\{\text{radical type 1, radical file names}\},\ \{\text{radical type 2, radical file names}\},\ldots\}$$

$$\text{Radical files} = \{\{\text{radical file1}\},\{\text{radical file2}\},\ \{\text{radical file3}\}\ldots\}$$

Therefore, adding a new type of radical only requires adding one record in Radical Index File and adding the corresponding radical files in one directory.

Algorithm to modify dihedral angles

Algorithms for the functions of modifying bond length, bond angle and dihedral angle are very similar. In this section, only the algorithm to modify dihedral is discussed in detail.

A dihedral angle is defined as the angle between two planes that are determined by three connected atoms (Fig. 14). When we modify the dihedral angle determined by four atoms of A, B, C, D, the positions of all atoms rooted from atom C are recalculated and their three-dimensional displays are updated.

Modifying the dihedral angle determined by atoms of A, B, C and D requires rotating all the atoms rooted from C along the axis of BC. We define rotation matrix Ra along an arbitrary axis BC by angle t below.

$$\text{Ra} =: \begin{bmatrix} mc(t)*a^2+c(t) & mc(t)*a*b-c*s(t) & mc(t)*a*c+b*s(t) & 0 \\ mc(t)*a*b+c*s(t) & mc(t)*b^2+c(t) & mc(t)*b*c-a*s(t) & 0 \\ mc(t)*a*c-b.s(t) & mc(t)*b*c+a*s(t) & mc(t)*c^2+c(t) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{5b}$$

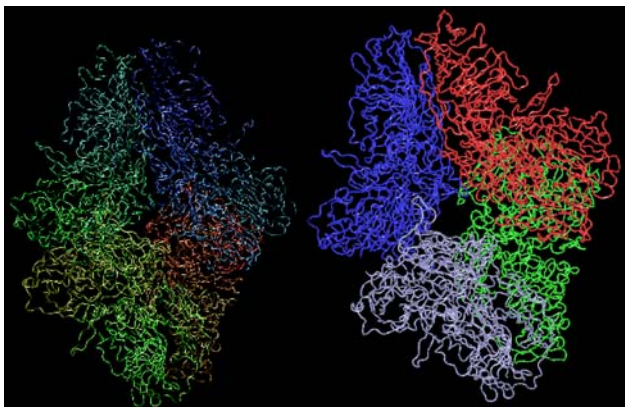

Fig. 13 Tube model for 1DP0, JMV (*left*) and GridMol (*right*)

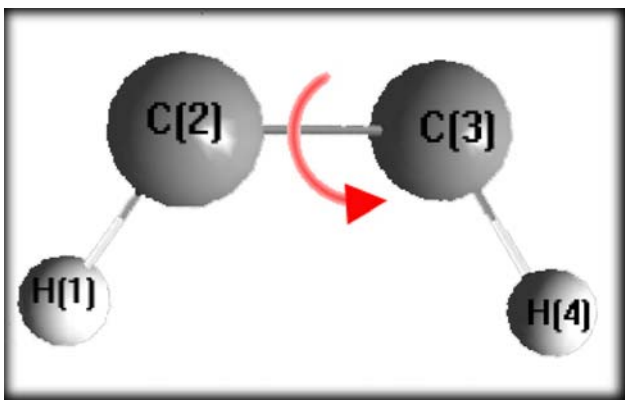

Fig. 14 Modify dihedral

When H is rotated along AB, H′ is defined as:

$$H' = Ra(H) \quad (6)$$

With $BC = [a,\ b,\ c]^T$, $c(t) = \cos(t)$; and $a^2 + b^2 + c^2 = 1$; $mc(t) = 1.0 - \cos(t)$; $s(t) = \sin(t)$

The algorithm used to modify dihedral angles determined by atoms of A, B, C and D is four-step:

Step 1 Use Depth-First Search algorithm to search all the atoms that are rooted from atom C. Store all these atoms into the set of S.

Step 2 Calculate matrix Ra that is the transform matrix of rotation along BC by the angle of t, the difference of the old dihedral angle and the new dihedral angle. See Fig. 14.

Step 3 The new address of any atom R in set S is recalculated by multiplying the old address and the transform matrix Ra.

Step 4 Three-dimensional views are updated according to new addresses of atoms.

Figure 15 depicts a manually built peptide consisting of 5 amino acids: Ala, Val, Cyt, Gly, Ser. More changes such as modifying any bond length, bond angle or dihedral angle of the new molecule can be made to satisfy our requirement.

Implementation and performance

GridMol is coded using Java and Java3D. Most visualization and modeling tools are based on C/C++ and OpenGL. Only a few, such as JMVS [14], JMV, WebMol [15] and FPV [16], are coded using Java and Java3D. Programs using Java and Java3D are more seamless for both users and developers. However, performance is often too slow.

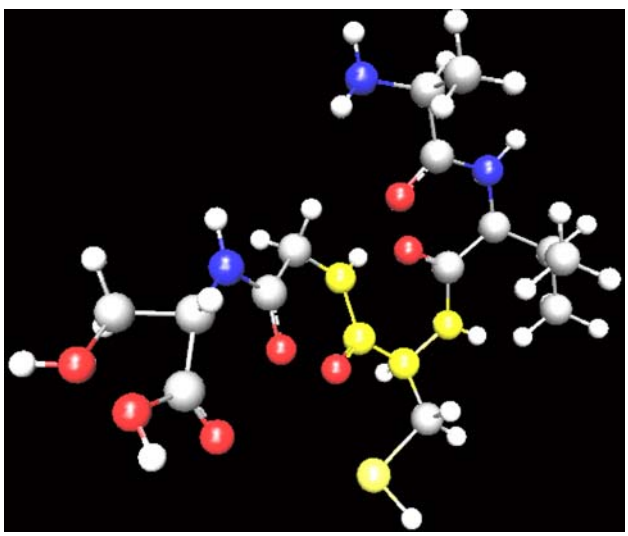

Fig. 15 Model a molecule

We have compared our program (GridMol) to another molecular visualization tool (JMV) according to performances of memory used and real time interaction speed. Compared to JMV, our program has many more features. Two important features are molecular modeling and grid computing job submission (Section "Grid computing"). JMV has similar visualization functionality with GridMol. Thus, we compared the performance for molecular visualization of GridMol and JMV.

Our tests were performed using JAVA JRE 1.6.1 and JAVA 3D 1.5.1 (DirectX version) on a Microsoft Windows XP machine with Intel® Core™ 2 CPU 6300 at 1.86 GHZ and 2 GB of RAM. We have dedicated 512 MB of memory as the maximum size of memory allocation pool for Java Virtual Machine. And all the test data are produced by JConsole 1.6.0-b105. The graphics accelerator card used for the tests was 256 MB ATI Radeon x1300 p. The data set was from RCSB [17] protein data bank. The proteins are ranging in size from 1BH0 (29 amino acids, 242 atoms) to huge molecule 1F8V [18] which has 60 models each of which has 7 chains, 25 RNA bases, 1225 amino acids, 8726 atoms. Thus, 1F8v has a total of 75,000 groups and 523,560 atoms.

GridMol can use tube model, ribbon model and cartoon model to display backbone of biomolecules while JMV only has tube display model. Fig. 16 shows results of the stack memory cost for the protein set for ribbon model, tube model, and cartoon model in GridMol and tube model in JMV. According to the data collected, the ribbon display model uses one-sixth of the memory used by the tube model in GridMol and one-twelfth of memory used by the tube model in JMV. For the same tube display model, GridMol uses half as much stack memory as JMV. GridMol can visualize Molecule 1F8V with 75,000 groups (see Fig. 18) using about 380 MB stack memory for ribbon model while JMV can only visualize biggest molecule 1GYT with 6,036 amino acids using 518 MB stack memory for tube model. The ribbon model uses less memory because a "triangle strip array" is used for 3-dimensional

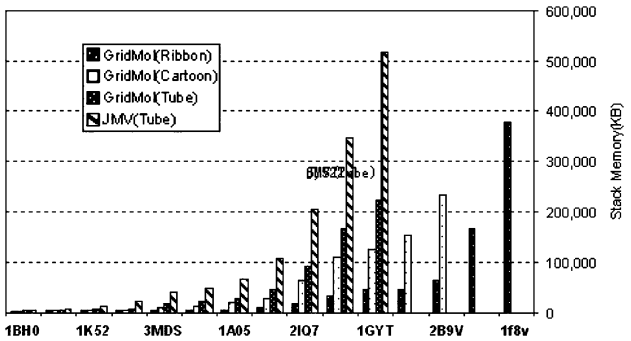

Fig. 16 Stack memory cost for the ribbon, cartoon, tube model of GridMol and tube model of JMV

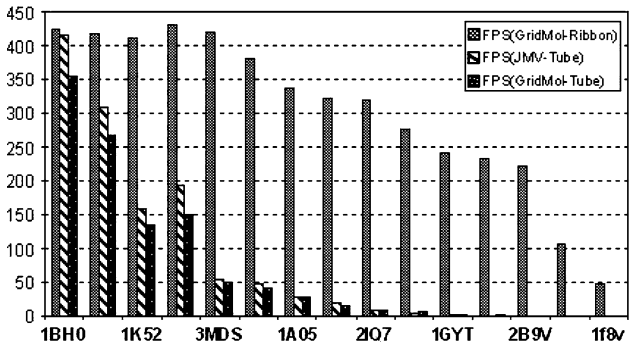

Fig. 17 Rendering speed for ribbon and tube model in GridMol and tube model in JMV

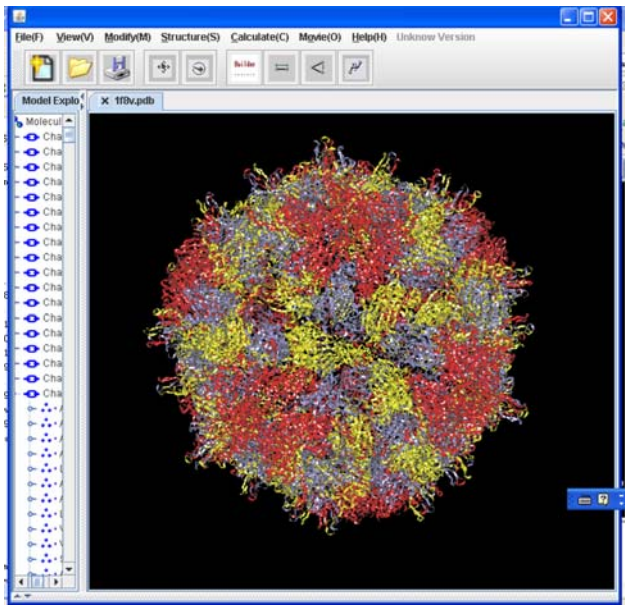

Fig. 18 the whole user interface of GridMol with 3D display of 1F8V

rending, which is less memory intensive than rendering other shapes. The reason why GridMol uses less memory than JMV for molecules displayed in tube model is that in GridMol basic cylinders which constitute the whole tube are uniform.

Figures 17 and 18 graphs the results of the average animation rendering speed for displaying entire proteins in ribbon and tube model in GridMol and tube model in JMV. GridMol and JMV have similar rendering speed for the tube model. GridMol has better performance for the ribbon model. Even for huge molecule 1F8V, GridMol has a rendering speed of 40 frames per second.

Grid computing

Since GridMol is Grid enabled, chemists can easily model new molecules using basic radicals and subsequently choose a computing method to submit a job to Grid Middleware, responsible for scheduling jobs to remote high performance computers. After computing is finished, results can be easily downloaded and visualized within GridMol.

We have integrated GridMol into CNGrid [19] (China National Grid) which was developed for connecting and using high performance computers distributed in eight different supercomputing centers in China. GridMol can submit different jobs to different supercomputing nodes through CNGrid.

Job submission

To submit a grid computing job, GridMol uses standard Job Submission Description Language (JSDL), which provides information about Job Name, Application Name, Executable commands, Stage in File Names, Stage out File Name, and more.

GridMol maintains a history of job descriptions to remember the jobs for future operation.

When users want to submit a computing job, a job submission request is sent to CNGrid middleware, which decides how to submit the job to high-performance computers and communicates directly with high performance computers.

GridMol retrieves the response message from the CNGrid middleware to analyze whether the job has been submitted successfully or has failed.

When job status is requested, a job status query message is sent to CNGrid middleware and then the response message is analyzed to decide what the job status is. When job is finished, the output files are downloaded for future processing.

Computing example

Here we use a Gaussian computing example to show the process of modeling a molecule, submitting a Gaussian computing job to high-performance computers, and visualizing the computing results. See Fig. 19.

(1) Login to CNGrid with username and password. After logging in successfully, a session will be created and saved to track the user's behavior. As long as the session exists, user can use grid all the time.
(2) Model CH_4 molecule. However, the configuration of the molecule is not correct.
(3) Fill the proper job information and then submit a Gaussian optimization job to CNGrid.
(4) Send job status querying message to grid and view the job status.

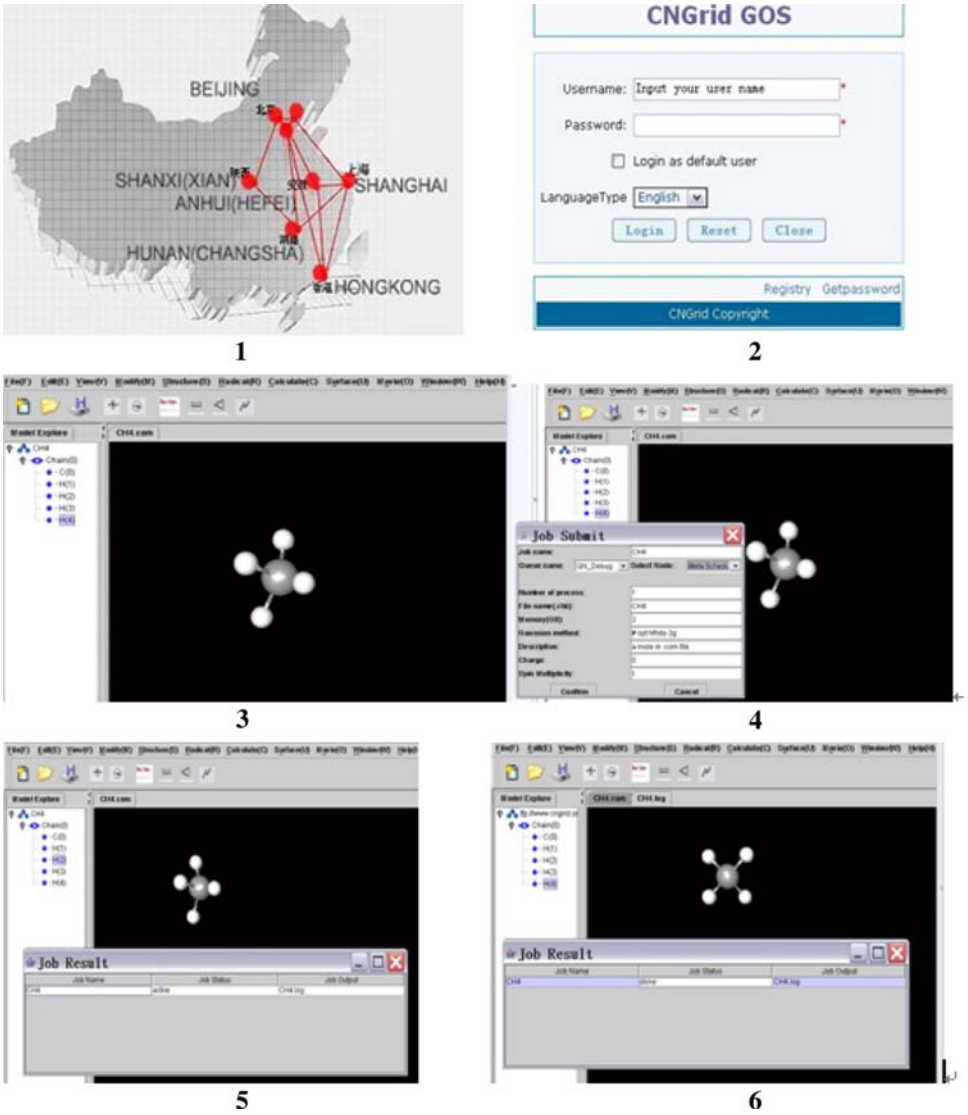

Fig. 19 Gaussian job submission

(5) After the job is finished successfully, visualize the output. We can see that the molecule of CH_4 is now correct.

The main functionalities of grid computing in GridMol have been shown by the above example. Becauseo of the modular design of GridMol, interfaces for other computing programs such as GAMESS computing, Molpro computing and others will be easily implemented.

Discussion

In this paper, we have presented GridMol—a Grid application for molecular visualization and modeling. We have presented techniques for molecular visualization, molecular modeling and Grid computing. Java and Java3D API are used to enable control of GridMol through web browsers.

In order to make our software more attractive to researchers, we are looking for ways to increase the functionality of our system. An important functionality is to add more interfaces to transparently submit more kinds of computing jobs to remote high performance computers. Another important functionality is to analyze more computing results and display more visualization results such as an electronic density map display. Since GridMol is designed in modules, it will be easy to add new functionalities depending on our needs.

Acknowledgements We wish to thank Ryan Ferrell for his comments on this paper. This work is funded by The National High Technology Research and Development Program of China (863 Program) from The Ministry of Science and Technology, PRC, under grant 2006AA01A119, National Natural Science Foundation of China under grant 60673064.

References

1. Sayle RA, Milner-White EJ (1995) RASMOL: biomolecular graphics for all. Trends Biochem Sci 20:374–376
2. Humphrey W, Dalke A, Schulten K (1996) VMD—visual molecular dynamics. J Mol Graphics 14:33–38
3. Koradi R, Billeter M, Wüthrich K (1996) MOLMOL: a program for display and analysis of macromolecylar structure. J Mol Graph 14:51–55
4. Gaussian. http://www.gaussian.com/ (accessed Oct 2007)
5. GAMESS. http://www.msg.ameslab.gov/gamess/ (accessed Oct 2007)
6. GROMACS. http://www.gromacs.org/ (accessed Oct 2007)
7. CSE-online. http://cse-online.net/ (accessed Oct 2007)
8. GridChem http://Grid-devel.sdsc.edu/gemstone (accessed Oct 2007)
9. Reality Grid. http://www.realitygrid.org/ (accessed Oct 2007)
10. GEMSTONE. http://Grid-devel.sdsc.edu/gemstone (accessed Oct 2007)
11. JMV—Java Molecular Viewer http://www.ks.uiuc.edu/Research/jmv/ (accessed Oct 2007)
12. Kabsh W, Sander C (1983) Dictionary of protein secondary structure: pattern recognition of hydrogen-bonded and geometrical features. Biopolymers 22:2577–2637
13. Catmull-room Spline. http://www.mvps.org/directx/articles/catmull/ (accessed Oct 2007)
14. JMVS. http://www.adcworks.com/projects/jmvs/ (accessed Oct 2007)
15. Walther D (1997) WebMol—a java based PDB viewer. Trends Biochem Sci 22:274–275. http://www.embl-heidelberg.de/cgi/viewer.pl
16. Can T, Wang Y, Wang Y-F, Su J (2003) FPV: fast protein visualization using Java 3D™. Bioinformatics 19:913–933
17. RCSB protein data bank. http://www.rcsb.org/ (accessed Oct 2007)
18. 1F8V. http://ndbserver.rutgers.edu/ftp/NDB/coordinates/na-biol-mirror/1f8v.pdb1 (accessed Oct 2007)
19. China National Grid. http://www.cngrid.org/web/english/home/ (accessed Oct 2007)