第35卷 第20期 计 算 机 工 程 2009年10月
Vol.35 No.20 Computer Engineering October 2009
·开发研究与设计技术· 文章编号：1000—3428(2009)20—0242—04 文献标识码：A 中图分类号：TP311

GridMol系统中蛋白质可视化与建模的性能优化

欧阳刘彬，孙衍华，刘继凤，金 钟，陆忠华，迟学斌
(中国科学院计算机网络信息中心超级计算中心，北京 100190)

摘 要：基于网格计算思想开发一个具有计算化学前、后处理功能的系统GridMol，其主要功能包括分子可视化、分子建模和计算作业提交。针对GridMol系统中蛋白质大分子显示和建模遇到的性能问题，给出调整Java 3D场景图进行性能优化的方法，通过GridMol和其他分子可视化软件的性能比较以及自身优化前后的性能比较，证明优化方法取得了良好的效果。

关键词：分子可视化；分子建模；性能分析；GridMol系统

Performance Optimization for Protein Visualization and Modeling in GridMol System

OUYANG Liu-bin, SUN Yan-hua, LIU Ji-feng, JIN Zhong, LU Zhong-hua, CHI Xue-bin
(Super Computing Center of Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190)

【Abstract】 GridMol is a computational chemistry system for pre-processing and post-processing, which is based on the idea of grid computing. Its main functions include molecular visualization, molecular modeling and job submission. This paper analyzes the performance issues in GridMol system and presents techniques to optimize the performance by re-organizing Java 3D scene graphs. It demonstrates the effectiveness of the techniques by comparing GridMol with another molecular visualization software.

【Key words】 molecular visualization; molecular modeling; performance analysis; GridMol system

1 概述

随着生物信息学的不断发展，分子可视化和分子建模对于生物信息分析具有越来越重要的作用。目前已有很多分子可视化软件，如RasMol[1], MOLMOL[2]，它们使用C语言和OpenGL开发，具有很好的性能，但是只限于单机版运行。其他一些基于Java和Java 3D API的分子可视化软件如WebMol[3], JMV[4]可以运行在网络环境中，但是性能和功能上还有欠缺。GridMol系统[5]基于Java和Java 3D API开发，提供分子可视化和分子建模功能，具有跨平台和网络运行的优点。

Java 3D API是一套开发3D图像程序的编程接口，它不像OpenGL和Direct3D那样与3D硬件设计紧密结合，而让开发者在一个更高的级别上编程，不用考虑底层硬件的具体情况。但在为开发者提供程序设计便利的同时，Java也会给程序带来性能问题：在对蛋白质大分子进行显示和建模时，内存消耗大，程序响应速度慢，这是本文要解决的问题。

本文介绍Java 3D进行蛋白质可视化的原理，分析蛋白质显示建模时存在的性能问题，给出性能优化的方法，通过性能对比测试说明优化的效果。

2 Java 3D蛋白质可视化原理

2.1 Java 3D场景图

Java 3D采用场景图的开发方式。场景图包含场景中的所有对象以及如何渲染它们的全部信息。它是一个树型结构，由内容(content)和视图(view)2个分支组成。视图分支定义视点等视图模型。内容分支是程序中重点考虑的部分，用于描述场景所包含的所有图形对象以及针对这些对象的空间变换、光照、行为等。

图1是一个简单的Java 3D场景图，图中虚线框中的部分是SimpleUniverse类负责的部分。本文重点讨论以TG(TransformGroup)对象为根的场景图的内容分支，这些分支主要由相互关联的Group对象组成。每个Group对象节点包含若干子节点，在处理该节点时将渲染这些子节点。TG对象是特殊的Group对象，负责对其子节点应用3D变换(平移、缩放或旋转)。Shape3D对象是场景图的叶子节点，用于定义场景中的三维图形。

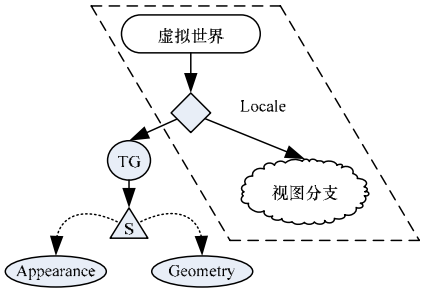

图1 Java 3D场景图

2.2 蛋白质显示的传统实现

GridMol软件对蛋白质的显示提供多种显示模式，本文以线模式、管状模式和大球模式为例分析不同模式的场景图实现及其性能问题。

基金项目：国家“863”计划基金资助项目“高性能计算化学应用系统”(2006AA01A119)；中国科学院知识创新工程青年人才领域基金资助项目(0814051103)

作者简介：欧阳刘彬(1985－)，男，硕士，主研方向：高性能计算及应用；孙衍华，硕士；刘继凤、金 钟，博士；陆忠华，研究员；迟学斌，研究员、博士生导师

收稿日期：2009-03-25 **E-mail：**oylbin@sccas.cn

图 2 是分子的线模式显示效果，整个分子由表示化学键的线段拼接而成。图 3 是线模式对应的场景图，图中内容分支根节点是表示蛋白质分子的 TG 对象，其子节点是表示蛋白质链的 Shape3D 对象，构成链的所有线段的几何信息都存放在该 Shape3D 对象中。

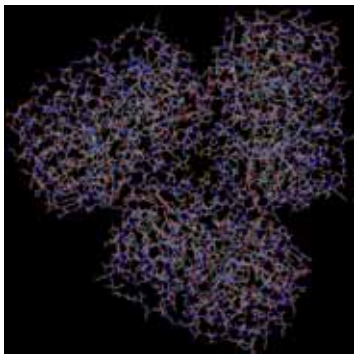

图 2　线模式显示图

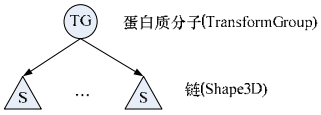

图 3　线模式场景图

如图 4 所示，管状模式下整个分子由代表氨基酸的圆柱体连接而成。图 5 是此模式的场景图，其根节点依旧是一个 TG 对象，该对象的子节点 Group 对象表示每个蛋白质链。Group 对象的子节点代表构成这条链的氨基酸。本文使用 Java3D 提供的 Cylinder 类建立代表氨基酸的圆柱体(图 5 所示场景图的叶节点)。但是 Cylinder 类只能建立以原点为底面中心、以 *Y* 轴为中轴的圆柱体，因此，场景图中每个圆柱需要用一个 TG 对象把它平移和旋转到相应氨基酸的位置。

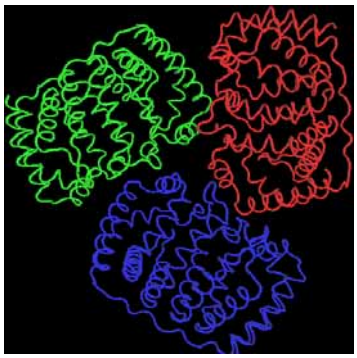

图 4　管状模式显示图

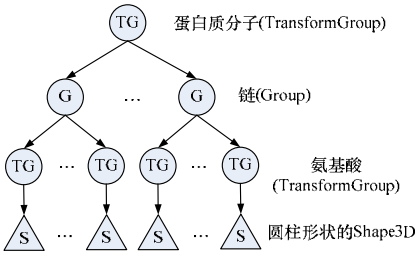

图 5　管状模式场景图

大球模式下原子用球体表示，如图 6 所示。此模式的场景图如图 7 所示，代表分子的 TG 对象为根节点，下层的 Group 节点依次表示链和氨基酸，氨基酸的子节点就是各个原子。由于 Java3D 提供的 Sphere 类只能建立以原点为球心的球体，因此在场景图中，每个原子需要一个 TG 对象把球体平移到相应原子的位置。

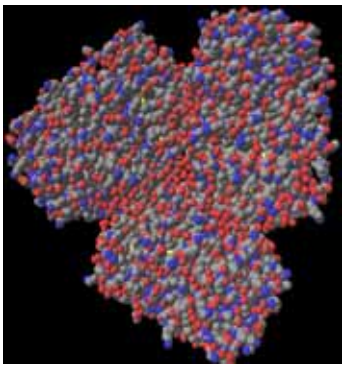

图 6　大球模式显示图

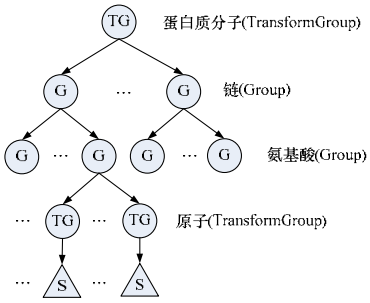

图 7　大球模式场景图

上面给出了直接利用 Java3D 提供的基本类和基本场景图模式构造的 3 种显示模式的显示效果图及场景图。在 Java3D 场景图中，Shape3D 对象和 TG 对象的数量是影响性能的重要因素：过多的 Shape3D 会带来大量的内存消耗，过多的 TG 会增加程序的响应时间。在线模式下，只有 1 个 TG 对象，Shape3D 对象的数目和蛋白质链数目相同。在管状模式下，TG 对象和 Shape3D 对象的数目和分子中氨基酸数目相同。在大球模式下，TG 对象和 Shape3D 对象数目和分子中原子数目相同。蛋白质分子中包含的氨基酸基团的数目通常很大，此时在管状模式和大球模式下，Shape3D 对象和 TG 对象数目很多，导致显示蛋白质大分子时出现性能问题。因此，为了减少内存消耗，提高程序的响应速度，必须减少 Shape3D 对象和 TG 对象的数目。

3　性能优化

3.1　显示模式的优化

显示模式的优化步骤如下：

(1)减少 TG 对象。以大球模式为例，在此模式下要使用大量 TG 对象的原因是 Java3D 提供的 Sphere 类只能在指定的位置建立球体，每个球都需要一个 TG 对象进行位置转换。因此，本文自定义一个可以在任意位置建立球体的 Sphere 类，以此省去这些不必要的 TG 节点。此时的场景图如图 8 所示。

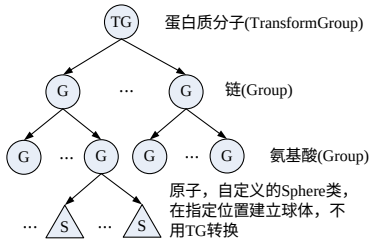

图 8　去掉不必要 TG 后的场景图

(2) 减少 Shape3D 对象。Shape3D 对象中包含若干 Geometry 对象和一个 Appearance 对象。Geometry 对象定义

图形的几何形状，Appearance 定义图形的外观(颜色、材质等)。在分子可视化中，相同类型的原子有相同的外观。所以，把同类型的原子 Shape3D 对象中的 Geometry 对象提取出来，合并到一个 Shape3D 对象中，使它们共用一个 Appearance 对象。另外，场景图中表示氨基酸的 Group 节点只是方便表达分子结构，对于构图没有实际意义，可以省掉。此时的场景图如图 9 所示。

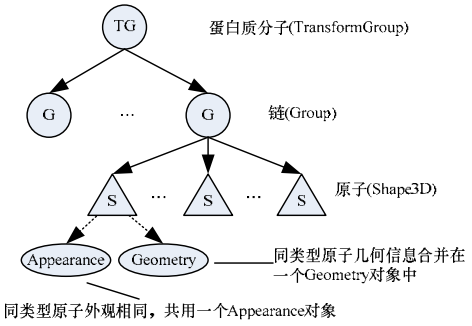

图 9　合并 Shape3D 对象后的场景图

大球模式经过上述优化后，整个分子只使用一个 TG 对象，Shape3D 对象个数等于蛋白质链个数与蛋白质中包含的原子类型个数之积。在一般蛋白质分子中，由于原子类型主要是 C, H, O, N, P, S，其他类型的原子很少，因此这时候 Shape3D 对象很少。

管状模式的优化类似。首先自定义并实现在任意位置建立圆柱的 Cylinder 类，去除不必要的 TG 对象，再把每条蛋白质链上的圆柱体合并在一个 Shape3D 对象中(因为同一条链上的圆柱体有相同的外观)。优化后的管状模式场景图和图 3 所示的线模式场景图相同。

3.2　建模模式的优化

GridMol 的建模操作在球棒模式下进行，场景图和图 7 所示大球模式场景图相同。在对大的蛋白质分子建模时，也会遇到和大球模式一样的性能问题。许多建模操作最终都会分解成对单个原子的位移操作，每个原子对象必须独立出来，不能合并，因此，上一节提到的优化方式不适用于建模模式。为优化建模模式的性能，需要对场景图做另外的调整。

考虑到建模操作发生在蛋白质的某个局部，如蛋白质的修饰[6]是在某个氨基酸的侧链上做修改，此时只需要用球棒模式显示局部的氨基酸，其余大部分使用线模式，以减少不必要的 TG 对象和 Shape3D 对象。改进后的建模模式场景图见图 10。

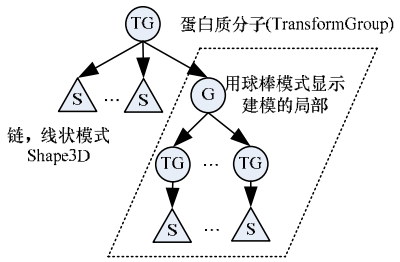

图 10　改进后的建模模式场景图

根节点下方虚线框外的节点是代表蛋白质链的 Shape3D 对象。虚线框的内部是进行建模操作局部的场景图分支，其中每个原子由一个绘制化学键和原子球体的 Shape3D 对象和一个存放原子坐标的 TG 对象共同表示。虚线框外部 Shape3D 对象数目和蛋白质链数目相同，虚线框内部 TG 对象和 Shape3D 对象数目和建模局部的原子个数相同。

图11是对PDB ID[7]为1F2E的蛋白质分子做局部建模操作时的画面。建模的局部是球棒模式，保留了建模操作的直观性。其余部分用线模式表示，图中需要的TG对象和Shape3D对象从优化前的5 948个(1F2E的原子个数)减至17个(图11中以球棒模式显示的原子个数)，大量减少了内存的占用，同时提升了程序的响应速度。

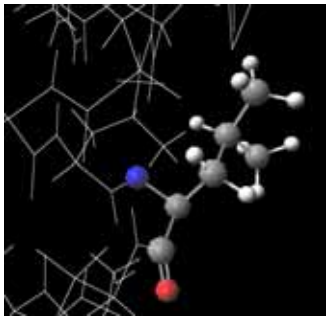

图 11　优化后的修饰建模

4　实验结果与分析

上一节根据减少 TG 对象和 Shape3D 对象的思路调整场景图，优化 GridMol 的性能，本节通过测试说明优化的效果。给出 2 个对比测试：(1)GridMol 和经典的分子可视化工具 JMV 的蛋白质显示性能比较；(2)GridMol 优化前后的建模性能比较。

测试环境：JRE1.6.0, Java 3D1.5, Windows XP, Intel 2.8 GHz Pentium 4 处理器，2 GB 内存。测试数据是 8 个 PDB 分子文件，每个分子的数据参数见表 1。

表 1　测试所用蛋白质数据

蛋白质(PDB ID)	余基数量	原子数量
1CQ0	28	413
1BYP	100	732
1C6X	199	1 514
1BVG	199	3 120
1F2E	808	5 948
3CCW	1 654	12 259
1R9N	2 957	24 028
1G0U	6 052	48 906

首先比较 GridMol 和 JMV 在 3 种显示模式下的内存消耗和渲染速度。内存消耗使用 JDK 中的 Jconsole 工具测量，另外，修改了 JMV 的源代码用于测量其渲染速度。

图 12~图 14 是 3 种模式下的内存消耗对比。GridMol 优势明显，合并不必要的 Shape3D 及使用自己实现的 3D 对象减少了内存消耗。如管状模式下，JMV 使用 Java 3D 提供的 Cylinder 类显示圆柱体，而 GridMol 中圆柱体实际上是一个长方体，通过调节光照使它有圆柱体的外观。JMV 圆柱体分辨率很高，对比单个圆柱体对象消耗的内存，JMV 远远大于 GridMol。如图 13 所示，管状模式下，JMV 最大的内存消耗达到 525 MB，而此时 GridMolk 只有 120 MB。

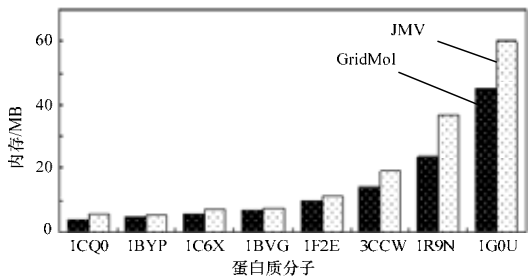

图 12　线模式内存对比

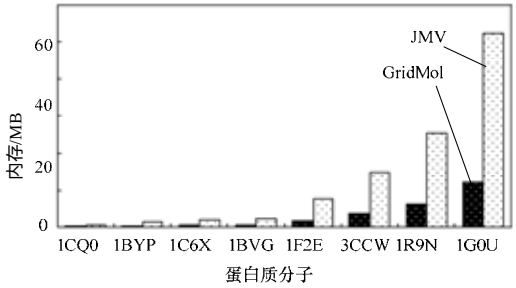

图 13　管状模式内存对比

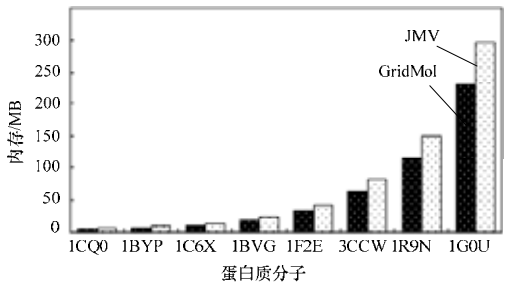

图 14　大球模式内存对比

图 15~图 17 是 3 种模式下渲染速度对比，FPS>100 的部分没有画出。在线模式下，两者差距不大，而在管状模式和大球模式下，GridMol 优势明显。在显示 1F2E 时，大球模式下 GridMol 的 FPS 是 JMV 的 2 倍，管状模式下 GridMol 的 FPS 是 JMV 的 4.5 倍。这都是 GridMol 精简不必要的 TG 对象达到的效果。JMV 没有建模功能，所以，对比了 GridMol 建模模式优化前后的显示性能来体现优化效果。图 18 给出了优化前后对蛋白质进行局部修饰操作时的渲染速度对比。

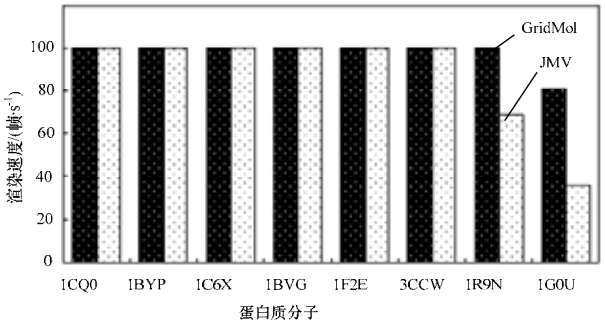

图 15　线模式渲染速度对比

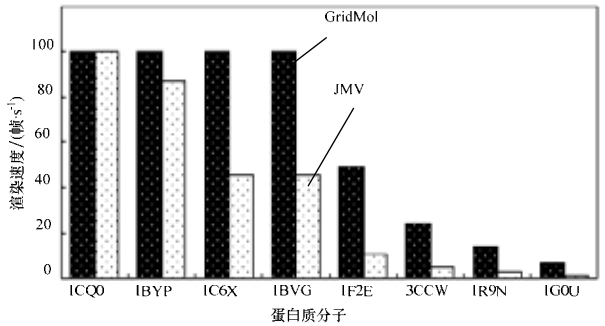

图 16　管状模式渲染速度对比

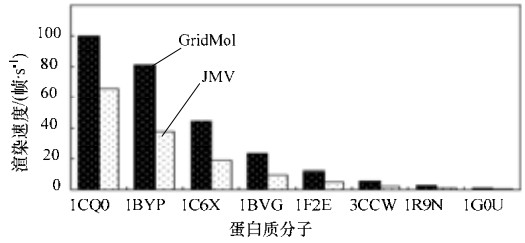

图 17　大球模式渲染速度对比

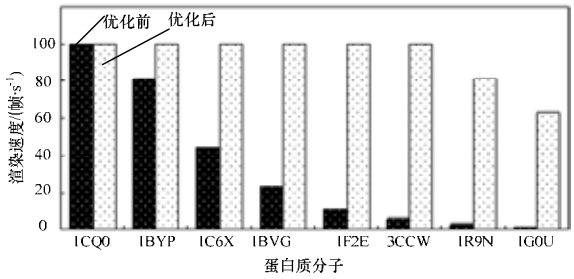

图 18　建模模式优化前后的渲染速度对比

图 18 优化后只在建模的局部使用球棒模式，其余部分用线模式，所以，显示时的 FPS 值与线模式的 FPS(图 15 中所示)很接近。在实际操作中，FPS<25 时才会让人有响应速度缓慢的感觉。优化前对 1BVG 分子做建模操作时 FPS 已经小于 25，而优化后对测试用到的最大分子进行建模，FPS 仍然大于 60，有极好的性能。

5　结束语

本文分析了用 Java3D 进行蛋白质显示建模的性能问题，通过对场景图的调整优化 GridMol 的性能并取得了良好的效果。但 GridMol 在大球模式下显示蛋白质大分子时还有较大的内存消耗，因此，下一步将在完善 GridMol 系统功能的同时继续研究提升性能的方法。

参考文献

[1] Sayle R A, Milner-White E J. RASMOL: Biomolecular Graphics for All[J]. Trends Biochem. Sci., 1995, 20(9): 374-376.

[2] Koradi R, Billeter M, Wuthrich K. MOLMOL: A Program for Display and Analysis of Macromolecylar Structure[J]. Journal of Molecular Graphics, 1996, 14(3): 51-55.

[3] Walther D. WebMol——A Java-based PDB Viewer[J]. Trends Biochem. Sci., 1997, 22(6): 274-275.

[4] JMV. Java Molecular Viewer[Z]. (2008-10-10). http://www.ks.uiuc.edu/Research/jmv/.

[5] Sun Yanhua, Shen Bin, Lu Zhonghua, et al. GridMol: A Grid Application for Molecular Modeling and Visualization[J]. Journal of Computer-aided Molecular Design, 2008, 22(1):119-129.

[6] 周海梦, 王洪睿. 蛋白质化学修饰[M]. 北京: 清华大学出版社, 1998.

[7] Berman H M, Westbrook J, Feng Zao, et al. The Protein Data Bank[J]. Nucleic Acids Research, 2000, 28(1): 235-242

编辑　张正兴