矩阵数据读写方法在系统仿真中的应用与研究

王 磊[1,2] 卢显良[2] 张 伟[1]
(电子科技大学电子科学技术研究院 成都 610054)[1] (电子科技大学计算机学院 成都 610054)[2]

摘 要 为提高软件仿真系统运行效率,缩短系统仿真运行周期,在仿真中引入了Sybase Open Client组件技术,提出了大容量矩阵数据读写方法,并通过实验和分析得出多Image数据写入方法。首先介绍了Sybase Open Client接口技术,然后详细阐述了Sybase数据访问控制结构、采用的仿真数据描述形式,最后对矩阵数据读写方法进行了研究,并针对矩阵数据写入方法进行了相关实验和结果分析。

关键词 Open client,矩阵数据,数据描述形式,Image
中图法分类号 TP311.52 **文献标识码** A

Application and Research of Matrix Data Read-write Method in System Simulation

WANG Lei[1,2] LU Xian-liang[2] ZHANG Wei[1]
(Research Institute of Electronic Science and Technology, University of Electronic Science and Technology of China, Chengdu 610054, China)[1]
(School of Computer, University of Electronic Science and Technology of China, Chengdu 610054, China)[2]

Abstract In order to enhance the efficiency of software simulation system and shorten the running periods of system, the technique of Sybase Open Client Component was introduced and the matrix data read-write method was proposed, multi-image data witting method was given finally through experiments and analysis. Sybase Open Client interface technique was introduced firstly, and then the access-control structure of sybase database, the data described form adopted in simulation was expatiated in sequence, finally the method of matrix data for read-write was researched and the matrix data write method was experimented and analyzed.

Keywords Open client, Matrix data, Data described form, Image

随着仿真理论和技术的发展,系统仿真已经成为一个多学科、多领域交叉与融合的研究方向。与此同时,仿真系统规模越来越大,系统复杂程度越来越高,因此仿真系统的效率问题已成为当前关注的热点之一。仿真数据作为系统仿真验证的基石,将伴随着每次系统仿真过程动态产生并大批量地存储到数据库中。如何缩减大容量数据的存取耗时,以提高仿真系统的运行效率,目前主要依靠两个方面因素:其一,主要依赖于主流数据库技术,如Oracle,Sybase,DB2等,数据库性能的优劣直接影响仿真数据的读写速度;其二,主要依赖于大容量仿真数据的读写方法,这是本文研究的重点。

本文基于Sybase数据库在数值仿真计算方面的优点,在项目中使用Sybase数据库作为仿真数据存储中心,采用Solaris9 OS作为系统运行环境,对数据访问控制结构进行了剖析,并结合Open Client应用接口技术对仿真系统数据接口层次进行了开发与研究,其中重点研究了大容量矩阵数据的读写方法,并进行了相关实验与分析。

1 Open Client

Sybase提供了两组产品,分别是Open Client和Open Server,能够自定义开发客户端和服务器端应用程序。

Open Client是Adaptive Server体系结构的一部分,该体系结构提供了网络透明性。只要使用同一种网络协议进行通信,此连接层允许客户应用程序(无论是什么平台)与服务器通话。如果一个服务器能够利用多种协议进行通信,这就意味着这些客户应用程序可以使用该服务器已经使用的任一种协议与其通信。这也意味着无论使用何种协议,应用程序都是用同一种方式进行编码[1]。

Open Client提供接口给自定义应用程序、第三方产品和其他Sybase产品,完成与Adaptive Server和Open Server的通信。Open Client主要包括两部分:(1)编程接口;(2)网络服务。

对于编程接口,提供相应的接口库,包括:Client-Library,DB-Library,CS-Library。

对于网络服务,包括Net-Library,对特殊的网络协议提供支持,包括TCP/IP等。

按照函数库类别可以进一步划分为以下3个部分。

(1)DB-Library:提供了用于编写客户端应用程序的接口集,包括批处理库和两层结构库。

(2)Client-Library:同样提供了用于编写客户端应用程序的接口集。该库的设计目的在于提供对游标和其他高级特性

到稿日期:2008-10-31 返修日期:2009-01-22

王 磊(1980-),男,博士研究生,研究实习员,CCF会员,主要研究方向为分布式仿真、计算机系统结构,E-mail:cmyan@uestc.edu.cn;卢显良男,教授,博士生导师,主要研究方向为计算机系统结构;张 伟 男,博士,副研究员,主要研究方向为雷达系统仿真等。

的服务。

(3)CS-Library:同时为客户端和服务器应用程序提供接口。

DB-Library/C 包含了一些 C 语言函数和一些宏定义,使得一个程序能够与 SQL 服务器和 Open Server 连接。这也是我们在开发过程中用到的接口。

如图 1 所示,在客户端利用 Open Client 的连接性编写应用程序。这些特殊的 Open Client 库接受任意请求并将它们转换为一种称为 TDS(Tabular Data Stream,表列数据流)的通用格式的数据,Net-Library 可以理解这种 TDS 格式的数据,并通过指定的网络协议传送这些 TDS 数据。Net-Library 是一个驱动程序,用于从 Client-Library 或网络上接受 TDS 数据。在服务器端存在一个 Server-Library,这是 ASE 可执行程序的一部分,它也与 Net-Library 通信。

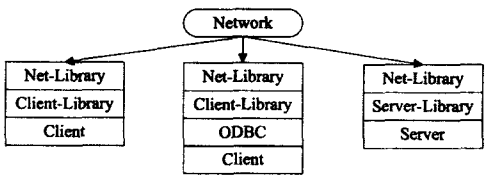

图 1 Open Client 基本结构

Client-Library 将客户应用程序的请求转换为一种通用格式的数据,该数据随后由 Net-Library 根据当前所使用的网络协议作进一步转换。在服务器端,Net-Library 将到来的网络数据流转换为 Server-Library 可以理解的格式,接着 Server-Library将该请求转换为特定的服务器调用。当 Adaptive Server 产生结果时,这些结果将通过相同的连接层按照相反的顺序传送到客户端;Server-Library 将产生的结果转换为一种通用格式;Net-Library 根据当前所使用的网络协议作进一步转换;Net-Library 将数据从网络协议转换为通用格式;Client-Library 将数据从通用格式转换为客户端应用程序可以理解的一种特定格式。接下来,客户端应用程序便可显示这些结果。因此,在不需要替换或重建所有其他层的情况下,可以替换该转换过程的任意部分(例如,编写一个新的客户应用程序,或者使用一种新的网络协议)[1]。

2 数据访问控制结构

控制结构(Control Structure)是指应用程序在与 SQL Server 建立连接和执行命令之前必须分配一些结构,程序通过设置结构的属性值存储相关的环境配置、连接信息等,以实现与 SQL Server 的通讯,并控制程序对数据库的操作动作。Sybase Open Client API 提供了数据访问三层控制结构(如图 2 所示),由外至内划分为:

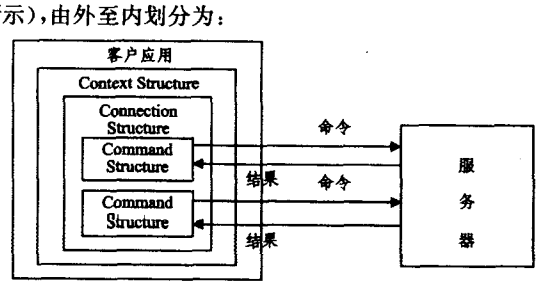

图 2 数据访问控制结构

(1) Context 结构(上下文环境结构)CS_CONTEXT[2]

在应用程序初始化 Client_Library 时必须分配 CS_CONTEXT 结构,它存储了操作环境的配置信息,例如语言类型等。通常一个应用程序至少有一个 CS_CONTEXT 结构,也可以有多个 CS_CONTEXT 结构。

(2) Connection 结构(连接结构)CS_CONNECTION[2]

一个连接结构存储了与 SQL Server 一次连接的信息,例如用户名、口令等。一个 CS_CONNECTION 结构与 SQL Server 的一个进程相对应。在一次连接中,一条命令的执行结果必须处理完,才能发送另一条命令(ct_cursor(CS_CURSOR_OPEN)命令除外)。如果想要在处理一条命令过程中执行另一条命令,则必须建立两个连接,再分配一个连接结构。

(3) Command 结构(命令结构)CS_COMMAND[2]

命令结构存储的是一系列送往 SQL Server 的命令。SQL Server 处理这些命令并返回结果。

三者的关系可以表述为:Context 结构是父结构;Connection 结构是 Context 结构的子结构,一个 Context 结构控制多个 Connection 结构。Command 结构是 Connection 结构的子结构,一个 Connection 结构控制多个 Command 结构。

3 数据描述形式

3.1 简单数据与矩阵数据

雷达仿真系统是一个分工协作的仿真系统集,涵盖了诸多功能仿真子系统,如场景生成子系统、雷达资源调度子系统、回波生成子系统、信号处理子系统、数据处理子系统、目标识别子系统、成像子系统、显控子系统等等。各仿真子系统在仿真过程中生成的仿真数据形式各不相同,比如成像子系统要对目标成像,其仿真数据表现为二维矩阵格式,用以描述成像样本。因此,根据雷达仿真系统构成特点,将仿真数据按照其表现形式划分为两种:简单数据和矩阵数据。

约定仿真数据采用矩阵表现形式:$M_{m\times n}(m>0,n>0)$。其中,m 表示矩阵数据对应的行,n 表示矩阵数据对应的列。当 $m=n=1$ 时,$M_{1\times 1}$ 被称为简单数据;当 $m\neq 1,n\neq 1$ 时,$M_{m\times n}$ 被称为矩阵数据。仿真过程中,读写数据常用的数据类型有整型(int)、高精度实型(double)、图像(Image)3 种。对于一个 int 或 double 数据,可以采用简单数据格式,通常描述编号、个数、距离、角度、速度等单维属性;对于 Image 数据,可以采用矩阵数据格式,通常描述回波、杂波、噪声、一维像、二维像等二维属性[3]。

3.2 简单数据与矩阵数据的转化关系

简单数据对应的数据类型包括整型(int)、实型(double),矩阵数据对应的数据类型为图像(Image)。图像类型(Image)数据可以抽象表示成一个串数据形式,即一个单行 n 列($n\geq 2$)的数据。如图 3 所示,有一个 3×3 的矩阵数据 $M_{3\times 3}$,图中标定的 0—8 表示该位置数据在串数据中对应的逻辑位序关系。0—8 位置上的每个数据可以是 int,double 等简单数据类型。显而易见,矩阵数据元素是由简单数据构成的。对于一个由 int 类型作为矩阵元素构成的矩阵数据,在程序接口中可以采用以下的声明方式[3]:

int * aMatrix = new int [$m \times n$];

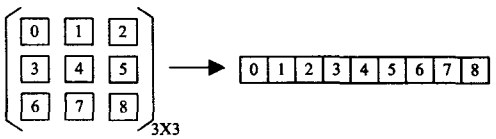

图 3　矩阵数据抽象形式

4　Image 数据

Image 类型数据的读写方法对软件仿真系统运行效率尤为重要，特别是诸如回波生成子系统模拟产生回波、成像子系统对目标进行成像处理等连续仿真过程中都将产生大容量的仿真数据，如何快捷地存储和读取这些矩阵数据，尽可能提高软件系统的读写效率，是研究的一个重点。

4.1　Image 数据类型

文本(text)/图像(Image)是 SQL Server 中的数据类型，用于存储大量的非结构化的数据。由于文本图像数据类型的字段中需要大量的非结构化的数据，因此文本图像数据没有实际存放在数据库的表中，而使用一种页链方式存取。表中字段存放链表下一页的地址，链表中的页存放实际数据。SQL Server 根据数据的大小自动分配链表的页面，链表中的一页为 2k。图 4 为文本/图像数据存储方式示意图。

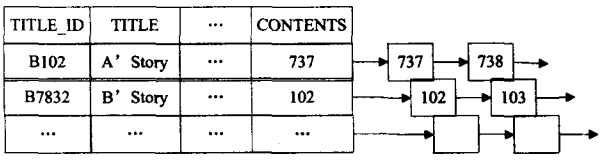

图 4　文本/图像数据存储方式

Image 和 text 数据类型是 SQL Server 设计用来控制大的 Image(或 text)数据的。text 数据类型可以支持到 2G 字节的可打印字符，Image 数据类型可以支持到 2G 字节的二进制数据。

由于这两种数据类型占用空间如此之大，因此 text 和 Image 数据实际上并非保存在数据库中数据表里。相反，保存在表中的是一个指向该 Image(或 text)数据的指针。这个指针称之为“Image pointer”。

为了保证竞争应用不能除去另一个应用在数据库中的修改，因此对每一个 Image(或 text)数据列配置一个时间戳。这个时间戳称之为“Image timestamp”。

4.2　矩阵数据读写

软件仿真系统采用了分布式结构，划分为子系统集{SSG，RCP，RG，SP，DP，TR，ISAR，RGWB，SC，DCP，MC}[4]。MC 负责传递主模块间的消息，目的在于确保雷达系统仿真流程的正确执行；SC 负责监控系统仿真运行状态；DCP 负责雷达终端的航迹显示；SSG 负责模拟生成场景；RCP 负责雷达资源的调度；DP 负责对航迹数据的处理；TR 负责目标识别；RG 负责回波的产生；SP 负责回波目标信号的检测；ISAR 负责中带目标成像；RGWB 负责宽带回波目标成像。在上述子系统集中，读写数据比较而言，写入数据花费的时间远远大于读出数据的执行时间。因此，研究写入数据的问题，将对提高系统执行效率、缩短系统仿真时间有着重要的意义。

仿真系统中仿真执行时间较长的模块是：RG，SP，ISAR，RGWB，其执行时间长的原因归结为：

(1)数据量计算大，计算复杂；

(2)写入操作频繁；

(3)写入数据形式多为 Image 数据(或者称为矩阵数据)，写入数据量很大(数量级达到数十～数百兆)；

(4)访问的数据表通常为多 Image 字段(≥2 个)数据表。对于多 Image 字段数据表的读写，采用不同的读写方法，耗费时间的差别也比较大；

(5)Image 数据的读写方式与非 Image 数据的读写方式不同，这也是直接影响读写时间的原因之一；

(6)仿真过程中 CPU 占用率在变化，CPU 占用率越低，读写时间也越长；

(7)其他原因等。

通常来讲，矩阵数据读取的关键步骤是：根据读取数据总长度多次调用 dbreadtext[5](dbproc，存放空间，矩阵数据长度)方法，只需特别注意逻辑地址位序的正确匹配即可。由于每次读取数据块的上限可以灵活设置，因此须考虑一次调用 dbreadtext 读取的数据量和调用 dbreadtext[5] 读取数据的次数(这在后文实验环节进行了讨论)。

矩阵数据写入步骤可以归结为：

(1)在数据表中插入除 Image 字段外的其他字段数据；

(2)更新这一行中 Image 字段为 NULL，以创建有效的文本指针；

(3)选择 Image 字段，将其文本指针地址和时戳值传递给 DBPROCESS[5,6]；

(4)调用 dbtxtptr[5](dbproc，column)从 DBPROCESS[5,6] 获取指定 Image 字段的文本指针；

(5)调用 dbtxtimestamp[5](dbproc，column)从 DBPROCESS 获取指定 Image 字段的时戳值；

(6)调用 dbwritetext[5](dbproc，size，text)一次写入 Image 数据到 SQL Server；或使用 dbmoretext[5](dbproc，objname，textptr，textptrlen，timestamp，log，size，text)分批多次写入 Image 数据到 SQL Server。

对于一次只有一个 Image 数据写入情况，可以直接采用上面给出的矩阵数据写入方法。但通常情况下，遇到的问题是一次要完成单行多个 Image 数据写入操作(对于一次写入一个 Image 数据的方法称之为“单 Image 数据写入”，将一次写入多个 Image 数据的方法称之为“多 Image 数据写入”)。

4.3　单 Image 数据写入

使用单 Image 写入方法完成单行多个 Image 字段数据表的写入通常采用以下策略：

(1)建立数据库连接，初始化 Image 数据写入环境；

(2)写入一个 Image 字段的数据；

(3)关闭数据库连接；

(4)按照步骤(1)－(3)执行下一个 Image 字段数据的写入。

上述策略存在一个明显的问题，即要完成一条多 Image 数据行的写入，必然要多次执行“建立数据库链接→写入一个 Image 数据→关闭数据库连接”。对于一条多 Image 数据行，其数据链接的形式都是相同的，因而在重复建立相同链接和关闭相同链接的操作时耗费了一定的时间。通常访问数据库时，建立数据链接是比较费时的。因此，如果能够在一次访问

中完成一条多 Image 数据行的 Image 数据的写入，将会在一定程度上缩短一条多 Image 数据行的实际执行时间。

4.4 多 Image 数据写入

使用多 Image 写入方法完成单行多个 Image 字段数据表的写入，就是在建立数据库连接、初始化 Image 数据写入环境之后，按照接口参数传递顺序依次写入每个不同的 Image 字段数据，最后关闭初始数据连接。这样可以改进单 Image 写入方法执行过程中因单 Image 写入方法多次调用而造成因为数据库连接多次建立、关闭在时间上的耗费。同时，多 Image 写入方法的调用一次可以对应单 Image 写入方法调用多次的执行效果，在程序风格上来讲更加简洁、明了，方便阅读，更加易于交流。

5 实验与分析

测试环境为 SunOS(Solaris 5.9)，以 Sybase12.5.1 作为后台数据库服务器，并结合 Sybase OpenClient 应用接口编写 C/C++程序。在用户数据库 MyTestDB 中创建一个结构如表 1 所列的中窄带回波输出表，该表有 14 个 Image 类型字段。

表 1 中窄带回波输出信息表

字段	类型	说明
ID	Int	主键，唯一标识用
Echo_SumR	Image	回波仿真数据
Echo_SumI	Image	回波仿真数据
Echo_AziR	Image	回波仿真数据
Echo_AziI	Image	回波仿真数据
Echo_EleR	Image	回波仿真数据
Echo_EleI	Image	回波仿真数据
Target_EchoR	Image	回波仿真数据
Target_EchoI	Image	回波仿真数据
Noise_SgnR	Image	回波仿真数据
Noise_SgnI	Image	回波仿真数据
Clutter_SgnR	Image	回波仿真数据
Clutter_SgnI	Image	回波仿真数据
Gate_Tran_SgnI	Image	回波仿真数据
Gate_Tran_SgnR	Image	回波仿真数据

对两种方法进行以下命名：

- WriteMultiImage_Method(多 Image 写入方法)
- WriteSingleImage_Method(单 Image 写入方法)

WriteMultiImage_Method 调用 1 次并完成 14 个 Image 字段数据的写入；WriteSingleImage_Method 调用 14 次，每次完成 1 个 Image 字段数据的写入。在测试结果数据表中，用“行 * 列”表示矩阵单元，单行数据量表示为“行 * 列 * double * 矩阵个数”，每次写入的总数据量表示为“行 * 列 * double * 矩阵个数 * 记录行数”。对两种写入方法的研究，其目的就是希望提高回波数据的写入速度，缩减仿真周期，提高仿真系统的运行效率。

对这两种写入方法进行实测，分别设置数据块上限(数据块上限就是执行一次 dbwritetext 动作时写入数据量的最大值。当写入相同数据量时，数据块上限设置太小，则执行 dbwritetext 动作次数将会增多；反之，数据块上限设置太大，则执行 dbwritetext 动作次数将会减少)为 32k，64k，1M 容量限制条件，其完成数据写入耗费时间测量结果如下：

(1)采用 WriteMultiImage_Method 方法的实测结果如表 2、表 3、表 4 所列。

表 2 每次写入数据块上限为 32k 条件下 WriteMultiImage_Method 的执行结果

矩阵单元	个数	单行数据量(B)	行数	总数据量(MB)	写入时间(s)			
1*512	14	1*512*8*14	1	0.055	1s	1s	1s	1s
10*512	14	10*512*8*14	10	5.5	13s	13s	13s	13s
50*512	14	50*512*8*14	10	27.3	55s	55s	54s	54s
50*512	14	50*512*8*14	100	273.3	548s	549s	548s	549s
100*512	14	100*512*8*14	100	546.6	1078s	1081s	1082s	1085s
512*512	14	512*512*8*14	10	280	558s	558s	560s	556s

表 3 每次写入数据块上限为 64k 条件下 WriteMultiImage_Method 的执行结果

矩阵单元	个数	单行数据量(B)	行数	总数据量(MB)	写入时间(s)			
1*512	14	1*512*8*14	1	0.055	1s	1s	1s	1s
10*512	14	10*512*8*14	10	5.5	13s	13s	13s	13s
50*512	14	50*512*8*14	10	27.3	54s	56s	55s	55s
50*512	14	50*512*8*14	100	273.3	557s	563s	551s	552s
100*512	14	100*512*8*14	100	546.6	1078s	1125s	1093s	1123s
512*512	14	512*512*8*14	10	280	586s	556s	563s	580s

表 4 每次写入数据块上限为 1M 条件下 WriteMultiImage_Method 的执行结果

矩阵单元	个数	单行数据量(B)	行数	总数据量(MB)	写入时间(s)			
1*512	14	1*512*8*14	1	0.055	1s	1s	1s	1s
10*512	14	10*512*8*14	10	5.5	14s	14s	14s	14s
50*512	14	50*512*8*14	10	27.3	60s	59s	57s	58s
50*512	14	50*512*8*14	100	273.3	564s	561s	552s	556s
100*512	14	100*512*8*14	100	546.6	1082s	1080s	1095s	1103s
512*512	14	512*512*8*14	10	280	569s	567s	574s	570s

(2)采用 WriteSingleImage_Method 方法的实测结果如表 5、表 6、表 7 所列。

表 5 每次写入数据块上限为 32k 条件下 WriteSingleImage_Method 的执行结果

矩阵单元	个数	单行数据量(B)	行数	总数据量(MB)	写入时间(s)			
1*512	14	1*512*8*14	1	0.055	2s	1s	1s	1s
10*512	14	10*512*8*14	10	5.5	14s	15s	17s	17s
50*512	14	50*512*8*14	10	27.3	66s	66s	63s	61s
50*512	14	50*512*8*14	100	273.3	610s	575s	572s	573s
100*512	14	100*512*8*14	100	546.6	1111s	1107s	1105s	1107s
512*512	14	512*512*8*14	10	280	550s	561s	564s	558s

表 6 每次写入数据块上限为 64k 条件下 WriteSingleImage_Method 的执行结果

矩阵单元	个数	单行数据量(B)	行数	总数据量(MB)	写入时间(s)			
1*512	14	1*512*8*14	1	0.055	1s	1s	1s	1s
10*512	14	10*512*8*14	10	5.5	15s	15s	15s	16s
50*512	14	50*512*8*14	10	27.3	57s	57s	56s	57s
50*512	14	50*512*8*14	100	273.3	581s	573s	576s	597s
100*512	14	100*512*8*14	100	546.6	1104s	1105s	1101s	1093s
512*512	14	512*512*8*14	10	280	560s	558s	571s	577s

表 7 每次写入数据块上限为 1M 条件下 WriteSingleImage_Method 的执行结果

矩阵单元	个数	单行数据量(B)	行数	总数据量(MB)	写入时间(s)			
1*512	14	1*512*8*14	1	0.055	1s	1s	1s	1s
10*512	14	10*512*8*14	10	5.5	16s	16s	16s	16s
50*512	14	50*512*8*14	10	27.3	59s	60s	59s	59s
50*512	14	50*512*8*14	100	273.3	596s	593s	565s	564s
100*512	14	100*512*8*14	100	546.6	1097s	1104s	1097s	1096s
512*512	14	512*512*8*14	10	280	553s	559s	560s	555s

分析以上实测数据，我们发现设定块数据量大小为 32k

作为一次写入最大数据量界值，其综合实际效果较好。分析其原因有以下两点：

（1）每次写入数据块上限值越大，则调用 dbwritetext 次数将会减少。但由于数据块上限值大，势必造成每次执行 dbwritetext 所花时间增多；

（2）每次写入数据块上限值越小，则调用 dbwritetext 次数将会增多。但由于数据块上限值小，因此每次执行 dbwritetext 花费的时间也较少；

下面将每次写入数据块大小界值设定为 32k，对不同量值矩阵数据写入进行了压力测试。对同一种条件下的数据进行 10 组连续测试，求其平均值，作为上述两种写入方式的比较指标，之所以进行多组测试，是因为数据库服务器在不同时刻的性能并非完全相同。考虑到数据库服务器性能存在一定的波动差、网络传输状况变化（客户端程序与数据库服务器架设在不同的主机上或者不同的磁盘上，因此数据库访问操作其实是进行远端的数据访问操作，网络传输状况也是影响因素之一），进行多组数据写入测试，分别得到每组测试的执行时间。求其平均值并进行比较，将会更为合理准确，从而更好地反映出两种 Image 数据写入方法的执行效果。表 8 列出了数据测试对比结果。

表 8　两种 Image 写入方法的对比表

矩阵单元	个数	行数	数据总量	写入时间(s) 每次写入数据块大小界值设定 32k(系统默认)
1	512	14	1	0.055	/ / / / / / / / / / 多个 Image 一次写入时间平均值 1s / / / / / / / / / / 一次写入一个 Image 时间平均值 1s
10	512	14	10	5.5	12 12 13 13 12 13 12 12 12 12 多个 Image 一次写入时间平均值 12.3s 14 16 15 15 15 15 15 15 14 14 一次写入一个 Image 时间平均值 14.8s
50	512	14	10	27.3	54 55 54 54 54 53 55 54 54 54 多个 Image 一次写入时间平均值 54.1s 54 55 55 55 55 55 55 54 55 55 一次写入一个 Image 时间平均值 54.8s
50	512	14	100	273.3	545 546 536 536 535 536 536 542 536 536 多个 Image 一次写入时间平均值 538.4s 553 549 550 551 555 551 550 550 551 551 一次写入一个 Image 时间平均值 551.1s
50	512	14	200	546.6	1081 1100 1083 1074 1074 1071 1079 1085 1082 1082 多个 Image 一次写入时间平均值 1079.1s 1289 1104 1098 1099 1100 1099 1109 1110 1105 1099 一次写入一个 Image 时间平均值 1121.2s
100	512	14	100	546.6	1069 1060 1069 1072 1063 1065 1058 1056 1057 1054 多个 Image 一次写入时间平均值 1062.3s 1070 1072 1073 1072 1074 1099 1090 1073 1073 1072 一次写入一个 Image 时间平均值 1076.8s

从表 4－表 8 的实测数据可以看出，两种方法在写入时间小于 1s 时，由于采用的时间精度不够，无法进一步对比。因此这里只对后面几组数据进行图形对比，结果如图 4－图 8（红色曲线表示单 Image 数据写入方法、蓝色曲线表示多 Image 数据写入方法、横坐标表示测试的次数序号、纵坐标表示数据写入耗时，单位为秒）所示。

由此看出，WriteMultiImage_Method 方法要比 WriteSingleImage_Method 方法更加省时，执行时效性更高。这表明多 Image 写入方法要比单 Image 写入方法更加有效，更加优化。在实际项目中，该优化后的矩阵数据写入方法已经得到推广使用。

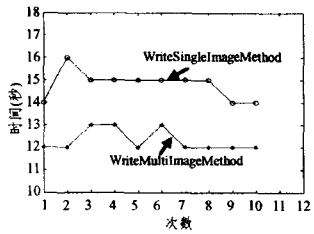

图 4　写入 10 行 14＊(10＊512)矩阵数据/行

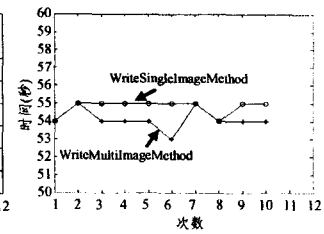

图 5　写入 10 行 14＊(50＊512)矩阵数据/行

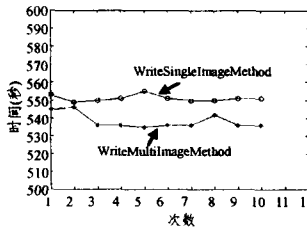

图 6　写入 100 行 14＊(50＊512)矩阵数据/行

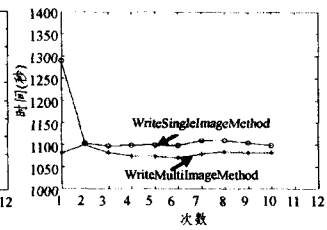

图 7　写入 200 行 14＊(50＊512)矩阵数据/行

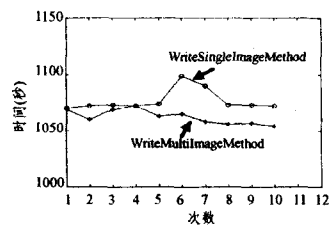

图 8　写入 100 行 14＊(100＊512)矩阵数据/行

结束语　本文引入 Sybase Open Client 组件技术，针对矩阵数据读写方法进行相关研究和实验分析，特别是多 Image 数据写入方法的使用大大提高了软件仿真系统的运行效率，缩短了仿真运行周期。这对其它领域系统仿真数据层接口开发将会是一个良好的借鉴与启发。随着系统仿真研究工作的不断深入，我们将会对仿真数据接口系统进行不断扩充、丰富和完善，同时对其进行大量实测以达到更加优化。

参 考 文 献

[1] Chang J G A, Tyrrell P G G. Sybase ASE 12.5 管理员指南[M]. 余安萍，俞俊平，译. 北京：电子工业出版社，2004

[2] 阎晓青，白占林，罗润升，等. Sybase Open Client 应用开发指南[M]. 北京：中国水利水电出版社，1997

[3] 王磊. 分布式雷达系统仿真应用研究[D]. 成都：电子科技大学，2008

[4] 王磊，陈明燕，张伟，等. 系统仿真数据接口归一化应用研究[J]. 系统仿真学报，2009(1)

[5] DB-Library™/C Reference Manual. Sybase Inc.，2003

[6] Client-Library™/C Reference Manual. Sybase Inc.，2003