分布式 PowerBuilder 应用程序的调试策略和方法*

Debug Strategy and Method of Distributed PowerBuilder Application Program

胡振宇　　林士敏
Hu Zhenyu　　Lin Shimin

(广西师范大学计算机科学系　桂林　541004)
(Comp. Sci. Dept., Guangxi Normal University, Guilin, 541004)

摘要　由于分布式应用的独特性，使得分布式应用程序的调试远比单机应用程序的调试复杂得多。文中对分布式应用程序提出一种“先单机单进程调试，次单机多进程调试，最后再多机远程调试”的调试策略，给出在 PowerBuilder 开发环境下的实施方法，并对分布式 PB 应用程序的常用调试工具、技术和调试中应注意的问题进行探讨。所述策略和方法在开发《广西普通高校计算机等级考试管理系统 MCT》网络版中得到成功使用，证明是正确可行的。

关键词　分布式应用程序　PowerBuilder　程序调试
中图法分类号　TP 311.5

Abstract　Debugging of distributed application is much more complex than that of a single program because of the unique characters of distributed application. A debugging strategy for debugging process is put forward as " single computer and single course debugging, then single computer and multi-courses debugging, and lastly multi-computers and remote debugging". The common debugging tools, operation and problems that have been faced in debugging process of distributed PowerBuilder application are discussed. The strategy and method mentioned have been run successfully in Management System of Computer Degree Test for Guangxi Universities (MCT) (network version).

Key words　distributed application program, PowerBuilder, debugging

1　分布式应用程序的 N 层结构模式

分布式处理提供了一种方式来分解事务处理的位置，这种分解以对客户透明的方式工作。应用程序利用来自网络的资源来完成处理。一个分布式应用程序主要由以下部件组成。

(1) 表示逻辑：应用程序的图形用户界面 (GUI)，用来管理键盘与鼠标输入、屏幕绘制及窗口操作；(2) 商务逻辑：以计算机化的形式体现的商务处理逻辑。主要用第三代或第四代高级语言编写；(3) 数据库逻辑：包括对应用程序中的数据的处理以及由 DBMS 完成的数据库中的数据的处理。

这种部件交叉出现就形成所谓的 N 层应用模式，N 层结构主要有以下特点：(1) 通过把

2001-06-25 收稿。
*广西教育厅高教处专项资金资助，广西教育厅立项科研项目(编号：1999 年自筹 19)。

大量的应用处理移出客户方，可以减轻客户方计算机的负担，使客户方可利用服务器来获得更快的处理速度，而无需对客户机做内存与 CPU 的升级；(2) 可以对应用服务器做应用修改而不必对每个客户机做修改，易于管理软件的修改和升级；(3) 通过把数据移出客户应用，使客户机通过远程与数据库通信，切断客户与数据库的直接通信，有利于加强数据的安全性。

2 PowerBuilder 开发环境中的常用调试工具和技术

2.1 PowerBuilder 调试器

PowerBuilder 带有内置调试工具 debuger，可帮助开发者找出任何应用程序的脚本中的错误。PowerBuilder 6.0 的调试器给用户提供 1 个灵活和强大的环境。调试器允许用户设置断点，在调试模式下运行应用程序时，PowerBuilder 在执行含有断点的 PowerScript 语句前将进程挂起。这时就可单步执行每一行代码，观察执行了什么代码，查看什么对象加载到了内存中，以检查当前作用域内的变量。调试能使用户查看变量在断点处的值，且可以改变变量的值。

PowerBuilder 调试器有 4 种使用方法：(1) 第一种方法也是使用最多的方法，就是打开调试器，在合适的地方设置断点，然后在调试模式下运行程序，程序将在设置的第一个断点处停止。然后进入调试模式；(2) 不打开调试器设置断点，而是直接在脚本中放一条触发调试模式的语句 DebugBreak () 函数。当在 PowerBuilder 开发环境下执行时，程序运行到 DebugBreak () 就停止并打开调试器进入调试模式；(3) 第三种方法要首先在 System Option 中将 PowerBuilder 设置成为实时调试模式 (just-in-time-debugging)，然后，当用户在 PowerBuilder 开发环境中以正常模式运行程序遇到一个错误时，PowerBuilder 就自动进入调试模式；(4) 最后一种用法是在开发环境中以正常方式运行程序，此时 PowerBuilder 会缩成图标在任务栏上，当想在某个模块进入调试模式时，单击任务栏上 PowerBuilder 图标，选择 Debug 按钮，即可进入调试模式。后两种方法被称为实时调试方法。

2.2 PBDEBUG

PowerBuilder 还带有中另一个实用程序，用来帮助开发者调试自己的程序，这就是 PBDEBUG。PBDEBUG 跟踪对象的创建和删除以及脚本、系统函数、全局函数、对象层函数和外部函数的执行，在利用 PBDEBUG 之前，必须将程序编译成可执行文件。若将程序编译为机器码，必须在 System Option 中选中 Enable PBDebug Tracing 选取项，或在生成可执行文件时选择 Trace Information 和 Error Context，以告诉 PowerBuilder 编译时加上跟踪信息在可执行文件上（对于原生码可执行文件，跟踪信息总是加在可执行文件上）。这样生成的可执行文件夹以如下的格式执行：

C：\applican _ path\applican _ name. exe pbdebug

其中的 applican _ path 是可执行文件的路径，applican _ name. exe 是可执行文件名。当程序执行完毕，即会在可执行文件的目录下生成一个名为 applican _ name. dbg 的跟踪文件。此文件将显示每个对象创建的顺序、被执行的事件、特定的事件的执行行数以及每个对象的删除情况。此文件还可告诉用户许多关于事件和事件在何时运行的情况。这可以帮助知道脚本/函数被执行了多少次，也可以为优化程序确定区域。如果程序中有定时器，应尽快确定感兴趣的区域，否则可能会由于定时器的运行使跟踪文件很快变得十分大。

2.3 数据库跟踪

在确定与数据库的连接时，PowerBuilder 也提供了一个很有用的跟踪技术。在设置事务处

理对象的属性时，在DBMS属性值前加上关键字TRACE，这样，当程序执行时，PowerBuilder将会生成一个跟踪日志文件，记录程序执行过程中的SQL处理。

2.4 SQLPreview事件

如果怀疑DataWindow检索和更新数据时产生的SQL语句不正确，可用DataWindow控件的SQLPreview事件，在把SQL语句传送给数据库之前进行查看。

SQLPreview事件在调用Retrieve（）、Updata（）、ReselectRow（）函数之后，且在SQL传送给数据库之前触发，Retrieve（）和ReselectRow（）函数的每次调用只触发SQLPreview事件一次，而Updata（）函数则每更新一行就触发一次。

捕捉SQL语句的方法是利用SQLPreview事件中的参数SqlSyntax，可在此事件中将此参数的值输出，以便核查。

2.5 DBError事件

当DataWindow控件调用Updata（）函数并且产生错误时，此事件被触发，可作用该事件的Row和Buffer 2个参数，来显示出错的行号和该行所在DataWindow缓冲区。据此可再查看传送给数据库的值和由Updata（）函数产生的SQL语句。在此事件以及上面的SQLPreview事件中还可利用GetItemStatus（）函数来获得行或列的状态，由此可判断PowerBuilder是否为应用程序产生了所需的SQL语句。

2.6 屏幕消息显示

使用文本编辑控件或MessageBox（）函数，将脚本中一些键标变量的值显示在屏幕上，由此可判断脚本运行的情况。使用模态窗口或MessageBox（）函数是唯一可以使多任务环境中的后台程序挂起的方法。

2.7 Connection对象的Error事件

如果一个客户请求在服务器上发生了错误，则PowerBuilder为此客户线程生成一个Error结构，并将此情况发回到客户端，触发客户端的Connection对象的Error事件。在客户端的Connection对象的Error事件中有几个参数，可以提供错误码、错误文本、产生错误的对象名以及错误发生处的脚本等错误信息。另外此事件还提供一个参数可以用来指定出错后采取的动作，开发者可在此事件中对错误进行处理。

2.8 ConnectionObject类的Trace属性

分布式PowerBuilder中ConnectionObject类（包含Transport和Connect对象）有一个Trace属性，开发者可用此属性，在应用程序执行期间获得系统活动的记录和对客户/服务器的监控等调试信息。

3 分布式PowerBuilder开发中的调试策略和方法

在一个分布式环境中调试作为客户机或服务器的PowerBuilder应用程序带来了一些独特的问题。一个分布式应用程序需要由客户机和服务器的协作才能完成指定的任务，因此对于一个分布式系统来说调试的难点在于如何确定错误出现在客户机还是服务器。在正常情况下分布应用的客户机和服务器是在不同的进程运行的，若用非分布式的PowerBuilder调试器进行调试，则会出现不能对客户机和服务器同时调试的尴尬局面。因此调试分布式应用程序的首要关键在于将客户机和服务器统一到同一个进程。调试的策略应该是先单机单进程调试，次单机多进程调试，最后再多机远程调试。

3.1 单机单进程调试

在分布式 PowerBuilder 的调试和诊断工具中，最著名最有力的当属 PowerBuilder 调试器，但 PowerBuilder 调试器目前只能处理单进程的本地调试，要使用 PowerBuilder 调试器调试分布式 PowerBuilder 应用程序，首先要将客户机和服务器统一到本地同一个进程。为此，可以将客户机和服务器作成单一的应用程序，调试通过后再分开。这种情况下，为了将客户机和服务器统一到同一个进程，需要对客户机和服务器程序进行较大的修改，等调试后又要改回来，并且，单进程调试的通过并不能保证多进程或远程调试通过。因此这是一种费时费力且效率低的方法。一种重要的技巧是使用 PowerBuilder 的本地回送驱动程序——Local 驱动程序进行单进程符号处理。

当使用 Local 驱动程序时，PowerBuilder 将以本地方式并在与客户机应用程序相同的进程内模拟一个远程服务器，处理所有的远程服务请求，将通常在分布式处理期间发生的间接对象引用转变为以本地方式服务的直接引用。这样由 Local 驱动程序将客户机和服务器统一到同一个进程，就可以使用 PowerBuilder 调试器进行设置断点、单步运行、查看变量等在非分布式环境中的所有调试工作。使用 Local 驱动程序调试时应注意以下几点：

(1) 因为 PowerBuilder 接管了所有通常由传输对象 (Transport Object) 处理的通信函数，所以应用中就不需要创建传输对象或执行任何与使用传输对象相关的活动。服务器组件也不要提交传输对象的 Listen () 函数，否则请求将失败；

(2) 客户机和服务器应用程序是运行在一个单一的进程中的。因此应用中的客户与服务器组件使用相同的变量上下文和数据空间，这可能会导致使用冲突；

(3) 由于 PowerBuilder 将远程请求变为本地的直接引用，所以在客户组件中不能使用远程代理，而应将服务器组件中的远程对象所在的库加到客房机应用程序的库搜索路径中。并将远程代理所在的库从搜索路径中删除或放在远程对象的后面；

(4) 连接对象 (ConnectionObject) 的 Application、Location 和 Options 属性值被忽略，可保持最初开发时的设置；

(5) 为了提高系统的吞吐量，脚本中可能会使用由 POST 关键字触发的异步函数或事件调用，这会使在使用 PowerBuilder 调试器时不能及时跟踪这些函数或事件的执行情况，可暂将这些异步函数或事件调用改为同步调用，调试通过后再恢复成异步调用。

使用 Local 驱动程序是调试分布式 PowerBuilder 应用程序的一个重要技术，程序中的绝大多数问题可用此手段发现并解决。

3.2 单机多进程调试

单机单进程调试通过后并不能保证实际的连接和通信没有问题，况且在某些情况下使用 Local 程序无法将客户机和服务器统一在同一个进程中。这种情况在开发多层应用调用第三方产品时就可能发生，特别在开发 WEB 应用程序时更是如此。因此有必要进行单机多进程调试。单机多进程调试的方法是将客户和服务器分开为不同的进程，以测试它们的之间的通信问题。此时采用的主要手段是使用本地驱动程序——LocalHost，告诉 PowerBuilder 应用程序在本地机上，而不去进行实际的外部网络通信。单机多进程调试时应注意以下几个问题：

(1) 由于服务器和客户机将分别作为独立进程运行，因此必须选择既支持服务器又支持客户机的通信协议和操作平台。在 Windows NT 和 PowerBuilder UNIX 平台下可选用 WinSock 或 NamedPipes 通信协议，在 Windows 95 平台下可选用 WinSock 通信协议。Win-

dows3.x平台本质上是单任务的，因此不能用来进行此种调试。其它的操作平台如：AIX，HP-UINX，UINX-Solaris可选用WinSock通信协议；

（2）当选用WinSock通信协议时，将客户端的连接对象的Location属性设置成"LocalHost"，而服务器端的Transport对象的Location属性可以忽略；当选用NamedPipes通信协议时，将客户端的连接对象和服务器端的Transport对象的Location属性不设，让PowerBuilder创建一个本地命名管道；

（3）当选用WinSock通信协议并将Location属性设置成"LocalHost"时，应为本地计算机LocalHost指定一个回送地址，按惯例其IP地址为127.0.0.1。其方法是在主机文件中增加一行：

127.0.0.1 localhost

若客户机和服务器端的Application属性采用服务名，还应在服务文件中增加一行：

appname port/tcp

其中appname为设置的服务应用名，port为指定的端口号（应大于4096小于65536）；

（4）当进行单机多进程调试时，为了判断脚本的运行情况，可在脚本的适当位置放置模态窗口或MassageBox（）函数，这是唯一可使后台程序挂起的方法，也是进行多进程调试的关键性技术。

3.3 多机远程调试

当单机多进程调试通过以后，若网络通信正常，一般可保证分布式PowerBuilder应用程序的正常工作，否则要进行多机远程调试。远程调试时可使用客户连接对象或服务器传输对象的Trace属性进行监控，Trace属性的设置是通过给一些跟踪选项关键字赋值来实现的。下面是踪选项关键字的赋值说明：

（1）All=1记录所有客户和服务器之间的活动，该选项记录对调试有用的内部信息，包括内存使用情况、内部调用跟踪以及传递的参数的值和类型；

（2）Console=1将所有活动记录到控制台窗口（Windows 3.1不支持）；

（3）DumpArgs=1记录参数的值和类型；

（4）Level=1使Console=1，ObjectCalls=1，和ObjectLife=1选项有效；

（5）Log=filename将所有活动记录到日志文件。如果客户端或服务器使用多个连接或传输对象，就应该为每个连接或传输对象分别指定一个日志文件；

（6）ObjectCalls=1记录每一个对象方法的调用以及成功与否；

（7）ObjectLife=1记录每次创建或删除对象的操作以及成功与否；

（8）对服务器来说，还有另一个选项：WebPB=1记录服务器应用Web.PB的所有的活动。

4 结语

调试分布式PowerBuilder应用可能很复杂，但使用本文介绍的先单机单进程调试，进而单机多进程调试，最后再多机远程调试的调试策略，将使其变得比较容易。

参考文献

1 Simon Gallagher, Simon Herbert. PowerBulder 6.0 Unleashed. Sams Publishing, 1998.

2 William B, Heys. Special Edition Using PowerBuiler 6. Que Corporation, 1998.

3 王 蓉等. PowerBuiler应用开发技术详解. 北京：电子工业出版社，1999.

（责任编辑：黎贞崇）