计算机科学2002Vol.29№.12

一种新型的Web应用解决方案——Bullant*)

Bullant:A New Web Application Resolution

赵文兵　杜孝平　谢昆青

(北京大学信息科学中心　视觉与听觉信息处理国家重点实验室　北京100871)

Abstract　In this paper, we introduce a new resolution--Bullant to develop Web application, which represents user interfaces and business logic on the server while the clients just display the interfaces and accept users' inputs. The Web server can keep the data as persistent objects in no demand of external database, support types of devices such as PC, PDA, mobile phone etc. in wired/wireless mode through the lowest bandwidth of 9.6K/s and is linear scalable with Zero Friction Engine. It's a thoroughly new resolution compared with the others.

Keywords　Bullant, Web application, Web server

1　引言

伴随着互联网在全球的延伸,越来越多的企业希望把自己的业务拓展到Web上,能够随时随地地为广大客户提供服务。为了满足企业开发Web应用的需求,IT业界提出了多种基于Client—Server模式的解决方案[1~4],基本结构如图1[5]所示。这些解决方案的设计思想大致是:对于用户界面和数据服务器,直接从当前多种互联网上的用户界面开发技术以及多种数据库管理系统中各自选取合适的技术与系统来实现,最后在整体方案中提供多种相应的支持接口。解决方案的核心部分在于Web应用服务器的设计与实现,用户通过用户界面根据应用服务器上定义的业务逻辑远程浏览更新数据库服务器中的数据。其中有代表性的IBM WebSphere的Web应用解决方案的体系结构如图2所示。从图中可以看出,采用上述设计思想的解决方案,Web应用的内部结构一般都比较复杂,用户界面、应用服务器与数据库服务器都是相对独立的部分,开发工具各不相同,这就无形中增加了数据在这几层之间流动过程中的转换负担,同时也影响了对Web应用中涉及的数据与服务在整体上的统一组织与管理。

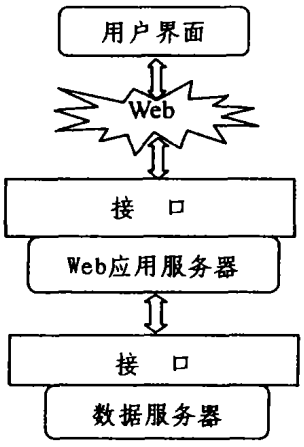

图1　基于Client—Server模式的Web应用结构

如果转换一下设计思路,即不是把当前用户界面与数据库管理系统的现有技术简单地集成到Web应用,而是直接在目前的操作系统和网络技术水平上开发完整的Web应用平台,会得到怎样的结果呢?在2001年8月发布的Bullant技术就是一种设计思路与众不同的新型Web应用解决方案:用户可以通过有线或者无线的方式,使用多种接入设备作为显示用户界面的对称远端与保存永久数据对象的Web应用服务器进行交互,从而实现Web应用。本文将在与现有的Web应用解决方案比较的基础上,介绍Bullant技术。

2　Bullant解决方案的体系结构

目前,硬件的性能越来越高,被普遍采用的服务器配置已经可以达到数千兆甚至数十千兆的超大内存(Very Large Memory),64位的寻址空间以及大型的对称多道处理系统(Symmetric Multi processing),等等。然而,现有的关于网络服务器的软件开发却大大滞后于硬件的发展。为了与这些硬件性能相适应,Bullant[6]提供了一种全新的Web应用解决方案。其体系结构如图3所示。Bullant采用传统的"胖服务器-瘦客户端"的结构,其中的网络服务器全面负责管理客户端用户界面的显示和对Web应用业务逻辑的控制,客户端通过专有的协议RAP与网络服务器建立连接,显示交互界面,接受用户的输入。在Web应用的开发过程中,客户端没有编程任务,只需进行服务器端的编程,使用一种类Java的面向对象编程语言——Bullant语言。同时,Bullant网络服务器具有能够处理满足ACID工业标准事务的内存,Web应用中需要长期保留的数据可以保存在永久对象中,无需使用额外的数据库来管理应用中的数据,因而整个Web应用运行在统一的环境中,其内部结构简单清晰,与前面介绍的用户界面、网络服务器与数据库服务器分别运行在不同环境中的Web应用解决方案有着本质不同。如果采用Bullant的解决方案开发一个新的Web应用,则无需任何外部系统的辅助;如果开发的Web应用需要集成一些原有的系统,Bullant的网络服务器也提供了多种接口来集成原有的数据源或者应用系统。对Bullant解决方案中的ACID内存、对等远端和实现线性扩展性的零摩擦引擎等主要特征将在下一节加以介绍。

*)本文研究得到国家重点基础研究发展规划(973)(G1999032705)的资助。赵文兵　博士研究生,研究方向为Web应用,信息集成。杜孝平　博士后,研究方向为信息系统,数据挖掘,Web应用。谢昆青　博士,副教授,研究方向为地理信息系统,数据仓库,Web应用。

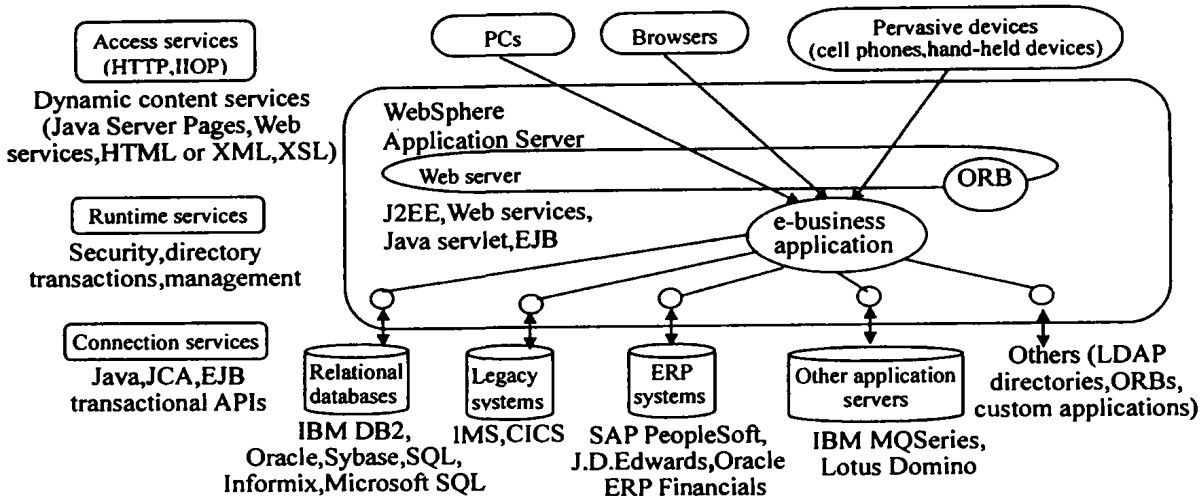

图2 IBM WebSphere 应用服务器4.0版所支持的 Web 服务的体系结构

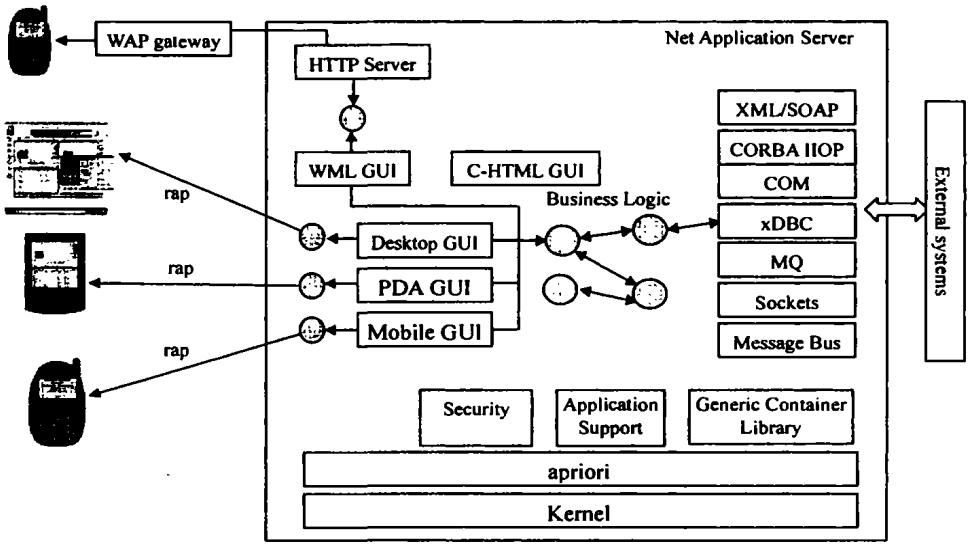

图3 Bullant 的 Web 应用解决方案的体系结构

3 Bullant 的主要特征

3.1 能够保存永久数据的 Web 服务器

Bullant 的 Web 服务器提供了一个可以让应用程序直接以事务方式存取数据的内存环境，所提供的事务机制完全满足关于事务处理的 ACID 工业标准。其中 A 代表原子性(Atomic)，即一个事务中的所有操作序列要么全部执行，要么全部取消；C 代表一致性(Consistent)，即事务必须保证系统中所有数据的一致性；I 代表隔离性(Isolated)，即每个事务的运行不会影响到任何一个其他事务；D 代表耐久性(Durable)，即当事务提交以后，数据的变化能在系统出错甚至崩溃的情况下保存下来。

和 Web 解决方案中常用的、与网络服务器分立的数据库服务器相比，Bullant 使用 ACID 内存来管理应用程序使用的数据，减少了数据在网络服务器与数据服务器之间的转换环节，节约了数据在这二者之间的传输时间。更为重要的是网络服务器上应用程序中涉及的数据以对象方式保存在本地，外界没有可以利用的接口获取这些数据，这就增强了数据的安全性。整个应用程序采用一种数据模型，解决了以往数据的应用模型与存储、管理模型之间的不一致性，使得应用程序的数据模型更为容易理解，也便于管理。

在 Bullant 服务器中，将应用程序使用的数据分为持久的和临时的两类来加以组织和管理。对于需要持久保存的数据，应用服务器是通过维护这些数据的可达性（Reachability）来实现的。应用服务器的 ACID 内存中有一个预先定义为“持久对象的根”(The Persistent Root)的初始化对象，如果需要把一个对象定义为持久对象，只需把它加入持久对象的根的列表即可；如果一个对象的祖先是持久对象，那么这个对象本身也将被持久化。所有需要持久化的对象以“持久对象的根”为起点形成了一张有向图。持久性作为可以传递的属性，赋给了持久对象关系图中的每一个对象。只有通过事务才能对持久对象进行修改。系统自动保证这些对象的持久性，即使是系统崩溃或者硬件断电都不会影响到这些对象在最后的事务提交以后的状态，只要网络服务器再度运行，这些对象都会恢复到最近的状态。对于只需要在应用程序的一次运行期间存在的对象，与对持久对象的管理类似，可以加入一个以预先定义为“全局对象的根”(The Global Root)的初始化对象为起点的有向图，系统通过对这张图的维护来管理临时对象。

Bullant 网络服务器内核中有一个名为 Epoch 的系统线程，用于专门负责保证持久对象的持久性。Epoch 会定时获取内存中所有持久对象的快照并把它们保存到物理磁盘上去。同时，内核中另有一个名为 Delta 的系统线程负责将在最后

一次 Epoch 线程执行之后发生的所有事务的执行记录存入物理磁盘。每个事务处理完毕后，Delta 线程都会立刻把它记录下来，不会丢失任何对持久数据的修改。这两个系统线程是在后台运行的，不会影响到应用程序的运行。当系统崩溃或者发生断电等故障之后，只要系统重新启动，Bullant 首先自动装入最近的 Epoch 存盘文件中的持久对象，然后执行这个 Epoch 存盘文件创建之后的所有 Delta 存盘文件中的事务记录。这样所有的持久对象就可以恢复到系统崩溃前的状态，从而保证了数据的完整性和一致性。

3.2 对等远端

对等远端是一种在网络服务器上运行应用程序、同时在客户端设备上显示相应用户界面、完成用户与服务器上的应用程序之间的交互的技术。只要在远端设备上安装小巧且免费的 Bullant Remote 软件，客户端设备不需要下载任何额外的代码或者插件，就可以根据名称连接所有正在 Bullant 服务器上运行的 Bullant 应用程序，显示相应的用户界面，而不需要额外下载任何 applet 或脚本，这就杜绝了运行不安全代码的可能性。Bullant Remote 是一种可以运行在多种设备上的瘦客户端软件，例如在手机上，Bullant Remote 只需要50k 的内存就可以顺利运行。同时，Remote 也支持以插件的方式在 Internet Explorer 或 Netscape 中运行。

Bullant 网络服务器与远端的客户设备之间是通过专有协议 RAP (Remote Application Protocol) 建立连接和保持同步的。RAP 是直接建立在 TCP/IP 上的，与其他网络协议相比，它大大减少了所需要的传输带宽，最小带宽只需要 9.6KB/s，因此，RAP 很适合手机、掌中宝等通过低速无线方式接入的设备。Bullant 网络服务器的内核直接使用 RAP 与远程的客户端进行连接，避免了传统解决方案中为每个网络用户创建连接进程的过程。RAP 可以与透明的、经过 X.509 认证的、有关安全传输的工业标准协议 TLS 1.0协同工作，用来提高传输安全性；也支持使用128位的 RSA 算法进行加密传输。这样，即使工作在开放的 Internet 上，RAP 也能够保证应用所需的安全性。

除了支持数以万计的并发用户和良好的安全性这些优点以外，对等远端另一个显著优势是对嵌入式设备和手持设备的支持。每一种可以显示用户界面的设备都有其特殊性，比如手机和 PDA，它们的显示区域比较小，界面元素比较简单，而 PC 的显示器就可以显示非常多的界面元素，比如文本框、按钮以及图标等等。针对每一种客户端设备，对等远端都加入了一些特殊的设计，使得每种设备都可以根据自己的显示能力，展示相应的用户界面。使用对等远端技术的直接结果是，网络服务器上运行的每一个 Bullant 应用程序的用户界面都可以在多种不同种类的硬件设备上显示，用户可以任选 PC、PDA 或者手机来接入同一个应用程序并与其交互。

3.3 线性可扩展性

与目前具有海量内存和多个 CPU 的网络服务器硬件体系结构相适应，Bullant 网络服务器的性能呈现线性的可扩展性，具体的数据如图4所示。对于 Web 应用供应商来说，应用系统的性能能够随着 CPU 数目的增加保持线性增长是一个非常重要的优势，因为这样能够在 Web 应用的初期开发阶段根据实际需要来合理配置服务器硬件，当网络客户增加以后只要增加服务器的 CPU 数目，就可满足众多客户对 Web 服务的性能要求，从而保护投资，运用有限的资金获取最大的利润。

Bullant 网络服务器的线性可扩展性是以服务器内核中的**零摩擦引擎**(Zero Friction Engine)为基础的。**零摩擦引擎**将所有运行程序中涉及的所有对象放在一个内存中管理，避免应用程序存取慢速的物理磁盘，从而提高应用程序的性能。对于多 CPU 的处理机而言，导致 CPU 时间浪费的有以下几个主要原因：首先是内存回收，现有的编程环境大多具有自动的内存回收机制，用来防止内存或资源在使用后没有及时回收这种致命的编程错误，然而内存回收机制是以占用大量 CPU 时间和显著影响系统性能为代价的；其次是共享资源的并发和加锁控制。一个系统中具有多种共享资源，例如 CPU 的使用权、I/O 操作权、特定内存空间的分配等等，在多任务的情况下，尤其是在用户或任务数目巨大的情况下，对于这些共享资源的管理会占用相当数量的 CPU 资源；最后是并发线程运行环境的交换，当一个系统中有数以千计的线程并发运行时，即使采用线程池(thread pooling) 技术，也会有大量的 CPU 时间用在并发线程运行环境的切换上。

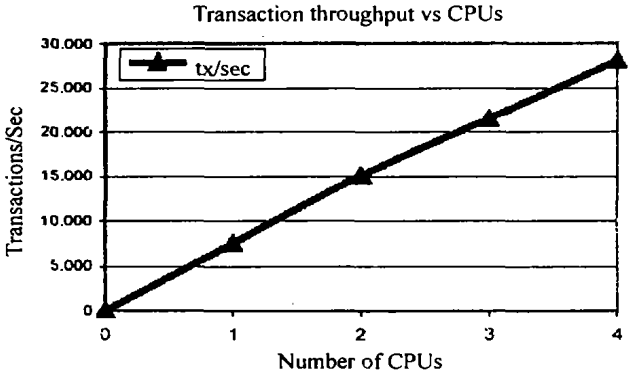

图4 Pentium Ⅲ处理器，内存为2G 时，事务流与 CPU 数目的线性关系

零摩擦引擎针对上述问题采取了不同的解决措施：首先，Bullant 的应用程序会自动进行内存回收。服务器内核中有一个名为 Cleaner 的系统线程负责内存的回收。每个应用程序在运行时主动通知 Cleaner 线程自身引用对象的情况，Cleaner 会及时销毁不被任何程序引用的对象，这样就可以在不影响程序正常运行的情况下进行并发的内存回收，从而保证系统的性能。

其次是对于每一种系统资源都由一个相应的系统线程——System Agent 负责管理，另外有一个名为 Governor 的系统线程与这些 System Agent 相互作用，把各种系统资源分配给具体的任务。使用缓冲技术，Governor 可以高效地进行资源分配，不需要 System Agent 之间保持同步，也几乎不需要对这些 System Agent 进行管理，充分发挥了所有 CPU 的计算能力，使得系统便于扩展，承担更大的任务负载。

最后，零摩擦引擎使用任务-线程映射的技术，将多个用户任务映射为系统中的一个线程，从而有效地减少了系统中实际发生的线程数目，节约了在并发线程运行环境切换上的 CPU 时间。根据实测报告，这种技术可以增加20%的 CPU 可用时间，而不会影响任务的正常运行。

4 对 Bullant 解决方案的评价

与其它解决方案相比，采用 Bullant 解决方案开发 Web 应用具有以下几方面优点：

Web 应用中需要长期保存的数据由 Bullant 网络服务器的 ACID 内存来管理，无需使用外部的数据库系统来管理应用程序的数据，所以在开发过程中，数据模型融合在程序模型

中，无需另外设计数据存储模型。在 Bullant 中，客户端的用户界面是在网络服务器上使用 GUI 类库创建的，无需为客户端的界面显示专门编写程序。因此，使用 Bullant 的 Web 应用开发是在网络服务器上一体化的环境中完成的，简化了应用内部结构的层次，降低了系统设计的复杂度。

在 Web 应用开发的初期，由于 Bullant 网络服务器中需要使用 ACID 内存，该服务器的硬件配置要求优于其它解决方案中的服务器，然而也是因为使用 ACID 内存，无需购买有关数据库管理系统的软件，可以显著降低在软件方面的投资。Bullant 网络服务器具有线性可扩展性，当 Web 应用的用户数目上升时，可以通过单纯增加服务器中 CPU 的数目以及扩充内存来提高 Web 服务的性能，无需对应用程序做任何改动。在其它解决方案中，往往是通过增加服务器的数目来提高 Web 服务的性能，在多服务器的情况下，产生了服务器之间的通讯连接、任务在各个服务器之间的分配等问题。为了维持系统运行的连贯性，通常需要对应用程序进行修改，不仅要在硬件方面追加投资，还要承担软件维护的费用。因此，从系统的整体运营成本方面考虑，Bullant 是一种更为经济的解决方案，可以最大限度地保护投资。但是，在系统开发之初必须考虑服务器配置的可扩展性，比如采用单 CPU 时，需要上千兆的内存，并且服务器需要采用能够添加 CPU 的体系结构，所以 Bullant 解决方案的初期硬件投资比较高，这可能是一个遗憾。

在 Bullant 解决方案中，客户端与网络服务器进行通讯所需带宽最小仅为9.6K，接入方式既可以是有线的，也可以是无线的，支持 PC、PDA、手机等多种接入设备。对于同一个应用，网络服务器根据不同接入设备的图形显示能力定义不同的用户界面的表现形式，用户可以利用多种接入设备作为客户端，随时随地与 Bullant 提供的 Web 应用进行交互。因此，采用 Bullant 解决方案的 Web 应用具有良好的可用性。

Bullant 应用中需要长期保留的数据保存在网络服务器的 Epoch 文件中，这些 Epoch 文件仅供系统使用，服务器中不存在数据库管理系统中那样的对外部开放的接口，所以这些数据无法被恶意窃取，即使攻击服务器得到了 Epoch 文件，其中的数据也是按照内部的格式组织的，无法解读。客户端在显示 Web 应用的用户界面时无需下载任何 Applet 或者脚本，从而避免了运行不安全代码的可能。同时，Bullant 服务器与客户端之间的专用协议可以与安全传输的标准协议 TLS 1.0协同工作，也支持使用128位的 RSA 算法进行加密传输。因此，Bullant 解决方案具有相当可靠的安全性。

结束语 本文介绍了一种新型的 Web 应用解决方案——Bullant，并将它同现有的其它解决方案进行了比较分析。Bullant 的 Web 应用解决方案中的 ACID 内存、支持多种接入方式与接入设备的对等远端以及建立在零摩擦引擎基础上的线性可扩展性，为 Web 应用的开发提供了一些新颖的思路。虽然开发中使用 Bullant 语言有别于目前流行的 C、C^{++} 或者 JAVA 等语言，但它是一种完全面向对象的程序设计语言，与 C^{++}，尤其是 JAVA 的编程方法极为相似，熟悉 JAVA 的程序设计人员甚至可以立即使用 Bullant 语言进行开发。采用 Bullant 的解决方案可以将用户界面、业务逻辑与永久数据相结合，对 Web 应用进行一体化描述，大大降低了应用内部层次的复杂度，使系统具有良好的扩展性。笔者所在的北京大学信息科学中心使用 Bullant 技术已经成功地开发了一个网上股票交易系统原型。相信 Bullant 技术会为 Web 应用的构建提供一个简洁、高效的解决方案。

参 考 文 献

1 Chong J. Real Business Processing Meets the Web, SIGMOD'98, Seattle, WA. USA, 1998.536

2 IBM Corporation. Integrating Data and Transactions for Agile e-Business, Aug. 2001. http://www-4.ibm.com/software/web-servers/appserv/whitepapers/wp_was_overview.pdf.

3 Microsoft Corporation. Introducing the Windows .NET Server Family, Nov. 2001. http://www.microsoft.com/windows.net-server/evaluation/overview/default.asp.

4 Teta srl Internet Consultant, StWeb Stratos Web and Application Server, Jan. 2002. http://www.stweb.org.

5 Serain D. Middleware, Springer, 1999

6 Bullant Technology Pty Ltd. CAN 069011114, bullant Technology White Paper, Jan. 2000

（上接第105页）

我们用四台采集器构成的机群分别用 HTTrack、Wget 以及我们改进后的 New_wget，在保证大致相同网络带宽及负载情况下，对本地局域网以及 www.sina.com.cn、www.peopledaily.com.cn、www.redhat.com 等一些国内外著名网站进行下载，并用 New_wget 对不同网站进行下载，将整个过程进行详细记录。图3是进行横向和纵向综合分析比较后得出的下载效率及内存占用情况分析结果。

从图中可以看出，由于采用新的策略，New_wget 在下载效率方面比 Wget 提高了近五倍，而在内存占用方面约是 HTTrack 的十分之一。

然而，New_wget 下载引擎在做 DNS 查询时效率较差，需要多次网络查询，较费时间，而且系统调用 gethostbyname()时速度较慢。鉴于此，下一阶段，我们将需要在相应部分做进一步改进，使用 pthread 机制，优化 Cache 机制，使下载引擎能发挥出最大优势。

参 考 文 献

1 http://www.LinuxAid.com.cn;

2 http://www.redflag-linux.com;

3 http://www.httrack.com;

4 http://www.gnu.org;

5 Stevens W R. UNIX 环境高级编程. 机械工业出版社，2000

6 怀石工作室. Linux 上的 C 编程. 中国电力出版社，2000.

7 邹海明，余祥宣. 计算机算法基础. 华中理工大学出版社，1995