为什么使用 EJB 组件？

Why EJB Components are Used

郭琳 （北京中医药大学信息中心 100000） 李环 （首都师范大学信息工程学院 100000）

1 什么是 EJB 组件？EJB 组件是为企业级应用设计的 java 组件模型

EJB 组件是基于标准分布式对象技术、CORBA 和 RMI 的服务器端 java 组件。EJB 组件总是分布式的，这是它们与标准 JavaBeans 组件最根本的区别。

EJB组件提供了应用的商务逻辑部分。由于它们不涉及表示层的问题，因此必须与其他的显示技术(如 servlets)，服务于 HTML 客户端的 JSP 技术，或者使用了诸如 AWT、Swing 技术的 java 应用一起使用。

实现了 EJB 规范的应用服务器提供了可以解决安全性、资源共享、持续运行、并行处理、事务完整性等复杂问题的服务，从而简化了商业应用系统。

Sun 公司制定的 EJB 组件模型要求 EJB 组件运行于 EJB 服务器(通常称为应用服务器)的环境下。我们的示例中使用了高级版 WebSphere 应用服务器，但所讨论的功能适用于大多数 EJB 服务器。

2 需要考虑的技术问题

当你在决定 EJB 组件是否适合实际情况的合适技术时，不妨先考虑几个问题。如果你对所有这些问题的回答都是肯定的，那么 EJB 组件就是你可以采用的合适技术。反之，别的技术可能更适合。

2.1 你需要将商务逻辑组件与面向外界Internet隔离开吗？

许多公司认为他们的应用软件，特别是构成商务逻辑的一些标准和数据结构，是极为重要的公司财产(例如，公司所拥有的分析应用工具构成了股票交易网站的一部分)。允许外人访问这些属于公司资产的原码和目标码将对公司产生极大的危害。因此，这些公司十分需要将商务逻辑置于一套安全防火墙后面(通常称为无戒备区，也称 DMZ)。

在这种情况下，分布式对象组件体系结构(例如 EJB 技术)允许你将有价值的公司资产隔离到 DMZ 以内，同时表示层代码可以访问 DMZ 内的 EJB 服务器。图 1 描述了这种分布式解决方案：

2.2 防火墙内部的 EJB(见图 1)

这里我们假设表示层逻辑不如后台的商务逻辑重要。如果不是这样，那么这种方案的安全性就要下降，整个系统可能都需要置于 DMZ 之内。如果整个应用必须(或者能够)置于第二层防火墙后面，那么选择其他技术(如通过 Java Servlets 发出 JDBC 请求来直接访问数据库)就显得更合理。

这种解决方案也有一些效率方面的缺陷。例如在安装 WebSphere 时，如果你将客户端(servlets 与 JSP 文件)与图示的位于另一个 java 虚拟机(JVM)中的 EJB 组件分隔开，这种选择将降低整体性能。与客户端和 EJB 组件位于同一个 JVM 中的情况相比，这种方式下每一个需要经过防火墙的请求都将增加 20% 的耗时。

2.3 你需要不止一种类型的客户端访问共享数据吗？

通常，一个应用会有多种类型，需要访问相同信息的客户端。例如，一个应用可能会有供外部客户访问的基于 web 的 HTML 前端，以及供内部人员使用的更完整的应用前端。通常，这个问题是通过为同一应用编写两个共享相同数据源(数据库表)的版本来解决的。但是，这种方法效率不高，无论是从编程时间还是从同时发生多个数据库锁定时数据库的利用率来说，EJB 技术的解决方案是将共享数据和商务逻辑集成到一套 EJB 组件中，以供不同类型的客户端(如 servlet/HTML 和 application)访问。EJB 组件控制着对后台数据的访问，并管理着当前事务以及数据库的内部锁定。通过去除应用中的重复代码，减少编写数据库控制逻辑的工作，这种方案降低了总的编程量。

在这个领域还有其他一些解决方案--比如，java 应用可以通过 HTTP 访问 java servlets，同时浏览器也可以通过 HTTP 访问 java servlets。这些解决方案的问题在于：如果 servlet 是用来在浏览器中显示信息的，它就必须包含一些表示层逻辑，这些表示层逻辑对于向另一个程序传递信息来说是多余的。因此，最终不得不采用两套部分重复的 servlets 来处理两种情况。此外，HTTP 不是程序间通信的效率最高的协议，你必须设计能通过 HTTP 管道进行程序间信息传递

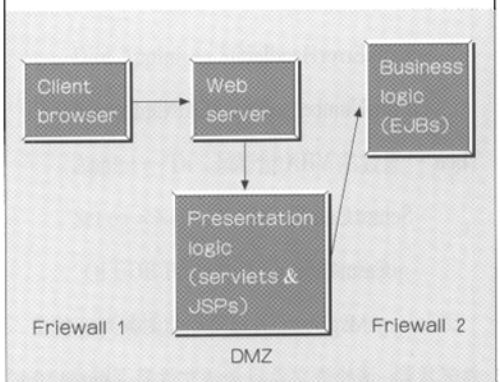

图 1 分布式解决方案

的数据格式--这通常或者是基于文本的格式，比如 XML（由接收端进行解析，由发送端生成），或者是基于对象的格式，比如 java 序列化。两种格式都需要大量的编程工作，它们都不如本地的 IIOP 速度快。

2.4 你是否需要对共享数据同时进行读和写操作？

通常，"胖客户端"解决方案要求应用在数据库级别上管理对共享数据的访问，其结果是：处理数据库锁定与同步的方案非常复杂，若不考虑数据库锁定与同步问题又会失去数据的完整性。

EJB 技术能自动处理这些复杂的共享数据同步问题。正如前面提到的那样，EJB 组件控制着对后台数据的访问，并管理着当前事务和数据库的内部锁定。这不仅省去了编写数据库控制逻辑的工作量，同时也保证了数据的一致性与正确性，从而降低了总的编程量。

2.5 你需要访问具备事务处理功能的多个异构数据源吗？

许多应用需要访问多个数据源。例如，一个应用程序可能既要访问来自中间层的 Oracle 数据库，又要通过中间件(如 MQSeries)访问 CICS、IMS 等大型主机系统。问题的关键是一些应用要求这种访问是完全事务化的，并且数据完整性在不同数据源间也能得到保证。例如，某个应用可能要求在处理用户的订购信息时，既要在 Oracle 数据库中存储详细的订购信息，同时又通过 MQSeries 在 CICS 系统中存储一份出货订单。无论是数据库更新或是 MQ 队列产生错误，整个事务都应被取消。

过去，构建这种系统的唯一选择是采用事务监视器，例如 Encina、CICS、Tuxedo，它们使用非标准接口并需要用 COBOL、C、C++ 等语言进行开发。例如，高级版 WebSphere 中的 EJB 容器支持多个并发的事务，具备在多个 DB2 数据源间进行完整的事务提交及事务取消的能力，这些都是在一个完全支持二状态事务提交的环境中进行的。目前，WebSphere 对其他数据源(如 Oracle, MQSeries 和 CICS)只支持单状态的事务提交。企业版 WebSphere 中的 EJB 容器能对更多的数据源支持二状态的事务提交。

大多数容器正在支持各种数据源下的二状态事务提交方面不断完善。随着时间的推移，我们将看到这一领域的不断进步。

2.6 你需要能与 HTML 文档、servlets、JSP 文件、客户端登录安全性无缝集成的方法级对象安全性吗？

某些类型的应用由于其安全性限制，使得以前它们很难通过 java 应用来实现。例如，某些保险业的应用程序为了满足管理规定的要求，必须限制对客户数据的访问。直到 EJB 技术出现后，才能够限制特定用户访问某个对象或方法。在这之前，可实施的办法只有：在数据库级别上限制访问，并捕获在 JDBC 层次上抛出的错误；或者通过客户安全密码在应用层上限制访问。

EJB 技术可以在任何 EJB 组件或方法上实施方法级的安全策略。创建的用户和用户组可以被授予或禁止对任何 EJB 组件或方法的操作权。在 WebSphere 中，用户组可以被授予或禁止对 web 资源(servlets, JSP 文件和 HTML 页面)的访问权，用户的 ID 可以通过底层的安全机制被安全地从 web 资源传递到 EJB 组件。

2.7 体系结构是否有标准化、轻量化、组件化的需要呢？

对于许多有远见的公司，关键问题是要实现平台、销售商和应用服务器设备间的相互独立。符合工业标准的 EJB 组件体系结构有助于实现这些目标，为 WebSphere 开发的 EJB 组件通常可以发布到其他类型的应用服务器上使用，反之。尽管这一目标尚未完全实现，但它已成为许多客户选择的战略发展方向。从短期看，利用一些可能优于标准化的特性会更方便、迅速，但从长远看标准化具有最大的好处。

当考虑到越来越多的可选工具和 EJB 标准的优化实现手段时，这些都是你无法从本地管理对象框架中获得的。由于大多数公司并不从事中间件业务，将注意力集中在与你的业务更直接相关的活动上会更有效。

2.8 你需要多个服务器来满足系统的吞吐量和有效性需要吗？

胖客户端系统显然不能适应 web 系统可能拥有的成千上万个用户，软件发布方面的问题也要求给胖客户端减肥。Web 站点的 24 小时不间断运行特点也使得时间成为关键问题。但并不是每个人都需要 24 小时不间断运行，并能同时处理上万个用户的系统。你应当能设计这样的系统：在不增加开发和标准化难度的前提下，实现系统的伸缩性。

3 一个使用 EJB 技术失败的例子(见图 2)

作为一个被授权评估新技术的高级技术小组的成员，某个方案小组最初设计了这个方案。该方案具有如下体系结构(每个方框代表运行于各自硬件上的不同程序)。

不可取的 EJB 体系结构见图 2。

该体系结构的一些具体设计使得 EJB 组件的使用不如它初看起来那么有吸引力：

错误！

应用的主体部分是信息的显示，这部分由 java servlets 实现。EJB 组件只用来获取、更新数据。

方案小组在后台主机的事务处理中使用的

图 2 EJB 体系结构

数据连接不包括遵从扩充体系结构(XA)标准的数据源。事务不能被成批取消或提交，对每个主机的访问都是一个独立的请求。

由于不能区分用户是来自 web 还是来自后台主机，EJB 技术的安全特性在此没有得到利用。

servlet 对 EJB 组件的每一次访问都是一个网络请求。组件的探测方法和 ejbLoad() 方法的内部逻辑执行了真正的主机请求，但这之后 EJB 组件仅仅进行数据缓存，直到事务提交后才由 ejbStore() 方法将信息回传给主机。直到事务提交前的每一次数据访问和更新都导致了大量的网络开销。

该方案小组仅仅把 EJB 组件用作主机数据与 servlets 间的数据映射机制，这不是 EJB 技术的有效使用方法。本例中的应用没有利用以下一些特性：

错误!

- EJB 组件的事务处理特性
- EJB 组件的方法级安全性
- EJB 组件的伸缩性与分布式特点(该方案小组仅使用了一个 EJB 服务器)
- 多种类型客户端间的商务逻辑共享
- 具备容器管理持久性(CMP)的 EJB 组件的自动数据映射功能

如果直接由和 java servlets 处于同一 JVM 中的 JavaBeans 组件来实现数据映射功能，系统的速度将比使用 EJB 组件快得多，同时避免了大量的网络开销。该系统还变得不必要的复杂(因为需要本地和远程接口，以及分布代码)。事实上，该方案后来被废弃并以不使用 EJB 组件的方式进行了重新设计，其中的数据映射由一套被 servlets 使用的标准 JavaBeans 组件实现。这使得最后的系统变得更简单、快速。

4 一个成功使用 EJB 技术的例子

一个金融机构的方案小组设计了这套方案，最初使用了 java 和 RMI 技术。其系统体系结构如下：

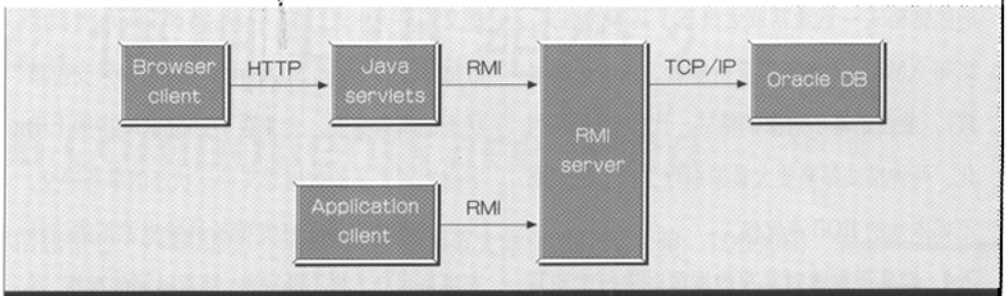

图3 RMI 服务的体系结构

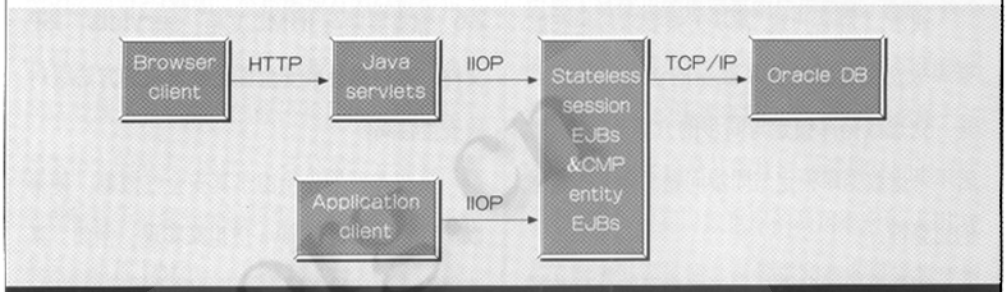

图4 新的 EJB 体系结构

4.1 RMI 服务器体系结构(见图 3)

除了多种类型的客户端，方案小组还在系统内构建了以下两个框架单元，在本图中未画出来。

一个数据库映射层，通过 RMI 服务器上的 JDBC 将其 Oracle 数据表映射成 java 类，或者反过来。

一个用户安全框架单元，用于验证来自浏览器或应用客户端的用户，并进而决定用户是否有权进行 RMI 和数据库请求。

由于他们的应用需求与 EJB 技术的目标紧密相连，该方案小组得以很快将其设计方案转变为以下体系结构：

4.2 新的 EJB 体系结构(见图 4)

通过使用容器管理持久性(CMP)，EJB 组件使得方案小组可以抛弃数据库映射框架，转而利用 WebSphere 和 VisualAge for Java 中完整高效的实现机制。通过利用 EJB 技术的安全性，方案小组得以在应用中保留方法级安全性的同时，抛弃了他们的用户安全框架。结果，由于需要维护的代码量减少了，他们的应用变得更为简单。

简单地说，该应用的以下特点使得它适合使用 EJB 技术：

- 具有共享相同商务逻辑的多种类型客户端
- 使用了事务处理
- 需要通过 CMP 组件来实现与对象有关的映射
- 需要方法级的安全性

由于系统的设计目标是通过使用与实体同时发布的会话级 EJB 组件来实现最小的网络流量，该系统最终的体系结构很好地使用了 EJB 组件。对于前一种体系结构，由于客户端需要发出大量的请求，很可能会增加网络负担，另外由于要求客户端进行事务的启动、提交、取消，使得系统复杂化。

5 总结

构建一个可能使用 EJB 组件的新系统时，做出正确决定的第一步是要懂得如何确定 EJB 技术是否适合于该应用，包括选择 EJB 组件作为一种实现手段所带来的正面和负面影响。

参考文献

1 USing VisualAge for Java Enterprise Version to Develop CORBA and EJB application

2 Design and implement Servlets, JSPs and EJBs for IBM Websphere Application Sever