J2EE 应用中以消息驱动 Bean 实现并发处理

The Implementation of Concurrent Processing with Message－Drive Bean in the J2EE Applications

陈智罡　（新疆喀什师范学院计算机科学系 844006）

摘要：在 J2EE 构架下的应用中添加并发性是受到严格限制的。然而许多应用为了保证可接受的响应时间又需要并发实现。本文详细介绍了如何通过消息驱动 Bean 有效地为 J2EE 应用提供并发性。

关键词：消息驱动 Bean J2EE　EJB　并发处理

1　概述

在许多实际应用中，并发处理是非常必要的，因为它可以提高程序的吞吐量、执行速度以及响应能力。在 JAVA 中可以很容易的通过多线程实现并发性，但在 J2EE 构架下，要想在具体应用中实现并发处理是受到严格限制的。这主要是因为 EJB 规范中严格限制了在 EJB 容器中创建新的用户线程，而且只能同步调用会话 Bean 上的方法[1]。这些都使得在 J2EE 构架下的许多应用无法直接利用并发处理，因此在对客户端的响应时间上就得不到保证。SUN 公司在 EJB2.0 中提出了一个新的类型的 Bean：消息驱动 Bean。消息驱动 Bean 允许 J2EE 应用可以异步处理消息，利用消息驱动 Bean 的异步性可以有效地为 J2EE 应用提供并发处理能力。

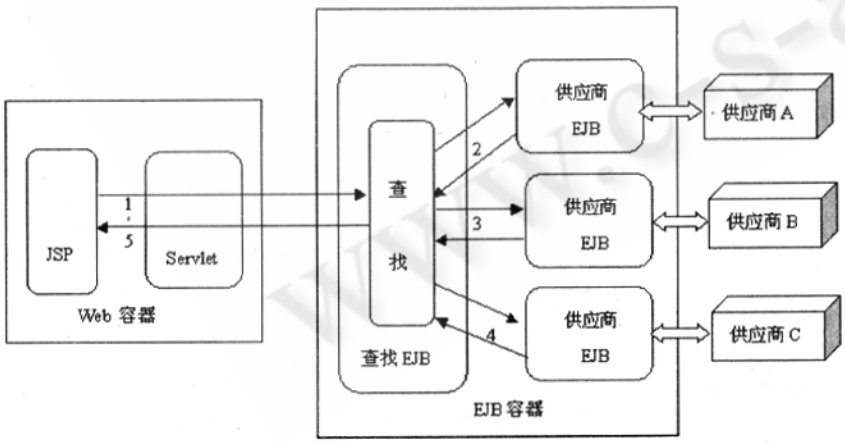

图 1　J2EE 架构下案例的典型实现

2　J2EE 架构中实现并发处理的可行性分析

本文以一个实例说明实现并发性的必要性和可行性。该实例要实现在多个供应商之间查询某个商品的价格和可供给的商品数量。用户只需要在客户端输入商品的名称及型号，就会调用一个后端处理去在多个供应商系统中搜索商品的价格及可供给数量，然后格式化查找到的数据，最后将数据返回到客户端。J2EE 架构下案例的典型实现如图 1。

该应用中查找 EJB 组件先后依次调用了三个供应商 EJB 组件，以获得三个供应商提供的商品信息。这里是以串行方式搜索供应商，如果从一个供应商中搜索商品信息的时间为 15 秒，由于是串行方式搜索的，则搜索三个供应商的最小时间为 45 秒，也就是客户端至少要经过 45 秒才能接收到返回的数据。由此可见，从用户提出请求到响应时间段内，大部分时间花费在多个供应商系统之间搜索信息上。尽管上述 J2EE 应用解决方案基本上是正确地，但这种解决方式有一个严重的设计缺陷，即响应时间随着供应商数量的增加呈线性增长趋势，若有 20 个供应商则至少要花费 300 秒时间。为了保证用户可接受的响应时间和能搜索更多的供应商，该应用的关键是必须以并发处理方式实现搜索。

并发处理的设计方案可以有如下三种方式：

（1）可以在宿主 servlets 的 JAVA 虚拟机上产生多个线程，每个工作线程直接调用一个供应商 EJB 组件而且每个工作线程只和一个供应商 EJB 组件相对应。当用户提交请求后 servlet 将输入请求分解成多个任务，每个任务负责与一个供应商打交道，这些任务都以线程的方式运行，这就使得搜索多个供应商可以通过多线程并行处理。这个方案看起来似乎可以很好的达到预期效果，但这种设计方式违背了 J2EE 设计思想，使得开发者的

精力都集中于多线程的复杂处理上，而不是致力于商务逻辑的开发上。

(2) 还可将查找 EJB 组件设计成能够产生多线程的方式，以达到可以并行调用供应商 EJB 组件的目的。但是 EJB 规范又限制在 EJB 容器内创建用户线程，所以这一方式并不可行。这一限制的主要原因是 J2EE 架构要建立一个高度可伸缩的服务器端组件，同时负责提供并行性和其他服务，该架构去除了企业 Bean 开发者在复杂多线程处理上的负担，而且 EJB 容器自己控制线程的产生能更有效的管理资源的分配。值得注意的是 EJB 服务器本身是多线程的，而且能并行处理多个客户请求。

(3) 可将供应商 EJB 组件实现为消息驱动 Bean 类型，这就使得查找 EJB 组件可以通过发送多个消息将搜索任务同时指派给多个供应商消息驱动 Bean，保证了供应商消息驱动 Bean 能够并行处理请求。这里利用了消息驱动 Bean 通过消息事件被异步调用的特性，从而较好的实现了并发处理请求。会话 Bean 却不能通过这种方式使用，是由于会话 Bean 被限制为只能同步且串行调用。

通过上述三个解决方案可以看出，使用消息驱动 Bean 实现并发处理的第三个方案是最佳解决方案。消息可以为异构系统的组件之间提供异步通信方式，在 J2EE 中由于 JMS 和消息驱动 Bean 的引入，使得消息服务能够很好的整合在 EJB 构架中。

3　使用消息驱动 Bean 实现并发处理

EJB 2.0 规范中定义了一种新的 EJB 类型，即消息驱动 Bean(Message－Driven Bean，简称 MDB)，它以 EJB 的形式实现 JMS 消息的接收者。消息驱动 Bean 实现了一组新的接口，这组接口使得 EJB 能够异步地接收和处理 JMS 消息生产者发送到队列或话题的消息[2]。如图 2 所示。

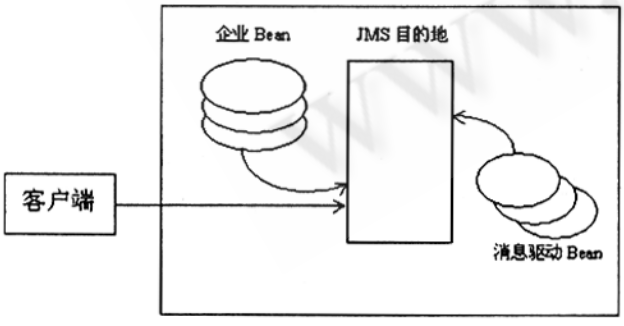

图 2　消息驱动 Bean 工作方式

EJB 客户程序的构造方式与普通 JMS 消息生产者的构造方式完全一样，也就是说，JMS 消息生产者不必知道消息的消费者是一个 EJB。相对于会话 Bean 和实体 Bean 而言，消息驱动的 Bean 最大的特点是客户程序不通过接口访问 Bean。与会话 Bean 和实体 Bean 不同，消息驱动的 Bean 只有一个 Bean 类。从某些方面看，消息驱动的 Bean 类似于无状态会话 Bean。一个消息驱动 Bean 的所有的实例都是等价的，这使得容器能够把消息指派给任意一个消息驱动 Bean 的实例。容器能够建立消息驱动 Bean 的缓冲池，实现消息的并发处理。一个消息驱动 Bean 能够处理来自多个客户程序的消息。会话 Bean 和实体 Bean 能够发送 JMS 消息，能够同步接收消息，但不能异步接收。一些时候，为防止过多地占用服务器资源，在服务器端的组件中想要避免阻塞，这时可以用消息驱动 Bean 异步接收消息。

上述实例利用消息驱动 Bean 的异步特性，将供应商 EJB 组件实现为消息驱动 Bean 类型，即可达到并发处理请求及提高响应时间的目的，如图 3 所示。

用户在客户端提供商品的名称及型号，然后 servlet 将依次调用查找 EJB 组件上的方法，将输入请求构造成为三个不同的 JMS 消息放入到请求队列中，每一个 JMS 消息与一个供应商系统对应，此后查找 EJB 组件进入等待状态，等待响应消息到达响应队列。进入请求队列中的消息触发了供应商消息驱动 Bean 的调用，供应商消息驱动 Bean 同时开始处理各自所对应的消息。在供应商系统中搜索到商品价格和供给数量之后，供应商消息驱动 Bean 将结果嵌入到 JMS 消息中并将其放入到响应队列中。这里所有的供应商消息驱动 Bean 是并发处理的，因此全部结果的返回大约为 15 秒，相当于搜索一个供应商的最小时间。然后处于等待状态的查找 EJB 组件从响应队列中取出所有消息装配成结果，返回到 Servlet /JSP 中。最后 Servlet /JSP 会将数据依次呈现给用户。在这个实现中响应时间取决于每个供应商响应的时间，而不是依赖于供应商的数量。即使供应商增加到 10 个，最小响应时间仍然大约为 15 秒。

在这个并发处理设计中，还应该考虑到当多个用户同时提出请求时，会有多个查找 EJB 组件的实例被激活，每一个实例服务于一个 Servlet 线程。这时应该有一种机制能将每个查找 EJB 组件的实例和与之相对应的响应消息关联起来。可以采取以下方式：

(1) 为每个请求建立一个临时的响应队列，这就使得不同请求的响应消息之间互不干扰。但是这种方式所带来的资源开销应该值得重视。

(2) 另一种方式是预先建立一个响应队列池，它将所有的响应消息共用一个响应队列池，而不是为每一个请求单独建立一个响应队列。这种情况下需要一些工具类去管理响应队列池。

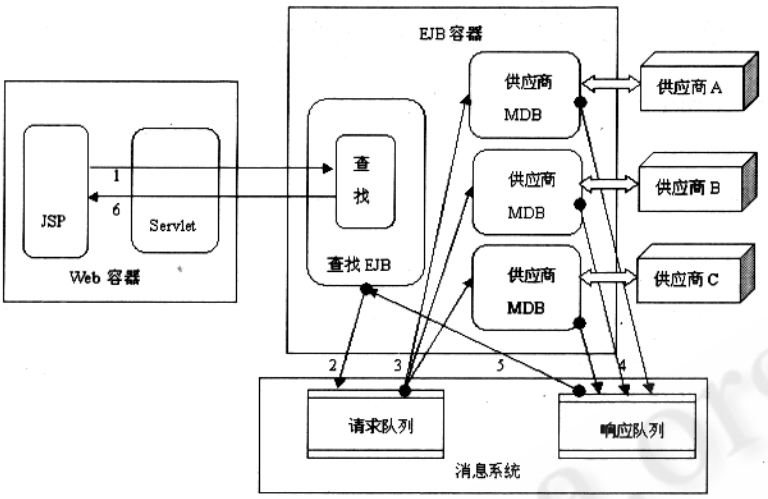

图 3　J2EE 构架下使用消息驱动 Bean 实现并发处理

(3) 还可使用 JMS 过滤器的特性，从响应队列中选择所要的消息。这种情况下对所有的请求仅需要一个响应队列，这需要查找 EJB 组件为每一个消息创建一个唯一的关键字，将它放在消息的头部，以使 JMS 过滤器可以对它选择。

4　消息驱动 Bean 实现的优势

基于消息驱动 Bean 实现并发性不仅提供了更快的响应时间，而且可以保证一个固定的响应时间。例如对于上述实例中的三个供应商系统，如果都正常工作的话，搜索数据返回的时间大约是 15 秒，但如果其中有一个供应商系统出了故障，响应时间还能保证么？基于消息驱动 Bean 实现是可以保证的。当客户请求任务分配给多个供应商 EJB 组件后，可以指定一个最大等待时间，查找 EJB 组件将放弃超出等待时间的响应消息，而把已接收到的响应消息返回到客户端，这使得固定的响应时间得到了保证。对于没有基于消息驱动 Bean 的串行方式实现，这是做不到的。因为一旦有一个供应商系统出了故障，由于是同步阻塞调用，将使响应时间得不到保证。

5　结束语

从上述讨论可以看出，并发性对于提供快速响应时间是必要的。在 J2EE 架构下的各种应用可以通过 JMS 和消息驱动 Bean，很容易实现对客户请求的并行处理，而且能够保证响应时间。

参考文献

1　SUN: Enterprise JavaBeansTM Specification Version 2. 1 [EB/OL]. http://java. sun. com/products /ejb.

2　SUN: The J2EETM 1. 4 Tutorial[EB/OL]. http://java. sun. com/j2ee/tutorial.