基于 MSMQ 的分布式应用架构

高 勇, 吴 健
(西北工业大学 计算机科学与工程系, 陕西 西安 710072)

摘 要: 分布式应用程序为了降低部件相互之间的耦合程度以及对可扩展、可操作性的要求, 通常采用消息中间件的松耦合架构设计。深入探讨了基于 MSMQ(消息队列) 的分布式应用开发, 同时结合东方人寿业务系统给出了一个.Net 平台下基于 MSMQ 的分布式应用解决方案, 说明了基于消息的松耦合技术在分布式应用开发中的可行性与优势。

关键词: 分布式; 消息; .Net; 微软消息队列
中图法分类号: TP311.5 文献标识码: A 文章编号: 1001-3695(2004)10-0180-02

Message-based Distributed Application Architecture

GAO Yong, WU Jian
(Dept. of Computer Science & Engineering, Northwestern Polytechnical University, Xi'an Shanxi 710072, China)

Abstract: By implement message-based software architecture, distributed application decouple the communication between components and then have the extensibility and interoperation. This paper researched MSMQ-based distributed application achitecture and then discuss it's feasibility and advantage through applied easternLife's management information system.

Key words: Distributed; Message; .Net; MSMQ

随着 Internet 的发展, 特别是企业信息化过程的加快, 要求系统具有实时响应、交互动作异步非耦合、高可用性、互操作性等特征。而传统架构模式已经不能适应新的环境, 因此近阶段关于软件体系结构方面的研究也越来越多, 其中以中间件支撑的分布式架构设计最为常见。伴随着 Web Service 和 XML 技术的发展, 对于企业级应用程序而言使用 DCOM/.NET Remoting 可以很方便地实现。但是从实用性、可维护性、系统未来集成等方面考虑, 使用 MSMQ 技术的组件体系结构更为适合。本文探讨了基于 MSMQ 的分布式架构, 提出了相应的应用模式, 并结合东方人寿业务系统得以成功实现。

1 MSMQ 体系结构

1.1 MSMQ 特点

MSMQ(微软消息队列) 技术是一种利用队列机制实现部件间或者是应用程序间通信的技术, 它允许以异步、实时的方式相互传递信息。消息队列是一种灵活而可靠的通信机制, 并且适合于各种程序, 开发人员并不需要了解许多的细节。MSMQ 具有以下特性: ①异步通信。MSMQ 可以分布式组件并以一种异步方式进行通信, 而不必担心占用宝贵的网络资源。②消息路由可靠。MSMQ 保证消息只传递一次, 这样使两个接收者不会意外的得到相同的消息, 消息也不会丢失。③事务集成。MSMQ 可以自动地调用事务服务来保证数据的完整性。④自动消息日志。MSMQ 日志保存发送和接收所有的消息, 并且进行自动的审计跟踪, 发生错误时易于恢复。⑤安全性。MSMQ 具有保密性、数字签名等特性。能够对网络上传送的消息进行数字签名及加密传输, 而且在 MSMQ 服务端可以过滤掉未经授权的消息, 从而保证了安全性。⑥优先级。MSMQ 支持消息队列优先级机制, 它将有选择地传送优先级较高的消息, 让程序能够优先处理重要事件。⑦协议无关性。MSMQ 消息的发送与网络协议无关, 只需提供消息队列的名字即可实现消息的发送与接收。

收稿日期: 2003-06-27; 修返日期: 2003-11-25

1.2 MSMQ 编程模型

.Net 中对于 MSMQ 的支持是由 System.Messaging 命名空间来实现的。对于 MSMQ 的使用, 分为发送端编程和接收端编程。下面分别从发送和接收两个方面进行程序设计。

(1) 发送端

```
public class MSMQSender{
private static MessageQueue m_Queue = new MessageQueue(@
"TESTpublic $ Queue");
public static void Send(string xmlContent) {
将要发送的内容, 调用 m_Queue.Send(xmlContent) 发送到创建的
队列, 如果需要事务处理可以使用类 MessageQueueTransaction 来支持
} }
```

(2) 接收端

```
public class MSMQReceiver{
private static MessageQueue m_Queue;
private ManualResetEvent Done = new ManualResetEvent(false);
public static void Listen() {
如果消息队列已经创建, 获得已有消息队列的应用, 访问该消息队列
while(true)
{
Done.Reset(); //设置异步信号量未"未激活"状态
IAsyncResult ir = m_Queue.BeginReceive(new TimeSpan(0,30,
0), m_Queue, new AsyncCallback(this.ReceiveCallBack));
Done.WaitOne();
ir.AsyncWaitHandle.Close();
} }
private void ReceiveCallBack(IAsyncResult ir) { //接收消息的回调
函数对接收到的消息进行相应的处理
Done.Set(); //激活异步信号量
} }
```

2　.Net 平台下消息模型

在. Net 分布式计算环境中通常采用分层的软件体系结构。①表示层: 用户接口可以使用 Window Forms 或者 Web Forms 的形式, 而对于复杂的用户接口, 可以在表示层提供一个用户处理组件, 对数据进行预处理, 同时对用户的交互进行控制和必要的处理。例如可以使用脚本对数据输入规则进行校验。②业务逻辑层: 它是应用程序的核心部件, 体现商业软件的各种功能。一个商业逻辑是一个完整的业务功能。业务逻辑可以直接供表示层调用, 也可以使用服务的形式提供给表示层。当商业逻辑需要外部的服务程序时, 可以使用服务代理来对外部功能进行访问。工作流是对整个业务流程进行控制的部件。例如利用 BizTalk Server 对业务逻辑进行工作流定义, 或者使用第三方的工作流控制软件来完成业务的自动化处理。③数据层: DALC(数据访问逻辑组件) 可以实现对后台数据库的无关访问, 为不同的数据库提供相同的接口引擎, 屏蔽与数据库相关的细节。因此各种部件之间的通信机制往往决定架构的框架。下面分别介绍基于 MSMQ 的架构模式。

(1) 架构一: "瘦客户"的表示层通过 MSMQ 对应用服务器进行访问, 这样对于表示层与逻辑层实现脱耦, 如图 1 所示。对于表示层可以相对对立地进行修改, 这种架构比较适合多界面风格的应用程序。同时对于安全认证等外部服务实现互操作性, 易于外部组件的替换与升级。但是对于响应较快的应用程序本方案不适合, MSMQ 的通信不能保证响应的质量。

(2) 架构二: 表示层可以直接调用逻辑层的服务, 从而达到较快的响应。而逻辑层与外部服务之间通过 MSMQ 进行通信, 如图 2 所示。这样对于多变的企业环境, 可以实现快速应变, 选择第三方产品或者战略伙伴不会给企业的核心业务造成影响, 从而实现快速应对的架构设计。

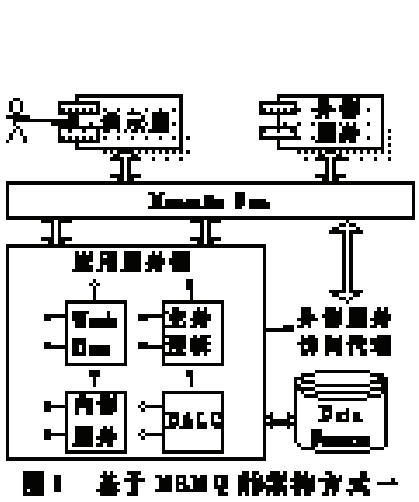

图 1　基于 MSMQ 的架构方式一

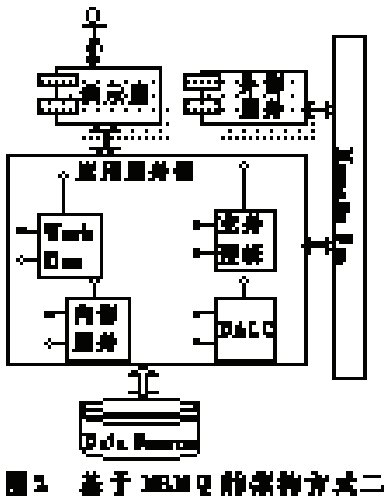

图 2　基于 MSMQ 的架构方式二

(3) 架构三: 从前两种架构种可以看出 MSMQ 应用与层级之间的松耦合, 但是对于小颗粒的组件间的消息机制同样可以使用 MSMQ 方式来降低耦合性, 如图 3 所示。从图 3 中可以看出, 对于商业逻辑与工作流等内部服务之间的通信采用消息队列, 从而小颗粒的组件之间的结构更加清晰, 易于维护和升级。

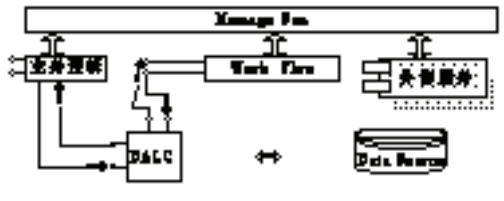

图 3　基于 MSMQ 的架构方式三

3　MSMQ 在东方人寿业务系统的应用

近年来随着经济的迅速发展, 保险公司的业务种类和规模在不断的增加, 因此随着保险公司业务量的急剧膨胀, 保险作业系统信息化越来越成为保险公司的核心竞争力。东方人寿保险作业系统借助当前的. Net 分布式计算环境, 采用了基于中间件技术的分布式应用解决方案。整个业务系统分成如下三层: ①基础平台层。它负责数据类访问逻辑, 同时提供客户端对服务端的对象引用以及在引用上的操作、相关的对象查找(定位) 、分布事务管理、安全机制等内容。②保险业务逻辑层。一般保险行业需求有关的设施、部件, 使平台能适应更多的保险行业。③东方寿险业务层。与东方寿险的具体业务相关的部分, 这是该软件系统中真正的应用层。同时东方寿险业务系统使用了应用服务器, 一些安全、事务处理/管理机制由该应用服务器完成。基于这种架构, EasternLife. Sys. Server 代表整个寿险系统的应用服务器, 部署在服务端, 相当于一个容器, 在此基础上运行商务逻辑和提供相关的工作流、安全、事务处理/管理机制、外部服务。EasternLife. Sys. DataClass 负责客户端对服务端对象的引用, 以及在引用上的操作。EasternLife. Sys. SecurityService 负责整个寿险系统的安全服务。EasternLife. Sys. WorkFlow 负责作业系统的工作流管理与调度。EasternLife. Sys. DALC 负责数据访问逻辑的功能(DALC) , 作为数据访问的组件。EasternLife. Sys. DALC. Integration 服务负责异构数据的集成功能, 从而与业务合作伙伴进行数据集成与转换。EasternLife. Sys. WinUI 负责表示层的界面元素, 实现界面元素的最大重用, 达到界面简洁、易操作的风格。而业务逻辑(如 EasternLife. NewBusiness, EasternLife. Sales 等) 根据东方寿险的业务进行组件化设计, 部署商业逻辑。应用服务器端各个层级的组件采取基于 MSMQ 的松耦合通信架构, 系统架构如图 4 所示。

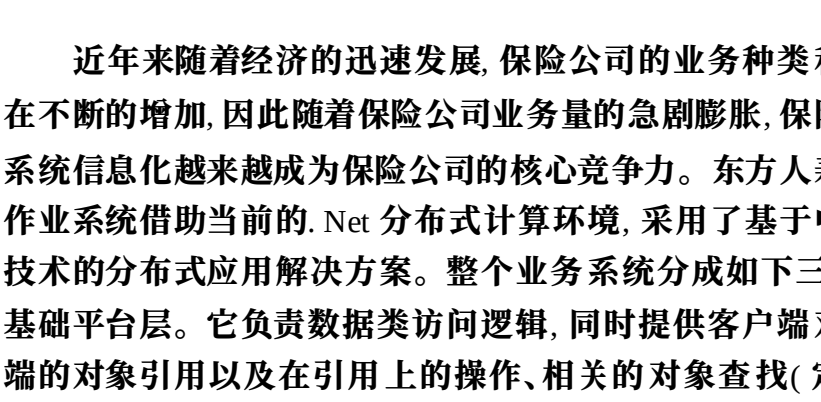

图 4　系统架构图

从图 4 中我们可以清楚地看出, 对于业务逻辑(新契约、销售管理、计划书等) 依靠工作流来驱动, 从而以任务驱动为基础的作业管理, 实现全国服务一体化, 使保险作业流程制度化、透明化。同时在不影响服务质量的前提下, 尽可能地实现业务的集中化。对于业务逻辑、内部服务部件与工作流之间的 MSMQ 详细实现如图 5 所示。

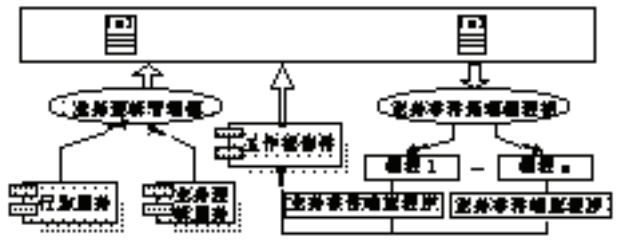

图 5　MSMQ 详细实现

(下转第 201 页)