计算机科学2002Vol.29№.2

# 消息队列技术研究:综述与一个实例

Study on Message Queuing Technology:Overview and a Case Study

周世杰　刘锦德　秦志光

(电子科技大学微机所　成都610054)

**Abstract**　The features and applications of message queuing technology are analyzed. After discussing the communication mode of message queuing, the differences among message queuing, remote procedure call and distributed transaction processing are clarified. Then, several kinds of application design patterns used in common application scenarios are carefully considered. Finally, how to design a real middleware system adopting message queuing technology is explained in detail.

**Keywords**　Message queue, Middleware, Message queue middleware, Enterprise Application Integration (EAI)

## 1　前言

商业企业正面临着沉重压力,如何适应急速变化的市场正成为他们所关注的焦点。现在,许多商业企业和IT界的专家都意识到,仅仅依赖开发并提供全新的应用来适应和满足市场需求,常常是不够的。对新市场机遇最快捷的响应是,重用已有的种种应用,将它们与一些最新技术成果置在一起并借助信息传递渠道来集成。目前流行的企业应用集成(EAI)正在沿用这样一种方式。对于EAI,在应用程序之间提供适应性强、使用方便的通信手段是成功的关键。EAI中所使用的通信手段有许多,而其中消息队列技术则是极受人们重视的一种[1],因为它能将运行在异种计算机上的应用有效地沟通、可靠地集成为一体。

## 2　基于消息队列的通信模式[1]

应用程序之间可用的通信方式主要有远程过程调用(RPC)、分布事务处理(DTP)以及消息队列技术(MQ),图1描述了这三种技术的本质区别。RPC技术的不足之处在于它的脆弱性,即当调用过程中有失败发生时,它无力提供可恢复的功能,因此RPC不能为应用程序开发者提供开发出健壮程序的手段。DTP技术适合用来构建重要的事务处理系统(比如银行事务系统),因为这种技术确保了每一事务执行的ACID属性,也即防止了调用过程中发生各种失败的可能性,这就使应用程序开发者减轻了负担。但是,DTP系统很昂贵,并且现成产品的DTP系统实际所支持的系统平台非常有限,从而限制了DTP的广泛使用。

上述RPC和DTP共同之处是:只支持同步通信,即客户方和服务器方必须同时工作着。但是,有许多应用场合客户方和服务器方不一定非要同时工作着(甚至有时不同时工作更合适),也即这时可以采用异步的通信方式。MQ就是适合这种情况的一种技术[1]。由图1可见,使用MQ技术时,服务器和客户机之间不需要直接通信。事实上,客户机不知道谁将要来处理自己的请求,它只是与本地某个实体交往(比如,在IBM MQSeries中这一实体被称之为队列管理器)、实际的网络和通信等具体操作则完全由消息队列系统来完成。另外,MQ技术通过存储-转发技术来提供健壮、可恢复的消息通信能力,例如消息在通过消息通道转发前先进行本地存储,只有成功传递到消息目的地后该消息才被删除。消息的本地存储技术可以是日志技术,也可以是基于内存的消息暂存技术[2],这取决于实际应用的需要。

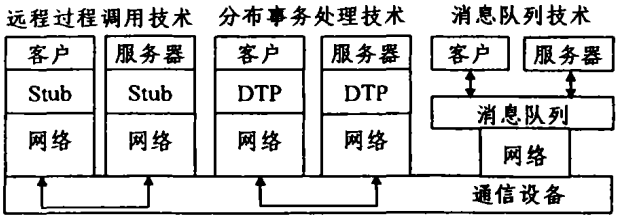


图1　RPC、DTP和MQ比较示意图

## 3　基于消息队列技术应用的设计模式[1]

在MQ技术中,应用程序不是直接将消息传递到合作对方的应用程序,而是将消息写入消息队列;随后第二个应用程序则异步地从消息队列中获取并处理该消息。当然,这个应用程序可能就是合作方,并向消息始发者(即消息源)发回应答(也可以只收不答);也可能它只是将该消息转发到另外的消息队列,而后为第三个应用程序所取用。这种利用消息流一步步地沟通应用程序的方式,与UNIX中依靠管道连接应用程序十分相似;但是所不同的是:在消息队列方案中,每一步的传递中都有队列(它是由内存和外存相结合而构成)介入,这就确保消息具有了可靠性和可恢复性。

由于众多的应用程序都可以用MQ技术来构建,这意味着在实际场合中MQ技术会面临种种复杂的情况。事实上,开发基于MQ技术的应用程序的关键就是把握住如何将MQ产品所提供的功能能够与具体的应用相结合,即为解决应用恰如其分地选用设计模式。以下是五种可供选用的设计模式。

### 3.1　异步查询(AI)

异步查询是一种应用设计模式,它允许客户向服务器查询信息,同时客户不需要等待确认消息的到来即能继续往下工作。图2是异步查询的消息流示意图。

周世杰　博士研究生,主要研究方向:开放系统与中间件技术,ITS技术。刘锦德　教授,博士生导师,主要研究方向:开放系统与中间件技术,网络与多媒体技术。秦志光　博士,副教授,主要研究方向:开放系统与安全技术,ITS技术。

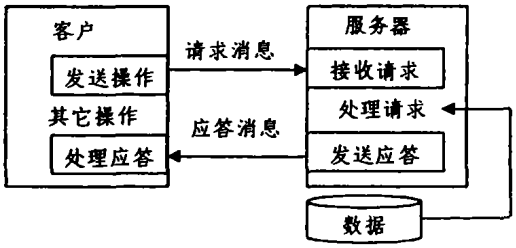


图2 异步查询设计模式示意图

由图可见，在本查询设计模式中，客户方的查询过程由两个独立的操作完成：发送请求操作和处理应答操作。发送请求操作完成后，客户方不需要等待应答，而是继续执行其它操作（比如发送另一新的请求等）；当应答到达时，由处理应答操作程序接收应答并作相应处理。因此，客户方可处于完全异步的方式工作。服务器方接收查询请求，完成查询操作，并将查询结果返回客户方。这是一个标准的查询过程，显然数据不会被客户的请求操作所破坏，因而即使发生失败，操作可以重复性地进行，这将简化应用程序的设计和开发。由于这种应用设计模式简单可靠，所以广泛地为查询应用程序所采用。

### 3.2 伪同步查询(PI)

伪同步查询是一种与 AI 稍有不同的应用设计模式，它允许客户机向服务器查询数据，并等待查询结果的返回。之所以称之为"伪"同步，即是在这一模式中，查询客户并不会永久地去等待一个不返回的查询结果。图3是伪同步查询的消息流示意图。

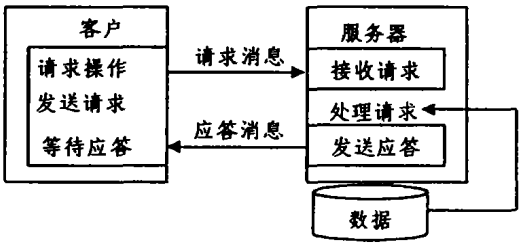


图3 伪同步查询设计模式示意图

由图可知，在这种查询设计模式中，客户机的查询过程被封装成一个操作（请求操作），这个请求操作分两步完成：发送查询请求和等待应答。服务器方的处理过程则分为三个独立的操作：接收请求、处理请求（包括读取数据）和发送应答。伪同步查询有两个主要特点：一是对客户方操作而言是同步方式工作的，即在应答没有到达（或者无超时指示）前，客户机不会放弃对当前程序的控制；二是这是一个查询过程，因而客户机不会影响服务器上的数据，这就意味着如果发生失败，客户操作可以重复性地进行。这种设计模式的难点在于当应答始终不到达时，客户方如何来确定其为超时。

### 3.3 异步确认更新(AU)

异步确认更新是一种应用设计模式，它允许客户机向服务器作更新数据操作，在确认消息返回前可以继续执行其它的操作。这是应用广泛，但设计难度较大的一种模式。图4是异步确认更新的消息流示意图。

在本设计模式中，客户方是执行的异步操作过程：客户方发送更新操作请求，随后继续其它操作；当应答消息到达时再处理该应答消息。在服务器方，接收更新请求，处理该请求并向客户方返回确认消息。因为这是一个标准的数据更新过程，所以不存在操作的可重复性。该设计模式的难点在于确保更新操作可靠地一次执行完成。这可以通过消息一次性传递技术和消息日志技术来解决[2]。

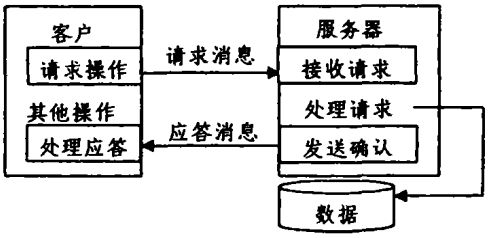


图4 异步更新设计模式示意图

### 3.4 无确认更新(UW)

无确认更新是与 AU 稍有不同的一种应用设计模式，它允许客户对服务器更新数据而却不需要等待确认消息。图5是无确认更新的消息流示意图。

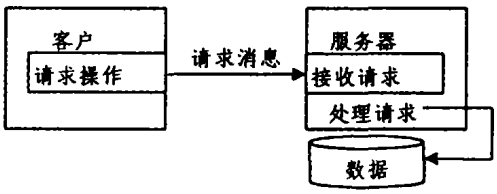


图5 非同步更新设计模式示意图

由图可见，客户方发送更新请求后，不需要等待更新成功与否的应答，因此是完全异步的过程；服务器方接收客户机的更新请求后执行数据的更新操作，但不向客户方返回更新结果的应答。由于客户方不等待更新操作的应答，为了确保更新操作的成功，客户机必须依靠服务器的可信性：客户机确信服务器一定会完成自己的更新请求。同时，这也意味着服务器不一定需要在接收到更新请求之后立即完成更新操作，该操作可以被延迟执行。因此，在无确认更新设计模式中，服务器方必须具有存储更新请求消息的机制（一般是日志技术），以确保更新操作一定能成功完成。无确认更新设计模式的主要缺点是无法提供失败处理的结果。

### 3.5 伪同步更新(PU)

伪同步更新是介于 AU 和 UW 之间的一种应用设计模式，它允许客户向服务器作更新数据操作并等待更新操作的结果。这种方式中，更新操作也允许失败，因此客户方必须提供处理更新失败的机制。图6是伪同步更新的消息流示意图。

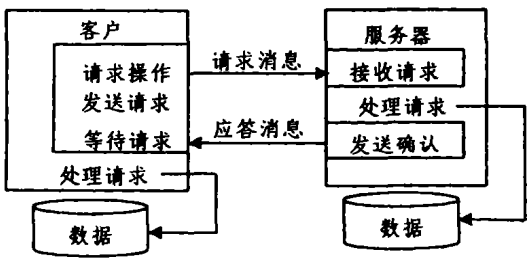


图6 伪同步更新设计模式示意图

在这种设计模式中，客户方的更新数据过程封装到一个操作中完成，是一个完全的同步过程；服务器接收更新请求，完成数据更新并向客户方返回更新操作结果的应答消息。因为这是一个标准的更新操作，所以不存在操作的可重复性。伪

同步更新设计模式的难点是，当更新确认消息没有返回时客户方如何处理。典型的方法是就假设服务器操作失败，记录失败信息，并重发更新请求。为此，应用程序必须提供这些额外的操作。

## 4 设计实例：如何利用消息队列技术构造一个中间件[2~6]

根据对消息队列技术中设计模式的分析可知，数据查询应用由于允许操作的可重复性，因此以采用异步方式（如 AI 模式）进行效率最高。数据更新应用中，由于操作没有可重复性，因此使用同步方式（如 PU 模式）才能确保数据完整性，但是这样效率较低；如果使用异步更新模式（如 AU 模式），效率高，可靠性好，但是设计的复杂度增加。参照以上的分析，用合适的设计模式来开发适用于实际应用的消息队列系统，它具有良好的效率、可靠性和其它优良的性能，是极有实用意义的。ITS-MQ 消息队列中间件[2]正是这样一个应用系统。图7是这一自行设计并已付之实用的消息队列中间件的消息流示意图。

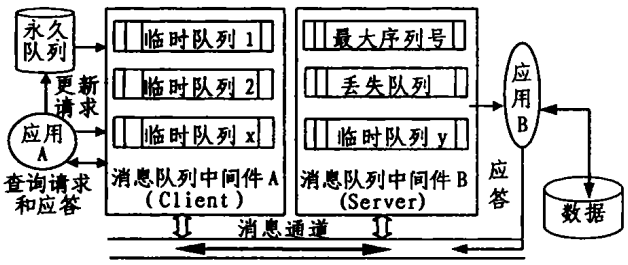


图7 ITS-MQ 消息队列中间件示意图

该系统采用消息队列技术解决可靠性问题。它的基本思想是对消息和消息的应答分别进行处理：发送端将消息同时放到临时消息队列和永久消息队列中，只有收到消息的确认消息后，发送端才将消息从队列中清除，否则根据需要重传消息；对于消息的确认信息则不进行可靠性保障。对重复消息的处理（即如何防止向应用程序传递重复的消息），则通过有状态的中间件技术来解决。其基本思想是让中间件系统中的服务器端运行平台是有状态的，即中间件系统必须记忆与它通信的客户机已经发送并被正确接收到的消息的信息，或者是与此消息相关的信息。维护这种信息的基本方法有三：最大序列号、接收队列或者最大序列号与丢失队列结合[2]。由于最大序列号法和接收队列法各有优缺点，都只适合于特定的应用，因此采用最大序列号与丢失队列结合的方法是一种合理的解决方案。在这种方案中，客户机对将要发送的消息进行唯一、连续、递增性编号，在服务器上保存每个客户机已经发送并被正确接收到消息的最大序列号，该最大序列号随即表示出下一个将要接收的消息的序列号；同时，服务器还为每一个客户机建立一个循环队列，该队列保存那些小于最大序列号且没有收到的消息序列号（因此可称为丢失队列）。如果收到消息等于最大序列号，则可以发送给应用程序；如果收到小于最大序列号的消息，则查找丢失队列。如果在丢失队列中，则该消息是第一次接收，发送给应用程序；如果不在丢失队列中，该消息是重发消息，将其抛弃；如果收到消息的序列号大于最大序列号，则将该消息发送给应用程序，同时将该消息的序列号与最大序列号之间（包括最大序列号）的消息加入到丢失队列中。

当执行数据查询时，应用程序在将该查询请求发送到消息队列中间件客户端后，可继续执行其它操作（比如发送另一新的查询请求），不需要等待查询结果；查询请求由消息队列中间件客户程序通过消息通道传送到对等的中间件服务器端，中间件服务器程序唤醒相应的查询程序（比如查询线程），执行数据查询，将查询结果返回中间件服务器；后者将查询结果通过消息通道返回到中间件客户端，中间件客户程序唤醒处理返回应答的应用程序（比如应答处理线程），完成应答处理（比如显示数据）；如果出现异常（比如应答丢失），应用程序则做重新查询。由此可见，数据查询操作是一个异步的过程，效率很高，也很可靠。

当需要执行数据更新时，应用程序在将该更新请求发送到消息队列中间件客户端后，继续执行其它操作（比如发送另一新的更新请求），不需要等待更新结果；同时，该请求消息被临时性地保存到本地永久队列中（比如本地数据库系统）。消息队列中间件客户程序将更新请求消息通过消息通道发送到中间件服务器端，该中间件服务器程序唤醒更新操作应用程序（比如更新数据库的线程），并保存该消息的信息（比如增加最大序列号）；更新操作应用程序完成数据更新操作（比如写数据库），并将更新操作结果确认消息返回中间件服务器端。中间件服务器程序将确认消息通过消息通道传送到中间件客户端，中间件客户程序唤醒应答处理程序；后者处理该应答信息（比如显示更新操作结果），并将该应答所对应的更新请求消息从本地永久队列中删除（比如从数据库中删除）。如果更新操作确认消息丢失，中间件客户程序会超时重发（或者周期性重发）消息队列中的请求消息，中间件服务器程序根据最大序列号和丢失队列判断出该请求是重发消息，所以返回更新确认消息，要求中间件客户程序删除该请求消息（包括从本地永久队列和中间件系统的消息队列删除）。由此可见，对于数据更新操作，这也是一个异步的过程，效率很高，且十分可靠。

**结论** 迅猛增长的企业应用集成需要新的应用间通信技术来支撑，消息队列正是这样一种技术。消息队列技术的通信模式与远程过程调用以及分布事务处理的本质区别在于，它能提供异步的通信模式。利用消息队列技术开发 EAI 的关键是掌握其常用的应用设计模式。这些设计模式可以分为查询模式和更新模式；同时，还有异步和伪同步的区别。善于运用这些设计模式可以设计和开发出符合实际需要的消息队列中间件。

## 参考文献


1 Lewis R. Advanced Message Application with MSMQ and MQSeries. QUE. 1999,11

2 周世杰. 网络环境下中间件技术研究与开发：[电子科技大学硕士论文]. 2000,12

3 Tanenbaum A S. Computer Networks. 北京：清华大学出版社，1998,4

4 Kruse R L. Data Structure & Program Design in C. 北京：清华大学出版社，1999,2

5 Stallings W. Operating System Internals and Design Principle. 北京：清华大学出版社，1998,6

6 Stevens W R. UNIX Network Programming ，Volume 1. 北京：清华大学出版社，1998,12