第 27 卷第 3 期　　苏 州 大 学 学 报(工 科 版)　　Vol. 27 No. 3
2007 年 6 月　　JOURNAL OF SUZHOU UNIVERSITY(ENGINEERING SCIENCE EDITION)　　Jun. 2007
文章编号: 1673－047X(2007)03－0010－05

消息中间件的体系结构研究*

李　璐, 张广泉
(苏州大学计算机科学与技术学院, 江苏 苏州 215006)

摘　要: 消息中间件可屏蔽分布式环境中的异构性和复杂性, 为分布式应用系统提供可靠、透明的通信服务。对消息中间件的定义和标准进行了概述, 对消息中间件的几种体系结构进行了介绍, 并以执行时间为标准对远程过程调用、消息中间件的发布/订阅模型和改进的发布/订阅模型进行了定量比较。

关键词: 消息中间件; 软件体系结构; 消息队列; 发布/订阅; 远程过程调用
中图分类号: TP311　　文献标识码: A

0　引言

随着计算机技术的飞速发展, 计算机的应用范围越来越广泛。硬件方面, 大批新型的硬件产品相继出现, 使得在一个系统中存在着多种不同类型的硬件产品, 这样的系统通常被称为异构系统; 软件方面, 系统的规模不断扩大, 由以往的集中式向分布式转变。为了解决在分布异构环境下不同类型、不同地理位置的软、硬件系统之间的通信、集成, 人们提出了中间件的概念。中间件是一种位于系统软件和应用软件之间的分布式软件, 它屏蔽了分布式环境中的异构性和复杂性, 使分布在多种硬件和不同操作系统平台上的应用可以透明地交互和共享资源。消息中间件作为一种提供高可靠消息传输的中间件, 应用十分广泛。

本文首先介绍消息中间件的概念、标准和相关产品; 然后对消息中间件的几种常见体系结构进行了介绍, 并以执行时间为标准对远程过程调用、消息中间件的发布/订阅模型和改进的发布/订阅模型进行了定量比较。

1　消息中间件概述

1.1　消息中间件的定义

消息中间件简化了应用之间的数据传输, 屏蔽了底层平台的异构性, 为分布式应用系统提供了透明的通信服务, 确保了分布式环境下可靠、高效、便捷的跨平台信息传输。它是一种异步、无阻塞、基于消息的通信技术, 具有良好的可靠性和灵活性, 特别适用于逻辑关系松散的分布式环境[1]。

1.2　消息中间件的标准和主要产品

虽然消息中间件作为一种主流的通信技术已出现多年, 但始终没有一个统一的技术标准, 厂商依据各自的标准开发消息中间件产品, 导致不同的产品难以集成。直到 SUN 公司发布了 Java 消息服务(JMS)规范后, 这一状况才有所改善。JMS 规范只提供了一组与具体实现无关的接口和相关语义, 由各个厂商来提供具体的实现[2]。

目前进行消息中间件开发的厂商主要有 IBM、BEA、Tibco、东方通科技等公司, 主流产品主要有

* 收稿日期: 2007－03－27
作者简介: 李　璐(1983－), 男, 硕士研究生, 主要研究方向为中间件与软件体系结构。
基金项目: 江苏省高校自然科学基金资助项目(编号 05KJB520119); 重庆市自然科学基金资助项目(编号 CSTC, 2006BB2259)。

MQSeries、MessageQ、Rendezvous、TongLink/Q 等。

2　消息中间件的体系结构

消息中间件有两种主流通信模型: 消息队列模型[3] 和消息传递模型。其中, 消息传递模型又可分为点对点模型[2] 和发布/订阅模型[2] 。

2.1　点对点模型

点对点模型是一种由应用程序直接到应用程序的简单通信模型。消息发送方根据消息接收方的地址将消息直接发送给接收方。由于通信双方采用的是直接、同步的通信模式, 所以通信双方必须是逻辑上相连的。

点对点模型的主要优点是实现简单, 不需要过多配置; 缺点是通信双方紧密耦合, 在灵活性上受到很大制约, 不适用于松耦合的分布式环境。

2.2　消息队列模型

消息队列模型是一种应用程序之间的非直接通信模型, 它使用消息队列来存储消息, 允许应用程序通过消息队列来实现异步的消息通信。消息队列分为暂时性队列和永久性队列。暂时性队列存放于内存中, 在系统崩溃时会消失, 无法恢复; 而永久性队列一般存放于硬盘或者其他备份设备中, 在系统恢复后仍然存在。

消息队列模型的主要优点是支持异步通信, 使得消息的发送方和接收方之间不必存在直接连接, 实现了发送方与接收方在时间上的松耦合; 缺点是队列需要配置, 性能不高, 且队列一旦丢失, 整个系统将受影响。

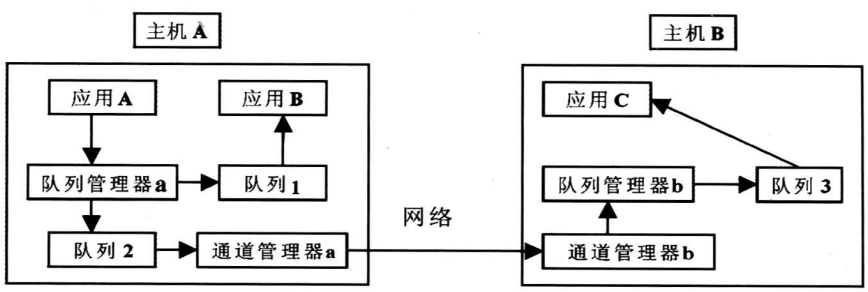

图 1　消息队列模型

消息队列模型的工作流程如图 1 所示。对于同一主机的应用 A 与 B 间的通信: 应用 A 通过队列管理器 a 将消息放入队列 1, 随后根据触发机制, 触发事件通知应用 B 从队列 1 中取走消息; 对于不同主机间的应用 A 与 C 的通信: 应用 A 通过队列管理器 a 将消息放入队列 2, 再经主机 A 的通道管理器 a 将消息转发给主机 B 的通道管理器 b, 然后通道管理器 b 通过队列管理器 b 将消息放入队列 3, 最后应用 C 被触发机制触发, 从队列 3 中取走消息。

2.3　发布/订阅模型

发布/订阅模型是一种使分布式系统中的各个应用以发布/订阅的方式进行交互的基于事件驱动的通信模型。在发布/订阅模型中, 发布者和订阅者之间所交互的消息称为事件[4], 其工作流程如图 2 所示。

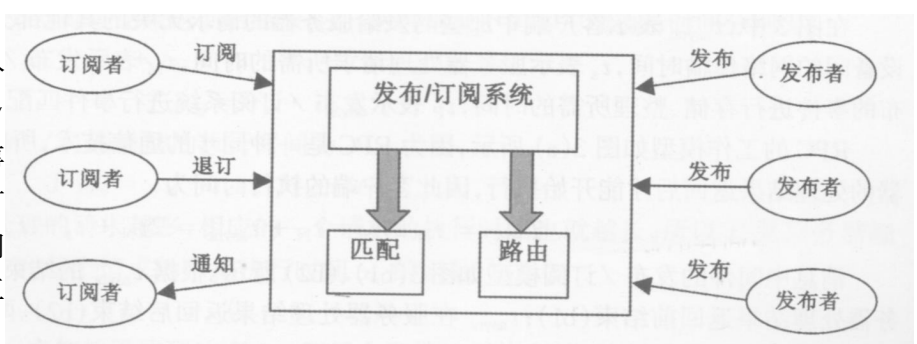

图 2　发布/订阅模型

发布者将事件发送给发布/订阅系统, 订阅者则向发布/订阅系统发出一个订阅条件, 这个条件用来表示订阅者对系统中的哪些事件感兴趣, 当订阅者对事件不感兴趣时, 也可取消先前的订阅。发布/订阅系统负责将发布者发布的事件传送给所有订阅它的订阅者, 通常使用消息代理技术实现。

发布/订阅模型的主要优点是通过发布/订阅系统实现具体的交互, 因此发布者和订阅者不需要同时处

于活动状态,也不需要知道对方的位置,实现了时间、空间上的松耦合[3];发布/订阅系统通常由消息代理技术实现,消息代理可以实现消息的动态路由,具有较好的容错能力。

发布/订阅是从现实中的贸易应用演化而来,对发布/订阅的相关研究包括发布/订阅系统的拓扑结构、事件模型、订阅模型、匹配算法、路由算法以及服务质量的保证等多个方面[4]。

3 消息中间件和 RPC 的定量比较

远程过程调用(RPC)是一种分布式环境下的同步通信技术,它允许运行在客户端的应用程序调用远程服务器上的过程。服务器端侦听客户端的请求,当客户端请求到达后,调用指定的过程进行处理,并将处理结果返回给客户端。在调用过程中,客户端与服务器必须同时处于活动状态,且在处理结果返回前,客户端将一直处于阻塞状态[6]。

消息中间件也是一种分布式环境下的通信技术,与 RPC 不同,它是一种异步的通信技术。消息中间件的发布/订阅模型除了具有消息中间件的异步通信特性外,还具有松耦合、配置灵活和容错性好等优点,应用非常广泛。传统的发布/订阅模型通过一个位于固定服务器上的发布/订阅系统来实现事件的发布/订阅,事件的储存、匹配、发送都由发布/订阅系统完成,这往往成为消息中间件的性能瓶颈,一定程度上限制了整个系统的性能。

为了解决性能瓶颈问题,一种可行的办法是将 Web Service 的体系结构引入到消息中间件中:发布者在发布/订阅系统上注册要发布的事件,发布/订阅系统对已注册的事件和订阅条件进行匹配后,将订阅者的地址和订阅信息发给对应的发布者,发布者直接将对应事件发给订阅者。这种发布/订阅模型在一定程度上减轻了发布/订阅系统的负担,提高了系统性能,我们称之为改进的发布/订阅模型。

为了进一步说明上述通信模型的特点,下面以客户端的执行时间为标准对上述 3 种通信模型进行定量比较。

3.1 基本模型

首先,我们考虑分布式环境下的一种常见情况:客户端在执行过程中,需要使用远程服务器上的资源,客户端将请求发往服务器,服务器处理后将结果返回给客户端。我们假定客户端在向服务器发送请求后,可以执行应用程序中与该请求无关的其他部分,直到处理结果返回客户端。在这种情况下,服务器为发布者,客户端为订阅者。显然,远程过程调用、消息中间件的发布/订阅模型和改进的发布/订阅模型都可用于客户端和服务器间的通信,图 3 分别给出这 3 种模型的通信流程。

3.2 计算公式

在图 3 中,t_{async} 表示客户端中那些与发给服务器的请求无关的其他部分的执行时间,t_{N} 表示任意两个设备间的网络传输时间,t_{s} 表示服务器处理请求所需的时间,t_{M} 表示发布 / 订阅系统(P/S)对服务器所发布的事件进行存储、整理所需的时间,t_{P} 表示发布 / 订阅系统进行事件匹配所需的时间。

RPC 的工作模型如图 3(a) 所示,因为 RPC 是一种同步的通信技术,所以客户端的其他部分必须在服务器的处理结果返回后才能开始执行,因此客户端的执行时间为

$$T_{\mathrm{RPC}} = t_{\mathrm{N}} + t_{\mathrm{s}} + t_{\mathrm{N}} + t_{\mathrm{async}} = 2t_{\mathrm{N}} + t_{\mathrm{s}} + t_{\mathrm{async}}$$

消息中间件的发布 / 订阅模型如图 3(b1)、(b2) 所示。根据 t_{async} 的结束时间可分为两种情况:t_{async} 在服务器处理结果返回前结束(b1);t_{async} 在服务器处理结果返回后结束(b2)。两种情况下的客户端的执行时间分别为

$$T_{\mathrm{P/S}} = t_{\mathrm{N}} + t_{\mathrm{M}} + t_{\mathrm{N}} + t_{\mathrm{P}} + t_{\mathrm{N}} = 3t_{\mathrm{N}} + t_{\mathrm{M}} + t_{\mathrm{P}}$$

$$T_{\mathrm{P/S}} = t_{\mathrm{N}} + t_{\mathrm{M}} + t_{\mathrm{async}}$$

综合上述两式可得发布 / 订阅模型的执行时间为

$$T_{\mathrm{P/S}} = t_{\mathrm{N}} + t_{\mathrm{M}} + \max\{2t_{\mathrm{N}} + t_{\mathrm{P}},\ t_{\mathrm{async}}\}$$

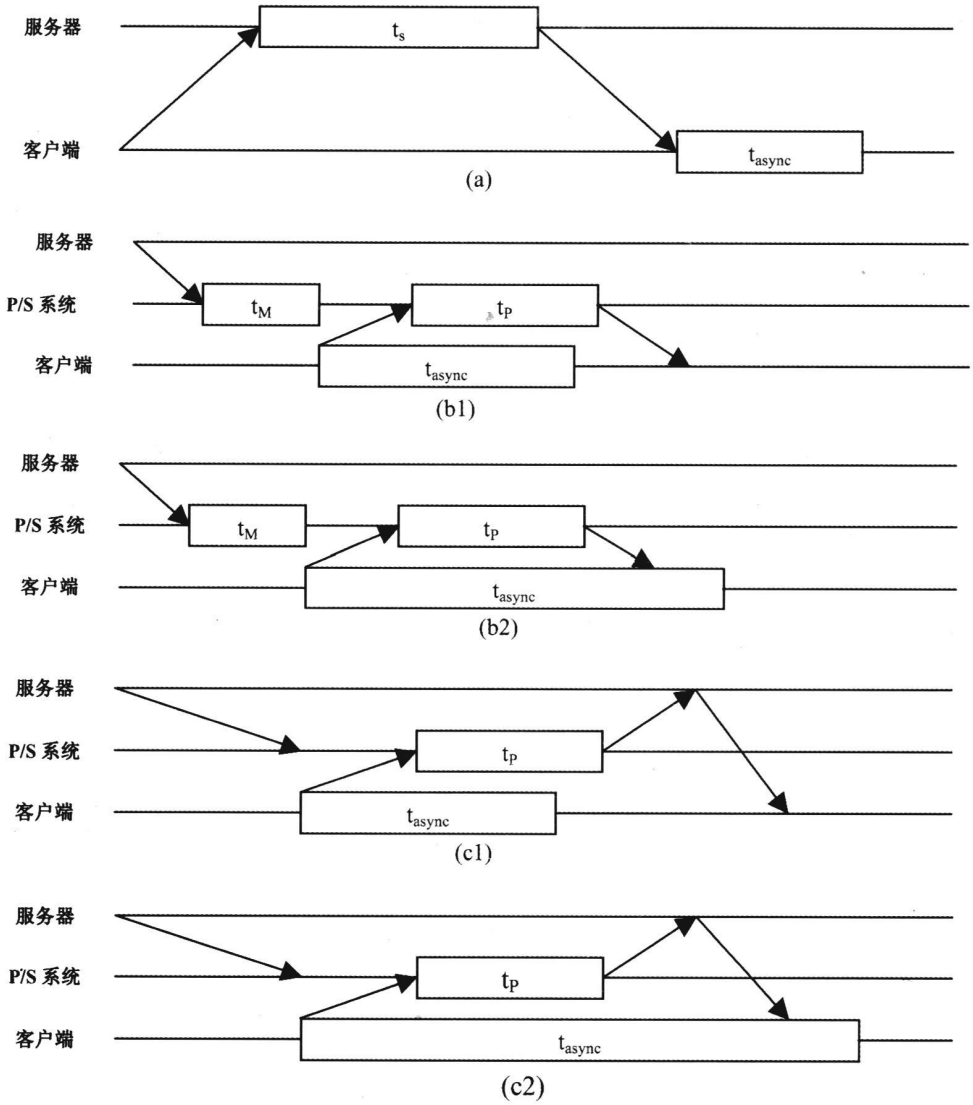

图 3　通信流程

改进的发布/ 订阅模型如图3(c1)、(c2)所示。与发布/ 订阅模型一样, 它也分为两种情况: t_{async} 在服务器处理结果返回前结束(c1); t_{async} 在服务器处理结果返回后结束(c2)。两种情况下的客户端执行时间分别为:

$$T'_{P/S} = t_N + t_N + t_P + t_N + t_N = 4t_N + t_P$$

$$T'_{P/S} = t_N + t_{async}$$

综合上述两式可得改进的发布 / 订阅模型的执行时间为

$$T'_{P/S} = t_N + \max\{3t_N + t_P, t_{async}\}$$

在同等条件下, 服务器所要处理的请求越多, 相应的一个请求的执行时间也就越长, 所以 t_s 是服务器端请求的平均到达率 h_s(请求/ 秒)和每个请求的平均执行时间 x_s 的函数[7]:

$$t_s = x_s/(1 - h_s \cdot x_s)$$

同理, t_M 是发布 / 订阅系统中事件的平均到达率 h_M 和每个事件的平均存储、整理时间 x_M 的函数:

$$t_M = x_M/(1 - h_M \cdot x_M)$$

考虑到事件的匹配时间 t_P 由服务器发布的事件和客户端的订阅条件共同决定, 所以令

$$t_P = k_1 \cdot t_s + k_2 \cdot t_M \quad k_1, k_2 \in (0, 1) \text{ 且 } k_1 + k_2 = 1$$

令客户端执行其他部分所用的时间 $t_{async} = k \cdot t_s$, k 为变量且 $k \in [0, 10]$。

对上述变量赋值: $t_N = 0.3$ 秒, $x_s = 0.1$ 秒, $h_s = 5$ 请求 / 秒, $x_M = 0.07$ 秒, $h_M = h_s$, $k_1 = 0.6$, $k_2 =$

0.4,可分别得到3种模型的执行时间 T 与变量 k 的函数,结果如图 4 所示。

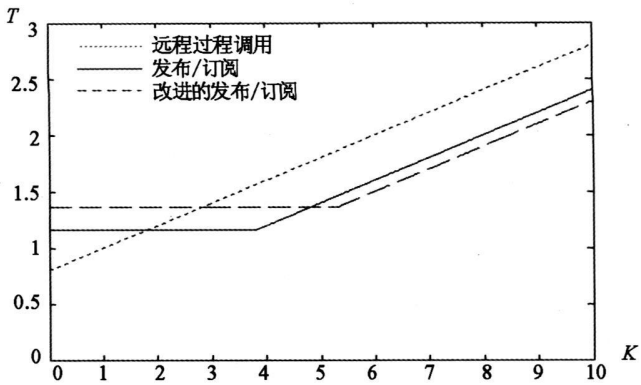

图 4 实验结果

3.3 结果分析

由图 4 并结合计算公式可知,当客户端的应用程序规模较小(k 较小)时,远程过程调用的执行时间最少,性能较好。当 $t_{async} = t_N + t_M + t_P - t_s$ 时,远程过程调用和发布/订阅模型的执行时间相同;当 $t_{async} = 2t_N + t_P - t_s$ 时,远程过程调用和改进的发布/订阅模型的执行时间相同;当 $t_{async} = 3t_N + t_P - t_M$ 时,发布/订阅模型和改进的发布/订阅模型执行时间相同。此后,随着客户端应用程序规模的扩大,改进的发布/订阅模型的执行时间最少,发布/订阅模型次之,远程过程调用的执行时间最长。由此可见,远程过程调用适用于数据规模较小的情形,消息中间件的发布/订阅模型和改进的发布/订阅模型则适用于数据规模较大的情形。

4 结束语

本文对消息中间件的相关概念和体系结构进行了介绍,并以客户端应用程序的执行时间为标准对RPC、消息中间件的发布/订阅模型和改进的发布/订阅模型进行了定量比较,从定性描述和定量分析两个方面对消息中间件的体系结构进行了全面研究。消息中间件作为一种处于发展中的中间件,还有很多方面需要做进一步的研究,如对服务质量(QoS)的保证[8],发布/订阅模型中对语义和复合事件的支持[4]等,都是目前的研究热点。

参 考 文 献

[1] 徐 晶,许 炜. 消息中间件综述[J]. 计算机工程, 2005, 31(16): 73-76.
[2] SunMicrosystems. JMS specification version1.1. [EB/OL] [2002]. http://java.sun.com/products/jms.
[3] Tran P, Greenfield P. Behavior and Performance of Message-Oriented Middleware Systems[C]. Proceedings of the 22nd International Conference on Distributed Computing Systems Workshops, Vienna, 2002: 645-650.
[4] 马建刚, 黄 涛, 汪锦岭, 等. 面向大规模分布式计算发布订阅系统核心技术[J]. 软件学报, 2006, 17(1): 134-147.
[5] Eugster PT, Felber PA, Guerraoui R, et al. The many faces of publish/subscribe[J]. ACM Computing Surveys, 2003, 35(2): 114-131.
[6] Vinoski S. Where is Middleware[J]. IEEE Internet Computing, 2002, 6(2): 83-85.
[7] Menasce DA. MOM vs. RPC: Communication Models for Distributed Application[J]. IEEE Internet Computing, 2005, 9(2): 90-93.
[8] 郭 胜,许 平,王 颖, 等. 中间件技术的研究[J]. 计算机科学, 2004, 31(2): 155-159, 180.

Research of Message-oriented Middleware Architecture

LI Lu, ZHANG Guang-quan
(*College of Computer Science and Technology, Suzhou University, Suzhou* 215006, *China*)

Abstract: Message-oriented middleware can shield the difference and complexity in distributed environment. It provides reliable and transparent communication services for distributed applications. The definition and standard of message-oriented middleware are summarized and several architectures of message-oriented middleware are introduced. Then the execution time is selected as a standard to quantitatively compare remote procedure call, publish/subscribe model and improved publish/subscribe model of message-oriented middleware.

Key words: message-oriented middleware(MOM); software architecture; message-queuing; publish/subscribe; remote procedure call(RPC)