

消息中间件的排队论模型研究^{*}

Study on the Queuing Model of the Message-Oriented Middleware

周世杰 秦志光 刘锦德

(电子科技大学计算机科学与工程学院 成都610054)

Abstract The general concepts in message queuing and message-oriented middleware(MOM) are discussed. The essential elements and model of the queuing theory are analyzed in detail. Then, after putting forward and analyzing the basic formulas of the queuing theory, the applicable model for message queue middleware is probed into. Some important formulas used in the designing of the message queue middleware are also elucidated. Finally, a case of designing the message queue middleware adopting queuing model we put forward, which is used in the freeway toll system of ITS, is studied.

Keywords Message queuing, Message middleware, MOM, Queuing theory, ITS, Freeway toll system

1 引言

分布式和跨平台用户需求的增加,使得中间件(middleware)技术应运而生。中间件是位于上层应用和下层服务之间,提供更简单可靠和增值服务的通用服务。中间件技术是IT界研究的热点之一。中间件一般可以分为事务处理中间件、分布式面向对象中间件和消息中间件。当应用程序需要在不同的网络硬件平台、不同的操作系统乃至不同的网络协议上通信时,适合选用消息中间件(MOM),因为此时应用程序对传送的要求,是所传消息内容的可靠性和可恢复性,而不要

求消息即时即刻传送到对方^[2]。

2 消息中间件技术^[1,2,9,10]

消息中间件又叫面向消息的中间件 MOM(Message Oriented Middleware),指的是利用高效可靠的消息队列传递机制进行平台无关的数据交流,并基于数据通信来进行分布式系统的集成。MOM通过提供消息传递和消息排队机制,可在分布环境下扩展进程间的通信,并支持多通讯协议、语言、应用程序、硬件和软件平台。目前流行的 MOM 中间件产品有 IBM 的 MQSeries^[22]、BEA 的 MessageQ^[23]等。消息传递和排队技术三个主要特点是:通讯程序可在不同的时间运行、对应用程序的结构没有约束和程序与网络复杂性相隔离。MOM 中两个重要的概念是消息和队列。

消息是对使用它的应用程序而言具有特殊含义的字符串。消息一般分为两部分:消息描述符和应用程序数据。应用程序数据的内容和结构由使用它们的应用程序来定义。消息描述符标示消息,并包含了消息的其它控制信息或者属性,如创建消息的日期和时间、消息类型以及发出消息的应用程序为消息所分配的优先级。消息有很多属性,其中重要的属性是:永久性、优先级、类型和消息长度。

在物理概念上,一个队列就是一种类型的列表,用来存储消息直到它们被应用程序取走。队列独立于使用它们的应用程序而存在,队列可以是以下几类:临时队列:保存在主存储

器中;永久队列:保存在磁盘或者类似的辅助存储器中;混合队列:同时保存在上述两中介质中。

3 排队论技术^[4]

排队论是运筹学的一个分支,它又叫做随机服务系统理论。它的研究目的是要回答如何改进服务机构或组织的被服务对象,使得某种指标达到最优的问题。比如一个港口应该有多少个码头,一个工厂应该有多少维修人员等。排队论最初是在二十世纪初由丹麦工程师爱尔郎关于电话交换机的效率研究开始的,在第二次世界大战中为了对飞机场跑道的容纳量进行估算,它得到了进一步的发展,其相应的学科如可靠性理论等也都发展起来。

因为排队现象是一个随机现象,因此在研究排队现象的时候,主要以研究随机现象的概率论作为主要工具,此外,还有微分和微分方程。排队论把它所要研究的对象形象地描述为顾客来到服务台前要求接待。如果服务台被其它顾客占用,那么就要排队。另一方面,服务台也时而空闲、时而忙碌,这就需要通过数学方法求得顾客的等待时间、排队长度等的概率分布。依据系统中服务台的服务员数量,排队论服务模型可分为单服务员模型和多服务员模型。其中单服务员模型使用较为普遍。本文介绍的 MOM 排队论模型是依据单服务员模型建立起来的。

3.1 单服务员排队模型

图1描绘了最简单的排队系统。这个系统的中心元素是一个服务员,它负责为顾客提供某种服务。从某种顾客群体来到顾客到达这个系统要求服务时,若服务员是空闲的,该顾客就立即得到服务,否则后者就加入到等待线上(也叫加入到队

^{*} 本文得到四川省重点科技攻关项目“智能交通系统”资金资助。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。秦志光 博士,教授,主要研究方向:开放系统与安全技术,ITS 技术。刘锦德 媒体技术。

博士研究生,主要研究方向:开放系统与中间件技术,ITS 技术。秦志光 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。

列中)。当服务员服务完一个顾客时,该顾客就离开系统。如果等待线上有顾客,那么下一个顾客立即被交给服务员。

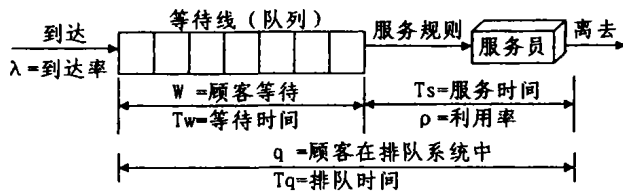


图1 排队系统的结构与单服务员队列的参数

3.2 单服务员排队模型的数学关系^[5,6]

表1列出了本文将要用到的符号参数及其含义。

表1 符号及其含义

符号及其含义
λ =每秒到达的顾客平均数
T_s =每次到达的平均服务时间
ρ =利用率
Q =系统中的顾客数(等待的及正在被服务的)
q =系统中的平均顾客数(等待的及正在被服务的)
T_q =顾客在系统中花费的平均时间
w =等待服务的顾客平均数
T_w =顾客等待服务的平均时间
N =服务员的个数
$M_x(r) = x$ 小于 $M_x(r)$ 的百分比是 r

3.2.1 单服务员排队模型的基本数学关系式 在一些简化的假设条件下,所得到的结果对于规划和设计系统而言是足够精确的。然而有些关系式对一般情况都是成立的,这些基本的单服务员排队模型的数学关系式如表2所示。

表2 单服务员模型中的基本数学关系式

一般关系	单服务员
$Q = \lambda T_q$	$\rho = \lambda T_s$
$W = \lambda T_w$	$q = w + \rho$
$T_q = T_w + T_s$	

3.2.2 单服务员排队模型的数学公式^[5,6] 为了总结制定排队论分析模型时所作的重要假设,现在已制定了一套方便的记号。这种记法就是 $X/Y/N$, 这里的 X 指的是到达间隔时间的分布, Y 指的是服务时间的分布,而 N 指的是服务员的个数。最常见的分布分别表示为: G —一般独立到达或服务时间; M —负指数分布; D —确定性到达或定长服务。

在排队论中,一个非常重要的假设是到达率服从泊松分布,没有这个假设,大部分排队论分析几乎是不可能的。因此, $M/M/1$ 指的就是泊松到达和负指数服务时间的单服务员排队。表3列出了单服务员排队模型的一些方程式。

表3 单服务员排队模型的数学公式

(a)一般服务时间(M/G/1)	
假设:	
1 泊松到达	
2 服务规则对于服务时间的长短不作区分	
3 没有顾客从队列中丢弃	
$A = \frac{1}{2} [1 + (\frac{\sigma_{T_s}}{T_s})^2]$	$q = \rho + \frac{\rho^2 A}{1 - \rho}$
$T_q = T_s + \frac{\rho T_s A}{1 - \rho}$	$T_w = \frac{\rho T_s A}{1 - \rho}$

这些数学公式的假设是到达率服从泊松分布,而服务时间是一般的。由于使用了比例系数 A , 使某些关键输出变量表达式简明易懂。注意比例系数中关键的因素是服务时间的标准差与均值之比。有两种特殊情况值得注意。当标准差等于均值时,服务时间的分布是指数的 ($M/M/1$), 这是最简单的情况,也是最容易计算结果的情形。另一种特殊的情形是标准差等于零,这等于假定是定长服务时间 ($M/D/1$)。

4 消息队列中间件中队列的排队论模型^[1]

利用单服务员排队模型的数学关系,可以建立消息队列中间件队列的排队论模型:

·计算模型: $M/M/1$

·已知:到达率 λ , 平均服务时间 T_s , 每一消息的平均大小 B_s (字节)

·求: r 百分比时间中系统需要的队列大小 M_s (字节)

利用单服务员排队模型 ($M/M/1$) 的基本数学关系公式,可以得到:

$$\rho = \lambda T_s, \quad T_q = \frac{T_s}{(1 - \rho)}, \quad q = \frac{\rho}{(1 - \rho)}$$

为了得到百分比时间 r 下需要的缓冲区的数值,利用公式:

$$\Pr[Q=N] = (1 - \rho) \rho^N$$

将公式写成累加的形式:

$$\frac{r}{100} = \sum_{i=0}^{m_q(r)} (1 - \rho) \rho^i = 1 - \rho^{1 - m_q(r)}$$

其中: $m_q(r)$ = 需要的缓冲区数

两边求对数得到:

$$m_q(r) = \frac{\ln(1 - \frac{r}{100})}{\ln \rho} - 1$$

通过这个公式计算出来的是需要的分组缓冲区数,如果知道每一个缓冲区的大小 B_s , 可以计算消息队列中间件中需要的共享内存的大小 M_s :

$$M_s = M_q(r) \times B_s = \frac{\ln(1 - \frac{r}{100})}{\ln(\lambda T_s)} \times B_s$$

其中单位是字节。

5 消息队列排队论模型在 ITS-MQ 中间件中的应用示例^[1,9,10]

智能交通系统(ITS)的收费系统是采用消息队列的典型应用。在该收费系统中,收费站的收费数据和其它控制信息,通过消息队列中间件(ITS-MQ 中间件),传送到收费站服务器和收费中心服务器。根据对系统网络结构分析和实验结果,系统的瓶颈是收费中心服务器(因为它要为所有的收费站服务,而收费站服务器只为自己管辖的收费站服务)。因此,如何设计收费中心服务器的消息队列是关键点之一。我们在设计过程中,是参照上述消息队列的排队论模型来进行的。

根据系统设计、系统主机性能和考虑到系统今后的扩展和升级,设计条件是:系统最多可向100个收费站提供服务,每个站最多可以有车道10个,每个车道平均每隔3秒向中心服务器和收费站服务器发送一条消息(包括收费消息和控制),每条消息的平均大小为512字节;服务器对消息的响应时间平均估计值为0.001秒;考虑系统在百分之九十九 ($r = 0.99$) 的时间内的消息缓冲。因此,

$$\lambda = 500 \times 10 = 5000 \text{ 数据包/s}$$

$$T_s = 0.01s \quad B_s = 512bytes$$

$$M_{s1} = \left(\frac{\ln(1 - \frac{r}{100})}{\ln(\lambda T_s)} - 1 \right) \times B_s$$

$$= \left(\frac{\ln(1 - \frac{99}{100})}{\ln(\frac{100 \times 10}{3} \times 0.001)} - 1 \right) \times 512 = 1633(bytes)$$

根据计算结果,只需分配1633字节的内存空间,但是,考虑到系统的突发事件,可以乘一个保险系数 $K=3$,则:

$$M_{s2} = 1633 \times 3 = 4899(bytes)$$

另外,由于某些原因(如需要维修、系统故障等),中心服务器可能在一段时间内无法处理收费数据或者其他消息,因此消息将暂时存放在队列中,而不能被丢弃。假设系统出现问题后可以在15分钟内恢复,因此最终的共享内存大小为:

$$M_s = 4899 \times 15 \times 60 = 4.20(M) \approx 5(M)$$

计算结果:系统大约需要5M的共享物理内存空间,当然,根据实际情况,该值可以上下偏移。

我们根据该计算结果,成功地设计了 ITS-MQ 中间件的消息队列,该系统已经成功应用在实际的智能交通收费系统中,并通过了国家软件评测中心的评测(CNACL 编号: No. 0414)。表4是 ITS-MQ 运行实验测试数据。

表4 利用队列的排队论模型设计的 ITS-MQ 实验结果

收费中心服务的 收费站数(个)	消息队列平均 使用率(%)	消息丢失 平均数(条)
1	0.01	0
10	0.09	0
20	2.0	0
40	5.4	0
80	30.0	0
100	50.9	0
200	70.7	0
400	100	10

由表4可见,在服务的收费站数很少时(小于40个),ITS-MQ 的消息队列使用率很低;即使在服务的收费站数为设计条件的80%负载时,消息队列的使用率为30%;当服务的收费站为设计条件的100%时,队列使用率仍仅为50%。当超过设计条件4倍(400个收费站)时,队列使用率达到了100%,并开始出现消息丢失现象。

此外,考虑到 ITS 收费系统中往往存在突发性消息到达(比如在某一时间段内有大量的收费数据上传,这可能是由于网络故障恢复后的批量数据上传),我们还对 ITS-MQ 作了强度测试,测试条件和结果如表5所示。

表5 利用队列的排队论模型设计的 ITS-MQ 强度测试实验结果

消息数(条/分)	队列平均使用率(%)	消息丢失平均数(条)
1000	30.4	0
2000	60.0	0
4000	80.1	0
10000	100	100

表5的测试结果表明,在低强度测试条件(消息数小于1000)下,ITS-MQ 中的队列使用率较低;在中等强度测试条件(消息数小于2000),队列使用率显著上升,但是不会出现消息丢失的现象;当消息数远远超过设计条件(消息数大于10000)

时,队列开始饱和,并有消息丢失现象发生。

总之,表4和表5的测试结果均表明,按照我们提出了消息队列的排队论模型设计的队列,完全满足实际运行环境的要求;在设计条件下的测试结果表明,没有出现队列满导致消息丢失的现象。消息队列平均使用率在30%左右,没有空间的明显浪费。在突发情况下,队列的使用率也仅为70%左右,因而大大提高了系统的效率和节约了物理空间。此外,系统在异常条件中的抗冲击能力也能满足实际的需要。所以,可以认为,MOM 的排队论模型从理论和实际角度,均具有科学性、实用性和可行性。

6 今后的研究工作

利用排队论模型,可以解决通用客户/服务器模型下消息队列中间件的消息队列计算和设计问题。但是,作为当前研究热点之一的消息中间件技术,还需要和诸如分布式计算和 Web 技术等有机结合,才能满足应用的新要求。因此,研究分布式计算技术下的消息队列模型更具有良好的应用价值和理论价值。如何利用排队论技术建立分布式环境下的队列模型,是今后的研究重点之一。另外,建立消息队列的多服务员排队论模型,也是我们的研究工作之一(而且也正在进行当中)。

总结 消息队列中间件技术可以提供异步通信的手段,从而为众多应用所采用,设计和开发消息中间件的关键之一是如何进行消息队列的设计。排队论适合用来分析和研究随机现象,而客户/服务器模型下的许多应用都有可能符合随机模型(比如网络环境下的智能交通收费系统)。在单服务员模型下的排队论系统中,通过泊松分布可以建立该模型下的各种基本计算公式,充分利用这些公式,可以设计消息中间件的消息队列。我们采用这种技术,建立了消息队列中间件的排队论模型,并根据该模型设计了用于智能交通系统的 ITS-MQ 消息队列中间件。实际运行结果表明,依据该模型设计的队列模型,基本符合实际的需要,且具有很好的抗冲击能力,从而有力地证明了该排队论模型的正确性和可行性。

参考文献

- 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]. 电子科技大学:电子科技大学图书馆, 2000. 12
- 郭乐深. IP 网络环境中支持 QoS 机制的中间件系统的研究:[电子科技大学博士论文]. 电子科技大学:电子科技大学图书馆, 2001. 8
- Lewis R. Advanced Message Application with MSMQ and MQSeries, QUE. 1999, 11
- Baccelli F, Bremaud P. Elements of Queuing Theory, 1994
- Palmer M. 局域网与广域网的设计与实现. 北京:清华大学出版社, 2000, 8
- Stallings W. 高速网络 TCP/IP 和 ATM 的设计原理. 北京:清华大学出版社, 1999
- Ottom R, Patric P, Roy M. CORBA 教程. 北京:清华大学出版社, 1999, 10
- COM 与 DCOM 从入门到精通. 北京:电子工业出版社, 2000, 4
- 周世杰, 刘锦德, 秦志光. 消息队列技术研究. 计算机科学, 2001(待发)
- 周世杰, 刘锦德, 秦志光. 消息队列中间件在高速公路收费系统中的设计与应用. 计算机应用, 2001(待发)
- 周世杰, 秦志光. 文件传输协议分析及应用. 计算机应用, 2001增刊
- 秦志光, 刘锦德. ODP 环境中全局安全系统内的安全域之间的连接. 电子科技大学学报, 1995, 24(5): 520~523
- Atkins D. Internet Security Professional Reference. 机械工业出版社, 1998
- 苏森, 唐雪飞, 刘锦德. 面向对象互操作技术. 电子科技大学学报, 1998, 27(1): 990~994
- 孙家昶, 等. 网络并行计算与分布式编程环境. 科学出版社, 1996
- 刘健. 分布式计算机系统. 人民邮电出版社
- 鞠九滨. 分布式计算系统. 高等教育出版社
- 周明天, 汪文勇. TCP/IP 网络原理与技术. 清华大学出版社

基于反射中间件的异构分布式实时调度系统

A Heterogeneous Distributed Real-Time Scheduling System Based on Reflective Middleware

杨仕平 熊光泽 刘锦德

(电子科技大学计算机科学与工程学院 成都610054)

Abstract In order to help manage the complexity, heterogeneity and dynamic inherent in distributed real-time systems, middleware of next generation should deal with changing environments and different client requirements. it is very necessary to resolve interoperability between heterogeneous schedulers. Therefore, this promotes the research of a heterogeneous distributed real-time scheduling system based on reflective middleware. The features of reflection technology and reflective middleware are described in detail first. As a result, proposed reflective middleware is used to resolve interoperability between heterogeneous schedulers in distributed real time system. The high performance, adaptability and feasibility of this middleware platform used to resolve interoperability between heterogeneous schedulers are proved in details by a concrete example. Future trends of research in this field are listed at the end.

Keywords Metaobject, Reflective middleware, Distributed system, Real-time scheduling, Heterogeneity

1. 引言

随着计算机网络技术的发展,分布式实时应用的领域也越来越多,其典型的商业化产品有分布式虚拟现实系统、分布式多媒体协作系统、多选手在线网络游戏等。这些系统往往具有复杂的、严格的 QoS 需求,如:时间延迟、抖动、可靠性需求等。为了能更好地实现这些系统,其关键问题是要解决产生于不同终端系统的竞争者(Competitor)(如分布式线程、对 CORBA 对象的操作等)之间的灵活通信及很好地保持其端到端的实时 QoS 需求,解决这些问题的关键是使分布式实时系统同时具有开放性(使系统各组件能灵活地连接及互操作而不需要预先静态配制)和可靠性(使系统能严格地保持其端到端的实时 QoS 需求,如时间特性和资源限制等)。

作为分布式实时系统中一个较难解决的问题——异构调度器间的互操作,引起了许多学者的关注。如其中一个终端系统使用的是最早死限优先(EDF)调度器,而另一个终端系统使用的是最小松弛度优先(LLF)调度器,当产生于前一终端系统的竞争者(分布式线程)迁移到后一终端系统中时,由于此时竞争者的属性集合是当前调度器特征集合的子集,导致的问题是竞争者的属性(Property)(如死限、重要性等)如何自动适应当前最小松弛度优先(LLF)调度器的特征(如死限、重要性、松弛度等)要求?相反的迁移也将导致类似的问题。目前关于异构调度器间互操作问题的解决办法虽然也采用了中间件的方式(如 RT-CORBA、RT-Java 及 DRT-Java 等),但其实现方式缺乏灵活性、适应性。如对 EDF 调度器与 LLF 调度器之间的互操作,RT-CORBA 则在各终端系统中

都配置了一“臃肿”的调度器,其特征集合为二者的并集(如死限、重要性、松弛度等)。而对多终端分布式实时系统,也采用了同样的思路,只不过其调度器显得更“臃肿”而已。尽管这些“臃肿”调度器的某些特征在实际中并未使用,但为了满足产生于各终端系统竞争者可调度的需要,这种缺乏灵活性、动态性及高效性的配置却是必要的。

反射(Reflection)技术由于能更好地访问(或调整)系统的内部结构和实现,因此它将成为下一代具有灵活性与自适应性中间件——反射中间件的主要实现技术。为了解决当前异构分布式实时系统中异构调度器间的互操作问题,作者进行了基于反射中间件的异构分布式实时调度系统的研究。本文首先详细论述了反射技术及反射中间件,最后提出了可解决异构调度器间互操作的反射中间件框架,并通过实例论述了该实现的可行性。

2. 反射原理及其特性

反射的概念最初由 Smith 在1982年提出^[4]。Patti Maes 首次将反射引入到了面向对象编程领域。抽象地讲,反射是系统的一种推理和作用于自身的能力。反射系统,是指这样一种系统:它提供了关于自身行为的表示,这种表示可以被检查和调整,且与它所描述的系统行为是因果相联的(causally connected)。因果相联,意味着对自表示(self-representation)的改动将立即反映在系统的实际状态和行为中(这也是使用“反射”这个词的由来),反之亦然。因此可以简单地说,反射系统是支持因果相联的自表示系统。就如在传统 OOP(Object-Oriented Program)中对象代表了现实世界中的实体一样,对

杨仕平 博士研究生,主要研究方向:实时操作系统中的防危核机制与实现。熊光泽 教授,博士生导师,研究领域:实时计算机系统及应用。刘锦德 教授,博士生导师,研究领域:开放系统、多媒体系统。

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

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

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

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

23 IBM MQSeries technical report. Available at: <http://www-4.ibm.com/software/ts/mqseries/>

24 BEA MessageQ technical report. Available at: <http://www.bea.com/product/messageq/>