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，导致的问题是该竞争者的属性（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 MQSeried technical report. Available at: http://www-4.ibm.com/sorftware/ts/mqseries/

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

象本身也可用其它对象来表现，后者又称为元对象(meta-object)。元对象的计算又称为元计算(meta-computation)，它主要用于观测和修改它们的指示物(referents，即所代表的对象)。元计算常常是通过捕获(intercept or trap)它们的指示物的正常计算、且由元对象执行的；换句话说，指示物的行为被元对象所捕获，后者执行元计算，要么替换要么封装指示物的行为。元计算实现原理如图1所示。当然，元对象自身也可由其它对象来表现，即它们是二重元对象(meta-meta-object)的指示物，以此类推。这样，一个反射系统可以是一种多层结构，组成了一个反射塔(reflective tower)。在基层的对象称为基对象(base-object)，它们对应用领域中的实体进行计算；其它层，即元层(meta-level)的对象，对其相邻低层(指示物层)的对象执行计算。基对象与元对象之间不一定是一一对应的关联：几个元对象可以共享同一个指示物，单个元对象也可有几个指示物。反射塔的相邻层之间的界面通常称为元对象协议(meta-object protocol 或 MOP)，如图1中细虚线所示。

在所有反射模型中，一个基本概念就是具体化(Reification)，如图1中的③所示。为了对相邻低层的计算执行计算，每一层维护一组支持这个计算的数据结构，即前述的因果相联的自表示，而创建这种自表示的过程就称为具体化。具体化就是将系统原本隐式的方面显式化。当然，相邻低层的哪些方面被具体化，取决于反射模型(如后所述的属性及竞争者等)。在任一种情况下，包含自表示的数据结构与系统被具体化的那些方面是因果相联的。在相邻层之间保持这种因果相联关系是反射基础设施的责任，而元对象的设计者和编程人员不必知道如何实现因果相联关系等细节。与具体化相反的一个概念是反射(Reflection)，如图1中的②所示，它负责把对元对象中的方法、属性等反射到基对象中去。

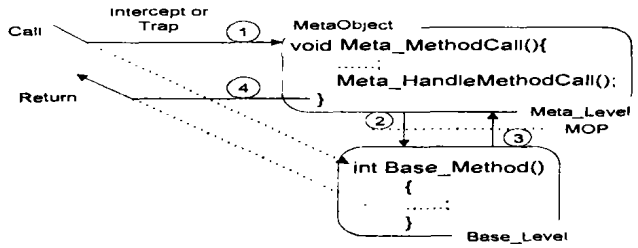

图1　元计算原理示意图

采用反射技术进行异构分布式实时系统的设计和应用程序的开发都得益于反射的基本特性：透明性、关注分离、可见性、反射粒度。

·*透明性*(Transparency)　在反射这个上下文中，透明是指位于基层的对象完全不知晓元对象的存在或工作情况。换句话说，元层被加到基层上而无需改动基对象本身。也即，反射系统在元层及基层之间执行因果相联，是以对元层编程人员和基层编程人员都透明的方式实现的，我们把这称为反射透明性(reflection transparency)。

·*关注分离*(Separation of Concerns)　反射塔的不同层关注系统的不同方面，即基层执行系统的功能，其它元层扩展(由其下分层组成的)系统的不同非功能性方面，比如：容错、并发、持久性等等。这样，利用反射就可以容易地扩展一个计算系统。将系统的不同方面(功能性属性或各种非功能性属性)委派给不同的分层。

·*可见性*(Visibility)　可见性表示元计算的范围，即哪些基层实体或基层实体的哪些方面涉及到了元计算。

·*反射粒度*(Reflection Granularity)　反射粒度指一个反射系统中，被不同元实体具体化的基层实体最小的方面。通常的粒度层次是：类、对象、方法、方法调用等。如果反射粒度是在方法这个层次，则同一对象的两个方法可以被两个不同的元实体具体化，从而表现出两种不同的元行为。小的反射粒度使软件系统有更好的灵活性和模块性，但元实体的数量也会激增，而本文所采用的反射粒度是对象，即基于对象的反射。

3. 反射中间件

反射中间件的研究是反射领域中一个新的研究分支。根据上面的阐述可知，所谓反射中间件，简单地说，是指通过适当的因果相联的自表示，而能够检查和调整其行为的中间件系统。在中间件设计中运用反射技术，至少可以获得如下好处：

·*检查系统的结构、状态和行为*　如CORBA的界面仓库和动态调用界面，可以使客户程序在运行时发现服务组件的类型，而动态地构造调用请求，以调用这些服务组件的操作。

·*灵活性和自适应性*　反射使系统更能适应新的环境，并更好地对付各种变化。这种变化往往发生在基于中间件的应用程序被部署到动态环境，如多媒体、组通信、实时和移动计算等环境时。

·*关注分离*　在系统设计中采用反射技术，可以分离功能性属性和非功能性属性。如前所述，典型的方式是，基层对象用来满足应用程序的功能性要求，而元层用于保证非功能性属性。这种方法的优点在于：首先关注分离增强了系统的可修改性。根据系统要修改部分涉及的是功能性属性还是哪种非功能性属性，可以只改动对应的分层。而且，关注分离还从两个方面提高了可重用性：一是同样的基层对象，既可以通过与一些元实体相关联而具备某些额外属性，也可独立使用；另一种形式的重用性体现在，同样的元实体，可以被重用于为不同的基层实体附加同样的非功能性属性。

4. 基于反射中间件的异构分布式实时调度系统

4.1　异构分布式实时调度系统

由第2、3节的内容可知，反射技术由于具有较强的灵活性与自适应性，本文将利用它来实现分布式实时系统中异构调度器间高效、灵活的互操作。为了方便后面的论述，本文用四元组〈C，P，S，A〉来表示异构分布式实时系统中的各终端系统，其中：

·C表示各终端系统的竞争者(Competitor)，竞争者是可调度实体的抽象表示。如：分布式系统中的分布式线程等，它们可参与共享资源(如内存、CPU、网络带宽等)的竞争。

·P表示各竞争者的属性(Property)集合，属性集合是竞争者属性值域的抽象表示。竞争者的属性如：死限(deadline)、周期(period)、松弛度(laxity)等。

·S表示可调度各竞争者的调度器(Scheduler)，调度器是各ORB终端系统调度器的抽象表示，它提供了调度竞争者的接口，同时它也测试竞争者调度的可行性。各调度器的具体实现如：最早死限优先(EDF)调度器、最小松弛度优先(LLF)调度器、速率单调调度器(RMS)等。

·A表示各ORB终端系统的适配器(Adapter)，为了适应新调度器特征集合的需要，它负责竞争者属性集合到另一属性集合(调度器的特征集合)的转换。如适配器负责把具有最早死限优先(EDF)竞争者属性集合转换到最小松弛度优先

(LLF)调度器的特征集合。

竞争者、属性、调度器与适配器四者的关系是:属性是竞争者特征的表现,如果竞争者的属性与调度器的特征集合不匹配,则竞争者负责发现一最佳适配器用于竞争者属性集合到调度器特征集合的转换,以便在新的环境下可调度,同时保证了竞争者端到端的QoS需求。

4.2 可实现分布式实时系统中异构调度器间互操作的反射中间件

由前可知,对于分布式实时系统中的任意终端系统(C,P,S,A),尽管竞争者、属性、调度器与适配器四者之间存在着相互作用关系,如果让分布式应用程序本身去实现四者之间的这些关系,一方面会使应用程序开发人员的工作量增大,另一方面则使得开发出的应用程序缺乏足够的灵活性与适应性,如调度策略的改变都会导致应用程序大量的改动。为对付分布式实时应用系统内在的复杂性和异构性,明显地简化分布式实时应用的开发,其解决办法是:在构造分布式实时应用的过程中,要充分利用以COTS分布式对象计算(DOC)为基础的反射式中间件技术。为实现分布式实时系统中异构调度器间的互操作,在中间件的ORB CORE中实现了与竞争者、属性、调度器及适配器四者相对应的元对象,即元竞争者、元属性、元调度器与元适配器。竞争者、属性、调度器及适配器四者之间的相互作用都由与之相对应的元竞争者、元属性、元调度器及元适配器来实现。图2为可实现异构调度器间互操作的反射中间件。由图2及如下论述可知,通过竞争者、属性、调度器与适配器及与四者相对应的元对象间的相互作用,灵活动态地实现了异构调度器间的互操作。而其中的传输协议元对象主要根据变化的环境灵活自适应地采用不同的实时网络传输协议,由于本论文着重讨论异构调度器间互操作的问题,故对此不作专门深入讨论。

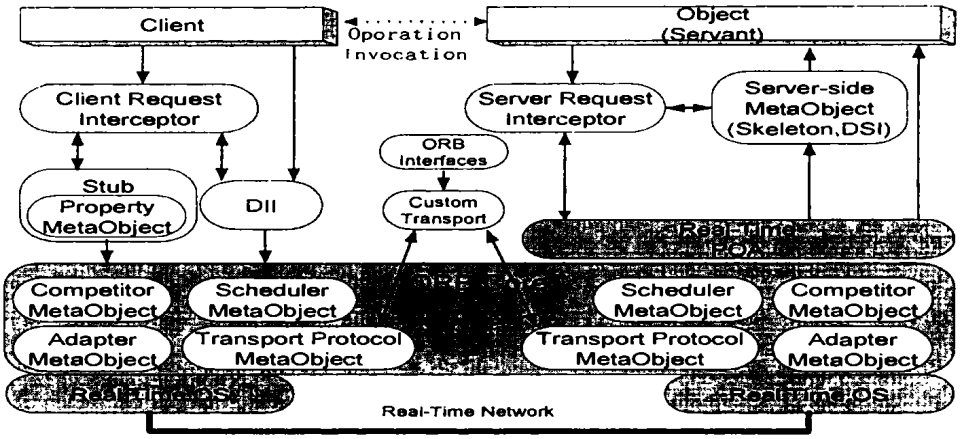

图2 可实现分布式实时系统中异构调度器间互操作的反射中间件

4.3 基于反射中间件的异构调度器间互操作的实现原理

在异构分布式实时系统中,各终端系统可能使用不同的调度器,当一终端系统中的竞争者(设为客户器方的分布式线程)迁往另一终端系统(设为服务器方)中时,竞争者的属性集合适配调度器的特征集合可采用两种方式进行:

·*在客户器方进行适配* 当客户器方竞争者的属性集合是服务器方调度器特征集合的子集时,为减小迁移过程中的网络开销,客户器方竞争者(分布式线程)的属性集合在客户方适配好以后再迁往服务器方参与调度器的调度。其原理如图3所示,当客户方有一分布式线程即将迁往服务器方时,由图2中的客户方请求拦截器(client request interceptor)捕获此请求,然后把此请求传递给桩中的属性元对象,由于在分布式线程真正迁移之前通过绑定已得到服务器方调度器的特征集合(设保存在基层属性集合中),此时桩中的属性元对象通过过程1反射、过程2具体化得到服务器方调度器的特征集合,通过过程3元属性把得到的服务器调度器特征集合传递给元竞争者,元竞争者通过过程4反射、过程5具体化得到竞争者的属性集合,根据竞争者属性集合与调度器特征集合的比较,再通过过程6找到合适的元适配器,最后元适配器再通过过程7让基层适配器作具体的适配工作。

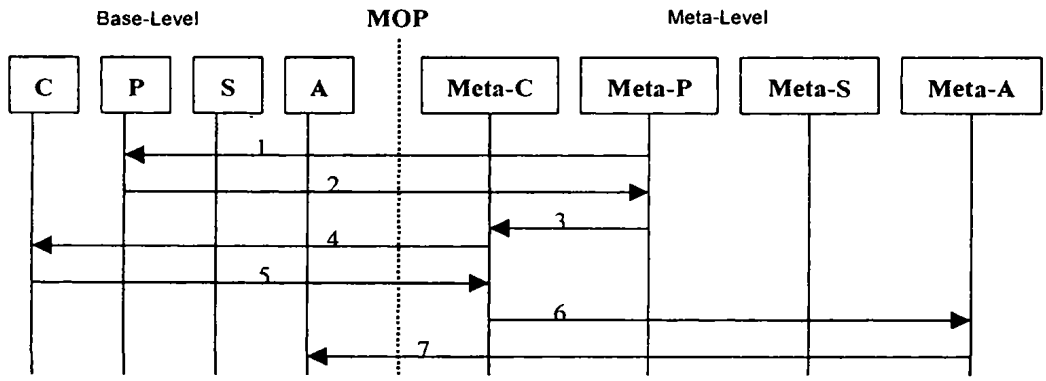

图3 客户方适配时基层对象与元层对象之间的相互作用

·在服务器方进行适配 当服务器方调度器特征集合是客户器方竞争者属性集合的子集时，同样为减小迁移过程中的网络开销，客户器方竞争者（分布式线程）迁往服务器方后才进行适配工作，进而参与调度器的调度。其原理如图4所示，此时当客户方的竞争者迁移到服务器方后，元调度器首先通过过程1反射、过程2具体化得到服务器方基层调度器的特征集合，通过过程3把结果返回给元竞争者，元竞争者再通过过程4反射、过程5具体化得到基层竞争者的属性集合，根据竞争者属性集合与调度器特征集合的比较再通过过程6找到合适的元适配器，最后元适配器再通过过程7让基层适配器作具体的适配工作。

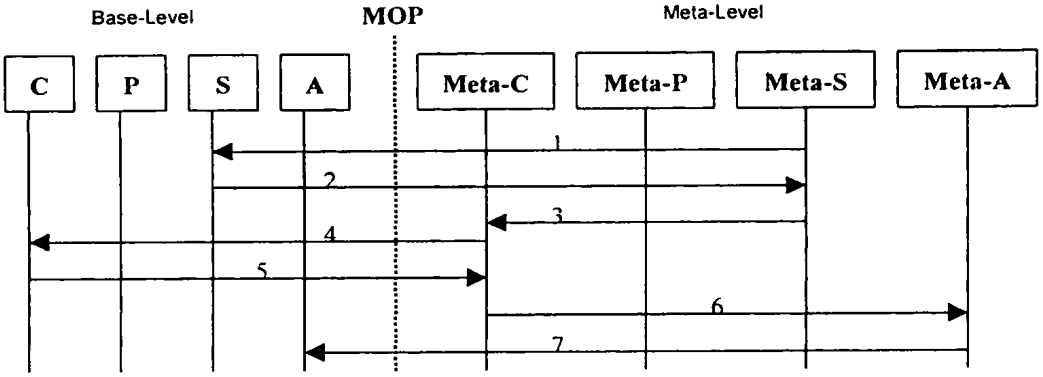

图4 服务器方适配时基层对象与元层对象之间的相互作用

通过以上对分布式实时系统中异构调度器间适配方式的讨论可知，基层属性、竞争者、适配器及调度器间的相互作用是通过与四者相对应的元对象间的相互作用来实现的。由前面对反射技术的讨论可知，对元层的改动完全不会影响基层的功能特性，正是由于这一点体现了基于反射中间件的分布式实时系统中异构调度器间互操作实现的灵活性与自适应性。例如，当一使用 RMS 调度器的新终端系统加入时，此时的客户器方分布式线程可以根据客户请求的需要而不必做过多的工作即可迁往使用 RMS 调度器的终端系统，因为此时所有的适配工作仍按前面所讨论的两种适配方式进行，且对基层竞争者（分布式线程）不可见。

5. 用反射中间件实现异构调度器间互操作实例的研究

现假设存在一基于反射中间件的分布式实时系统，其中两个 ORB 终端系统分别使用最早死限优先（EDF）调度器与最小松弛度（LLF）优先调度器，如图5所示。

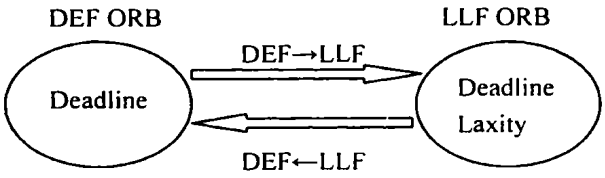

图5 异构调度器间的互操作

当产生于 EDF ORB 终端系统的分布式线程迁往 LLF ORB 终端系统时，由图5可知，此时竞争者（分布式线程）的属性集合与新的 ORB 终端系统调度器的特征集合不匹配，原因是 EDF ORB 终端系统中的竞争者属性集合是 LLF ORB 终端系统 LLF 调度器特征集合的子集，且二者只相差一个元素（松弛度 Laxity）。此时的解决办法是：为了减小网络的开销，将不使用 EDF ORB 终端系统的适配器去完成分布式线程属性集合适配 LLF 调度器特征集合的工作，而应在分布式线程迁入到 LLF ORB 终端系统后由该终端系统的适配器去完成该工作。此时 LLF ORB 终端系统的适配器将根据分布式线程的属性估算出该线程的执行时间 C，再根据该线程的死限 D 计算出其在新环境下所需的另一调度特征松弛度 Laxity（D-C），这样 LLF ORB 终端系统的适配器就完成了分布式线程属性集合与 LLF 调度器特征集合的适配工作。迁入的分布式线程将作为新的竞争者在 LLF ORB 终端系统同其它竞争者一起竞争 CPU 等资源。当然，以上所有工作将根据图4所示的序列图自动进行，而客户端应用程序对此一无所知。

而当产生于 LLF ORB 终端系统的分布式线程即将迁入 EDF ORB 终端系统时，二者间的协调又是如何进行的呢？由图5可知，此时只需将产生于 LLF ORB 终端系统的分布式线程的属性之一松弛度（Laxity）去掉即可，同样为了减小网络开销，此时的协调适配工作如图3所示在 LLF ORB 终端系统的适配器中实现。

通过该实例两种情况互操作的研究，足以证明用反射中间件来实现异构调度器间互操作的灵活性与自适应性，同时也说明了该实现的可行性。

结束语 反射技术虽然起源于90年代，以前关于它的研究主要集中于反射语言、自适应操作系统的研究，而目前关于它的研究主要集中于反射式中间件及分布式系统 QoS 的理论研究，而关于反射技术的分布式实时应用研究则很少见。本文所研究的基于反射中间件的异构分布式实时调度系统，则是反射技术应用研究的典范之一。本文通过对反射技术及反射中间件的研究，同时针对分布式实时系统中出现的异构调度器间互操作的问题，提出了用反射中间件来实现异构调度器间的互操作，并通过理论及实例的研究，证明了该实现的灵活性、自适应性与可行性。作为与反射技术相关的工作，作者正在从事用反射技术实现安全操作系统中防危核（Safety Kernel）的研究，其最终技术产品将包含在我校自主开发的国产嵌入式实时操作系统 DeltaOS 中。该产品的顺利完成将为我国家用电器、汽车电子及各种移动终端的开发带来巨大的经济效益，更主要的益处是为国防事业打下最坚实的基础。

参 考 文 献

1 杨仕平，熊光泽，陈慧. 基于 SDL 的硬实时调度可执行分析模型. 计算机科学，2002，7

（下转第39页）

DUET 客户端能够以 Applet 和 Application 两种方式运行，当以 Applet 方式运行时，远程用户的机器上只需要装有 Web 浏览器，就可下载至本地并在浏览器环境中使用；以 Application 方式运行时，远程用户必须下载 jar 打包文件并且将其设置类路径 CLASSPATH。

图3是几个 DUET 客户端的界面。

4 性能分析

因为在基于 CORBA 模型的 DUET 设计方案里，比较耗费时间和内存的操作，如产生图形界面和处理用户操作等，都被从服务器端实现中剥离出来在客户端完成，这就大大降低了服务器的工作负担。而且一个系统用户运行的不同系统工具在服务器端主机内存里将共用一个 Java 虚拟机，相应的服务进程只有一个，并不需要为每个连接新建立一个服务进程的拷贝，从而在根本上解决了以前那种 Java 语言编写的系统工具集在使用时占用过多内存空间的问题，尽可能地为用户计算保留了宝贵的内存资源，提高了系统的利用效率和对用户计算的支持能力。

表1 读取目录内容的网络传输性能的测试数据

目录	文件个数	传输字节数	传输时间(ms)	排序时间(ms)	响应时间(ms)
/	47	2242	50	0	50
/data2	5	302	0	0	0
/data2/home	80	4440	90	30	160
/data2/home/zhchen	59	2834	80	20	110
/bin	421	22384	600	200	930

同时，客户端和服务器端间的网络数据流量被降低到最小，使得多个用户在同时使用系统工具时不至于造成系统网络负担过重。例如在 RFB 远程文件浏览器中，文件的拷贝、删除、移动等操作完全在服务器上执行，服务器只返回执行该操作是成功还是失败的布尔值。从而在执行这类操作时，客户和服务器之间的通信量非常小，基本上可以忽略不计。而在读取一个目录或文件的内容时，客户和服务器之间的传输量是跟该文件或目录的大小成比例。表1是读取目录内容的网络传输性能的测试数据。

结论和未来的工作 相对于传统的高性能计算机用户环境，DUET 为用户提供了一个从本地局域网和 Internet 使用曙光3000的 GUI 接口，将显著提高曙光3000并行机的好用性。

未来我们将继续完善 DUET，使其能够包括高性能计算机几乎所有用户接口的功能。比如可以增加一个能支持编写、编译、运行和调试 pvm 和 mpi 程序的 GUI 工具，使用户能方便地在图形界面下开发并行程序；增加一个终端工具，允许用户执行在 telnet 中可以被执行的命令，这样用户可以在 DUET 中运行命令行交互的程序；在 RFB 远程文件浏览器中增加支持 put、get 文件功能，使曙光3000服务器和客户端可以传输文件，取代 ftp 等。

参 考 文 献

1 Buhlmann B, Bieri H. Supercomputing at the Desktop: An Improved Interface Using Internet Facilities. In: LNCS 1401, P. Sloot, et al. eds, Proc. of High-Performance Computing and Networking Intl. Conf. and Exhibition (HPCN Europe 1998), Springer, April 1998. 526～534

2 Sun Microsystems. Java Language - A White Paper. Sun Microsystems Computer Company, 1996

3 Orfali R, Harkey D. Client/Server Programming with Java and CORBA, Second Edition. Wiley Computer publishing, 1999

4 Bradley M. IIOP: OMG's Internet Inter-ORB Protocol: A Brief Description. OMG, May 1997

5 Westphall C M, et al. JaCoWeb Security - A CORBA Security Discretionary Prototype. CLEI Electronic Journal, 2000, 3(2)

6 Borland Corporation. VisiBroker SSL Pack 4–5 Programmer's Guide. Borland Software Corporation, June 1999

(上接第35页)

2 Deng J S Z. Scheduling Real-Time Applications in a Open System Environment. In: Proc. of the 18th IEEE Real-Time Systems Symposium (RTSS97), June 1997

3 Kalogeraki V, Melliar-Smith P, Moser L. Soft Real-Time Resource Management in CORBA Distributed Systems. In: Proc. of the Workshop on Middleware for Real-Time Systems and Services, (San Francisco, CA), IEEE, Dec. 1997

4 Smith B C. Reflection and Semantics in a Procedural Language: [Technical Report 272]. MIT Laboratory of Computer Science, 1982

5 Oliva A, Garcia I C, Buzato L E. The reflexive architecture of Guaraná: [Technical Report IC-98-14]. Institute of Computing, State University of Campinas, April 1998

6 Oliva A, Buzato L E. An Overview of MOLDS : A Meta-Object Library for Distributed Systems: [Technical Report IC-98-15]. Institute of Computing, State University of Campinas, April 1998

7 Costa F M, Blair G S. A Reflective Architecture for Middleware: Design and Implementation. In: ECOOP'99 PhDOOS Workshop, June 1999

8 Loques O, Leite J, Lobosco M, et al. Integrating Meta-Level Programming and Configuration Programming. In: Walter Cazzola, Robert J. Stroud, and Francesco Tisato, eds. Proceedings of the 1st Workshop on Object-Oriented Reflection and Software Engineer-ing(OORaSE'99), University of Milano Bicocca, Nov. 1999. 137～151

9 Kon F, Campbell R, Roman M. Design and Implementation of Runtime Reflection in Communication Middleware: the Dynamic TAO Case. In: Proc. of ICDCS'99 Workshop on Middleware, 1999

10 RM'2000. Workshop on Reflective Middleware of Middleware'2000. Available at: www. comp. lancs. au. uk/computing/RM2000. April 2000

11 Dowling J, Schafer T, Cahill V, et al. Using Reflection to Support Dynamic Adaptation of System Software: A Case Study Driven Evaluation. In Walter Cazzola, Robert J. Stroud, and Francesco Tisato, eds. Reflection and Software Engineering, Lecture Notes in Computer Science 1826, Springer-Verlag, Heidel-berg, Germany, June 2000. 171～190

12 Wang N, Schmidt D C, Kircher M, Parameswaran K. Towards a Reflective Middleware Framework for QoS-enabled CORBA Component Model Applications. IEEE Distributed Systems Online, 2001

13 Gill C D, Levine D L, Schmidt D C. The Design and Performance of a Real-Time CORBA Scheduling Service. Real-Time Systems, The International Journal of Time-Critical Computing Systems, special issue on Real-Time Middleware, 2001, 20