反射中间件研究综述*)

张 昕 丁晓宁
(中国科学院软件研究所 软件工程技术中心 北京 100080)

摘 要 反射是指系统可以感知外界环境变化并随之对自身实现进行相应调整。随着多媒体、实时和移动系统等对环境要求较高的应用的发展,中间件的"黑盒"模式已经不能满足这些新应用的需要,于是反射技术成为研究的热点。本文综述了反射技术的产生背景和研究现状,以及反射技术目前在中间件中的应用,介绍了相关研究项目和产品,最后对反射技术进行了总结和展望。

关键词 反射,中间件

A Surevy on Reflective Middleware

ZHANG Xin DING Xiao-Ning
(Software Engineering Technology Center, Institute of Software, The Chinese Academy of Sciences, Beijing 100080)

Abstract Reflection refers to the capability of a system to reason about and acts upon itself. As for the development of multimedia, real-time and mobile applications, the black-box philosophy of conventional middleware doesn't provide enough support for dealing with the dynamic aspects of these environment sensitive applications. Reflection becomes a hot research field. This paper first gives a retrospect to the background of reflection, then surveys some important research projects, and finally prospects the trend of this thechnology.

Keywords Reflection, Middleware

1 反射中间件的产生

中间件是构建分布式应用中的一个重要角色,位置处于操作系统和应用程序之间,负责二者的交互。目前一些主要的中间件技术包括 OMG 的 CORBA,SUN 公司基于 Java 的 J2EE,微软公司的.NET 等,它们屏蔽了网络通信、远程方法调用、命名、字节流传输、调度等与底层操作系统相关的接口细节,隐藏了底层软硬件平台的不同,增加了可移植性,便于构建复杂的分布式应用程序。总的来说,中间件的主要目标是对上层应用程序提供一个统一的编程模型,隐藏底层操作系统和硬件平台的异构性和分布性[1,2]。也就是说,中间件对上层用户而言,是一个"黑盒"模型(black box philosophy)。

传统的中间件模型对网络分布计算和分布式应用的发展起了极大的推动作用,但大都基于一体化体系结构(monolithic architecture)和黑盒抽象机制进行设计,近年来,随着一些新的应用领域,例如多媒体、实时系统、CSCW(Computer Supported Cooperative Work,计算机支持的协同工作)和移动系统等的飞速发展,"黑盒"模型这个中间件的优势却变成了劣势,有许多新的应用,如果能够了解底层环境的状态和细节并随之做相应调整的话,会取得更好的效果,因此客观上要求中间件能动态地进行变化以适应于外界环境的改变,并提供一种对应用程序可见的能动态控制和改变系统的机制。为了满足这些应用对系统不断变化的 QoS 需求,需要开放底层实现,让应用动态地感知底层内部状态,甚至替换底层实现,提供一个开放、灵活的体系结构。

传统中间件在提高开放性方面做了很多工作,如 OMG 为 CORBA 的一些支撑服务(如事务和安全性服务)提供了内部接口[6];而 Portable Object Adapter 也使 CORBA 应用系统的设计变得较为开放;RM-ODP(ISO/ITU-T reference model for open distributed processing)体系结构则通过区分对待计算和设计过程提高开放性。然而,或是由于最初设计思想的束缚,或是因为既成系统的调整存在很大的困难,这些传统的中间件的调整只是解决一些特定、局部的问题,不能满足体系结构开放性的要求。

反射(Reflection,有的地方也译为自省)是一种能力,意味着一个系统可以暴露内部的实现结构,亦即元数据给用户,用户可以在运行时通过修改元数据达到动态改变系统行为的目的;同时系统的改变也会动态反映到元数据上来[1,2]。反射的思想提出后,许多研究人员通过应用反射机制来实现运行期间可定制(customizable)、可重配置(reconfigurable)的体系结构,提高中间件的开放性和灵活性,由此产生了反射中间件(Reflective Middleware)[1~5]。

2 反射的定义和结构

反射的概念最早出现于1982年,是由 B. Smith 在其博士论文中提出的[7]。B. Smith 提出一种假设,即系统对外界提供某种解释器(interpreter),通过访问这个解释器感知到外界环境的变化,并进行相应的调整。这就是最初的反射的概念。文[2]中给出的定义是"反射指的是系统能感知到自身的运行状态并能够对运行状态进行动态的改变的能力";文[8]中也给出了一个定义"反射是实体具有的根据描述、操作和处理实体面临的主要问题域的相同方式描述、操作和处理实体自身的一种能力";而文[9]给出的定义则更为详细,"计算系统用来感知世界的一部分并基于此进行一些动作,计算系统与此部分因果连接(Causally Connected),即一方的改变将直接影响另一方。元系统(meta-system)是指将另一个计算系统当成

*)本文研究得到国家863高科技发展计划资助项目(编号2001AA113010,2003AA115440);国家重点基础研究发展规划973资助项目(编号2002CB312005)的资助。张 昕 博士生,主要研究领域为网络分布计算和软件工程;丁晓宁 博士生,主要研究领域为网络分布计算和软件工程。

领域的计算系统，作为领域的计算系统称为目标系统。反射是指计算系统能感知并操控自身的能力，而反射系统是指以自身为目标系统的元系统，反射系统对自身进行的感知和操控称为反射计算"。

上述观点对于反射的观点及定义基本一致，并且从这些定义可以看出，反射主要特点表现为因果连接和自表示(Causally connected & Self-representation，CCSR)[2,9]。CCSR的定义最早是由 Vrije Universiteit Brussel 大学的 P. Maes 在其博士论文[9]中提出的，指的是一个反射系统会对环境进行检查并做适应性改变，并对外提供自身行为的表示，其对外行为的表示和所描述的底层行为是有因果连接的。具体而言，自表示指中间件具体实现的内部结构和外在表示存在明确的对应关系，并且内部结构可以由外在表示来进行维护和控制，因此自表示也被称为自感知(Self-aware)的；因果连接指系统的底层实现和外在表示存在一种因果关系，即内部实现的改变会导致外在表示的变化，反之亦然。

从反射的定义以及特点可以看出来反射系统应该具备以下的结构：一部分针对于应用程序进行感知和操控，另一部分对系统本身进行感知和操控。前者形成基层实体，后者形成元层实体，然后两部分实体间形成因果连接，从而实现对系统的监控和调整，达到反射的目的。其结构如图1所示。

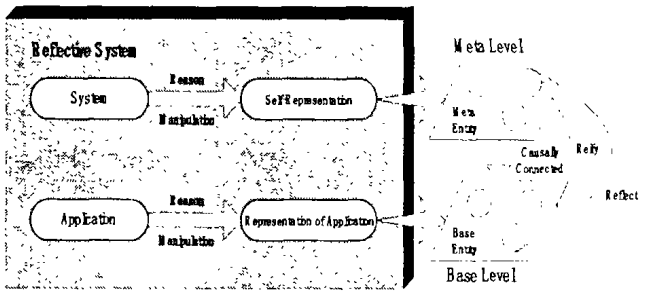

图1 反射系统的结构

在反射系统体系的组织上，有着不同的反射方式划分。一种划分是结构反射(Structural reflection)与行为反射(Behavior reflection)，前者元层只是静态的反射，即底层实体的依赖关系可以被元层实体监测和操纵，而后者能动态反射当前正在进行的活动，即底层实体的交互可以被元层实体监测和操纵，目前反射中间件大多同时支持这两种方式的反射。另一种方式划分则是过程反射(Procedural reflection)和声明反射(Declarative reflection)，前者客户端需要显式调用方法来改变系统策略，后者采用声明和修改声明的方式来改变。

3 研究情况

反射的概念自提出后，引起了不少研究人员的重视，并进行了很多工作。最初，其应用仅仅局限于程序语言的范围[10~12]。B. Smith 在文[7]中将反射引入 Lisp 语言，开发出 2-Lisp 语言；P. Maes 则在文[9]中将反射引入面向对象编程语言，开发出反射式面向对象语言3-KRS。早期的反射式编程语言还包括 CLOS[13]、ObjVlisp[14]、Smalltalk 等，近年一些新的编程语言如 Java[15]和 C#[16]也引入了反射性，而某些研究还通过对编译器进行修改进一步增强语言的反射能力，如 ReflectiveJava[17]、OpenJava[18]和 MetaJava[19]等。将反射应用于编程语言中的主要目的是为了显式地操控程序的结构和行为。目前的一些分布式对象平台，如.Net，J2EE 等，本身也提供了增加反射特性的相关接口，但是这里的反射基本上还是语言级别的，可以认为只是较完善的 RTTI，如运行期判别一个对象的类，方法类型，动态生成一个制定类的实例等，没有涉及到更改底层平台的结构和行为。

之后，反射逐渐被运用到操作系统领域中，其目的是可进行定制以适应不同硬件环境[20]。典型的反射式操作系统有 Apertos[21]，MetaOS[22]和2K[23]等。

然后反射被逐渐应用于分布式系统[24~26]。反射式分布系统的研究和反射中间件的研究比较接近，目的都是为了实现分布应用支撑平台的监控与调整，区别在于前者针对于特定系统，后者一般符合某个通用标准，因此反射在分布系统中的应用对于反射中间件的研究和发展起了重要的推动作用。

近年来，反射在中间件领域的应用得到了很大发展。文[27]尝试了将反射引入中间件以调整远程调用，随后，G. S. Blair 在文[1]中详细阐述了反射与中间件的结合，并且提出了反射是下一代中间件的主要特征。此后，反射得到了越来越多研究人员的重视，成为中间件研究领域的一大热点。由于组件在装载和替换等方面较方便，目前的反射中间件本身大多是基于组件风格的，可替换组件通常是网络协议，安全策略，加密算法等，而且主要集中在分布式对象平台上，如各种应用反射机制的 CORBA、COM、ORB、RMI 研制等，它们大多数使用面向对象的方法提供反射功能，从而提高系统的开放性。较著名的有 DynamicTAO[28]，OpenORB[29]，Flexinet[30]，OpenCORBA[31]，Quarterware[32]，mChaRM[33]等。

3.1 DynamicTAO

Dynamic TAO 由 UIUC 开发，是在现有的名为 TAO 的 ORB[34]基础之上扩展的，它允许对 ORB 内部引擎和运行的程序进行动态重配置。DynamicTAO 的开发开始于1998年，已于2000年结束，LegORB 和 UIC-CORBA 是它的后继者。在 TAO 基础上增加的对 ORB 内部引擎进行检查和重配置的功能使得 DynamicTAO 具有了反射特性。

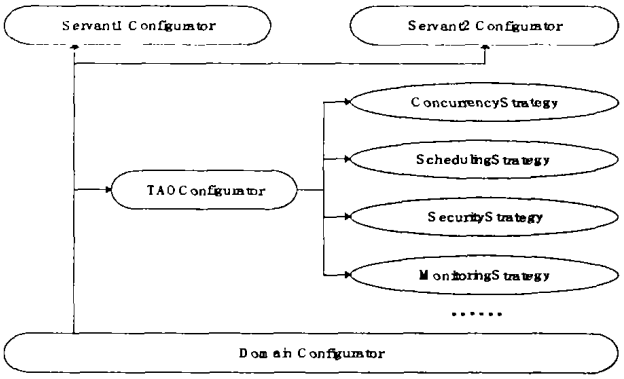

图2 DynamicTAO 的反射结构

DynamicTAO 包含两个主要特点：

·运行期间，不用重启就可以重新配置和替换一个底层的 ORB 组件。

·运行期间，上载 ORB 新策略和服务，代码采用 DLL 或 Java Byte Code。

在 DynamicTAO 中使用一系列的组件配置器(ComponentConfigurator)管理组件之间的依赖关系，如图1所示。组件配置器是一些 C++对象，当替换一个组件的请求到达时，用户可用组件配置器来检查此组件和其他组件之间的动态依赖关系，因此 DynamicTAO 支持组件的动态重配置。DynamicTAO 对外提供一组元接口(meta interface)，用于系统运行时装载卸载模块，以及检查和改变 ORB 状态的配置。DynamicTAO 将资源管理的任务分给可动态装载的组件，并使用

DSRT(Dynamic Soft Real Time Scheduler)[35]来对实时要求较高的应用程序提供 QoS 保证。

3.2 OpenORB

OpenORB 由 Lancester University 开发，其开发目的是成为具有高度可配置以及动态重配置的中间件平台，为动态要求较高的程序提供支持。OpenORB 采用一种与语言无关的反射体系结构，使用对象图描述复合组件之间的关系，支持结构性、行为性反射。

Open ORB 的可配置的特性是通过组件框架的使用来实现的，动态可重配置性则是通过反射的扩展来实现的。系统分为底层(Base-level)和无层(Meta-level)，底层由实现中间件基本服务的组件组成。元层对外暴露底层的实现，可以对底层实现进行检查和改变。元层和底层遵循同样的组件模型，元层的组件组成了 CCSR 的平台，并和底层的组件相对应。每个底层的组件都有自己的元层与其对应，元层构成一个元空间(Meta-space)，每个对象或接口均有一个元空间(meta-space)。

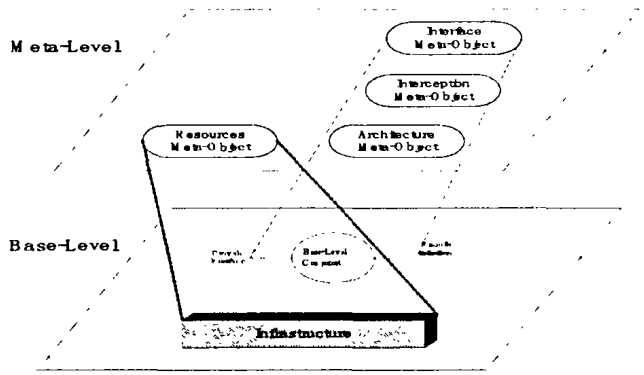

图3 Open ORB 的反射结构

OpenORB 注重元模型的概念，元模型分为组件相关的和平台相关的。元空间被划分为四种模型，每个模型提供了不同的底层实现，即 Interface、Architecture、Interception 和 Resource，其结构如图3所示。其中 Interface、Interception 和 Architecture 元模型属于组件相关的，而 Resource 元模型属于平台相关的。Interface 和 Architecture 元空间模型支持反射，前者负责组件的外部表示，后者负责内部组件的实现。

3.3 OpenCORBA

OpenCORBA 由法国 Eernationacole des Mines de Nantes 大学开发，是基于 CORBA 的一种加入了反射特性的实现，它采用 NeoClasstalk 开发，这是一种类似于 Smalltalk，具有反射性质的语言。OpenCORBA 基于元类(meta-classes)的概念，实现了对象的类定义，以及类的元类的动态替换，通过替换 COBRA 的元类提供反射功能。

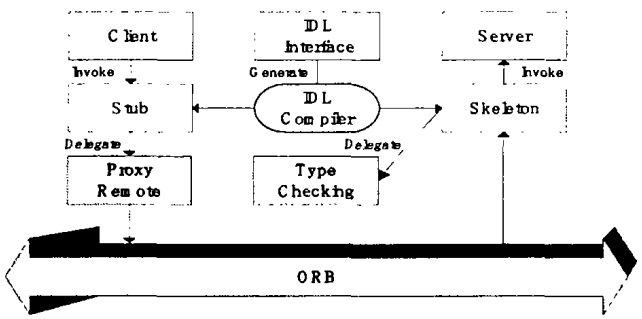

图4 OpenCORBA 的反射结构

如图4所示，Stub 由 IDL 编译器自动生成，是服务器对象在本地的映像，负责发送客户端调用请求以及接收应答，它是 IDL 接口的描述，不含实现代码，客户对代理的调用会转发到与其关联的元类，该元类才实际上实现 CORBA Stub 的功能。如图中所示，ProxyRemote 元类实现远程调用，TypeChecking 元类实现类型检查，用户还可以扩展其它的元类以实现不同的功能。在服务器端，ORB 对 Skeleton 的调用会转发给与其关联的元类，然后 Skeleton 将请求发送给目标对象实现。用户通过替换或者增加元类可以改变服务器端的行为。

3.4 mChaRM

mChaRM 是由 Università degli Studi di Milano 的 Cazzola 等提出和开发的一个面向消息和通信的反射式 RMI。Cazzola 在文[33]中认为，消息是中间件中反射的重点，应该独立于客户和服务器进行监测和行为调整，消息的反射应知道整体视图以考察各个消息间的关系。mChaRM 的结构如图5所示。

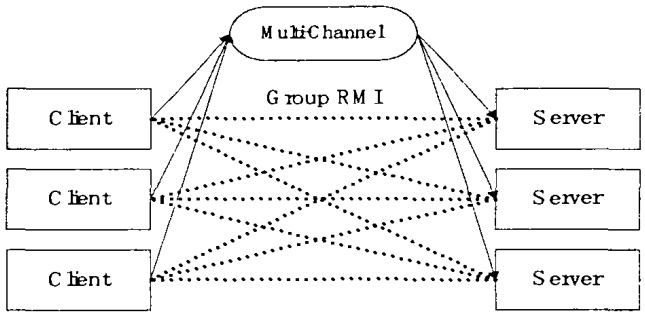

图5 mChaRM 的反射结构

mChaRM 实现了发布/订阅(publish/subscribe)通信方式，客户向某主题发送消息，消息由 Multi-Channel 转发到同主题的所有订阅者。Multi-Channel 可以监测消息，并根据接收到的消息进行相应的行为调整，而 Multi-Channel 负责接收和转发同主题的所有消息，因此具有消息的整体视图。

3.5 FlexiNet

FlexiNet 由英国的 ANSA 开发，是一个针对移动 Java 环境的可配置 ORB 开发平台，主要着眼于中间件配置/重配置以及应用部署等问题，它通过反射机制提供了一种基于组件的白盒复用方式。

FlexiNet 定义了一个开放的分布式绑定(binding)框架，并用模块的方式进行组织，它提供一个元对象协议(meta-object protocol)，允许对模块进行重配置，以改变这些绑定的属性，从而达到动态和自适应的目的。

3.6 Quarterware

Quarterware 是一个具有反射特性的中间件平台，支持包括 CORBA、Java RMI 和 MPI 在内的多种中间件标准。它使用组件框架，各种 ORB 都是用组件的形式实现，用户可以通过一个反射接口将定制组件插入到组件框架中，以增强系统的开放性。

3.7 其他

以上的反射中间件都是采用面向对象的方法，也有一些研究人员使用 AOP(aspect-oriented programming)来构建中间件元层体系结构[36]。

文[37，38]提出了一个应用反射的事务监控器的框架，可以使用模块化的方法，在传统的事务监控器上实现对扩展事务模型的支持。这个框架中引入了适配器(Adapter)的概

念，适配器是附加模块，用于在传统事务监控器中增加反射特性。适配器分为几个部分，各个部分和底层事务监控器的不同功能模块相对应，如事务管理器，日志管理器和锁控制器等。适配器能控制和改变底层事务监控器各相应功能模块的行为，同时适配器对上层应用程序提供元接口(meta interface)，即 MOP 的编程接口，因此用户通过元接口就能实现对系统底层实现和行为方式的改变。此框架已经使用在 Encina 事务监控器中，并实现了对 Split/Join 事务模型的支持。但是使用这种方案需要了解底层事务监控器的所有细节以编写相应的适配器，并对外提供 MOP 的编程接口，工作量比较大，且通用性不够。

值得指出的是，反射仅仅是一个手段，目的是允许运行期间动态修改底层的结构和行为。文[39] 基于 COM 进行扩展并提出了一种名为 COMERA 的框架，通过此框架，用户可以运行时动态修改通讯中间件的某些方面，因此能根据应用的需要对中间件进行定制，没有使用反射也同样达到了以上目标。

结束语 在构建分布式应用中，中间件起着极其关键的作用，目前的一些主流中间件存在开放性和灵活性不足等问题，尽管传统中间件已在开放的体系结构方面做了许多改进工作，但受限于原来的架构，所做的调整十分有限。面对传统中间件存在的问题，很多学者提出利用反射作为手段来解决，并随之进行了一系列的研究工作，提出和实现了一些反射中间件的模型，然而由于反射在中间件中的研究开始不久，现有工作在理论和实践方面都还不够完善。主要表现在以下几方面：1)目前的研究大多针对于特定产品，实现反射特性所采用的方法都不相同，反射的粒度和范围也不一样，缺乏一个通用的、可扩展的结构或框架。2)就目前的反射式中间件产品而言，如 DynamicTAO 等，大都基于已有的产品进行扩展，力图加入反射的特性，这样做能最大限度地利用原有代码，但是受限于以前的体系结构，所做的扩展有限，造成反射能力不够。3)要利用系统的反射特性，用户需要对系统反射性的底层实现比较了解，这样提高了用户的使用难度，降低了软件的易用性。

目前，对于反射中间件的研究范围主要包括系统完整性、性能、数学模型、形式化验证、体系结构、反射系统理论、应用以及反射中间件标准等方面。

尽管目前反射中间件的理论和实践基础仍不够完善，但是作为下一代中间件最主要特征之一，反射已经成为中间件技术的一大热点。如何应用反射机制，结合已有的研究工作，解决存在的问题，探索中间件的多阶动态可重配体系结构，必将具有广泛的前景。

参 考 文 献

1 Blair G S, Coulson G, Robin P, Papathomas M. An Architecture for Next Generation Middleware. In: Proc. of Middleware '98, The Lake District, England, Sep. 1998. 196～206

2 Coulson G. What is Reflective Middleware. IEEE Distributed Systems Online, 2001,2(8). http://dsonline.computer.org/middleware/RMarticle1.htm

3 Kon F,Costa F, Blair G,Campbell R H. The Case for Reflective Middleware. In Communication of ACM, 2002,45(6):33～38

4 Parlavantzas N, Coulson G, Clarke M, Blair G. Towards a reflective component-based middleware architecture. In:Cazzola W, ed. On-Line Proc. of ECOOP 2000 Workshop on Reflection and Metalevel Architectures, 2000. http://citeseer.nj.nec.com/331827.html

5 Duran C F,Parlavantzas H, Saikoski N, Blair G S,Coulson G. The Role of Reflective Middleware in Supporting the Engineering of Dynamic Applications. Lecture Notesin Computer Science 1826, Springer-Verlag, 2000

6 OMG, The object management group, Home page. http://www.omg.org, 2002

7 Smith B C. Procedural Reflection in Programming Languages: [PhD Thesis]. MIT, 1982

8 Ibrahim M H. Report of the First Workshop on Reflection and Metalevel Architectures in Object-Oriented Programming. OOPSLA/ECOOP'90, Ottawa, Canada,Oct. 1990

9 Maes P. Computational Reflection: [Ph. D. Thesis]. Technical Report 87-2, Vrije Universiteit Brussel, Belgium, 1987

10 Watanabe T, Yonezawa A. Reflection in an Object-Oriented Concurrent Language. In: Proc. of OOPSLA'88, ACM SIGPLAN Notices, Vol. 23, ACM Press, 1988. 306～315

11 Kiczales G, des Rivières J, Bobrow D G. The Art of the Metaobject Protocol, MIT Press, 1991

12 Agha G. The Structure and Semantics of Actor Languages. Lecture Notes in Computer Science, 489,Springer-Verlag, 1991. 1～59

13 Gabriel R P,White J L,Bobrow D G. CLOS: Integrating Object-Oriented and Functional Programming. Communications of the ACM, 1991,34(9):28～38

14 Cointe P. Metaclasses are First Class: the ObjVlisp Model. In: Proc. of ACM Conf. on Object-Oriented Programming, Systems, Languages and Applications (OOPSLA' 87), Orlando, FL USA, Oct. 1987. 156～167

15 Sun Microsystems. The Java Language Specification Second Edition. 2000. http://java.sun.com/docs/books/jls/second_edition/html/j.title.doc.html.

16 Obasanjo D. A Comparison of Microsoft's C# Programming Language to Sun Microsystems' Java Programming Language. 2001. http://www.25hoursaday.com/CsharpVsJava.html.

17 Wu Zhixue. Reflective Java and A Reflective-Component-Based Transaction Architecture. In: Proc. of OOPSLA'98 Workshop on Reflective Programming in C++ and Java, 1998

18 Tatsubori M, Chiba S, Itano K, Killijian M-O. OpenJava: A Class-Based Macro System for Java. In Reflection and Software Engineering, Lecture Notes in Computer Science, Vol. 1826, Heidelberg, Germany, pp. 117～133

19 Golm M,Kleinoder J. MetaJava - A Platform for Adaptable Operating-System Mechanisms. In: ECOOP' 97 Workshop on Object-Orientation and Operating Systems, Finland, LNCS 1357, Springer-Verlag, 1997. 507～514

20 Kiczales G, Lamping J, Maeda C, Keppel D,McNamee D. The Need for Customizable Operating Systems. In:Proc. of 4th Workshop on Workstation Operating Systems, Oct. 1993. 165～169

21 Yokote Y. The Apertos Reflective Operating System: The Concept and Its Implementation. In: Proc. of OOPSLA'92, ACM SIGPLAN Notices, Vol. 28, ACM Press, 1992. 414～434

22 Horie M, Pang J C, Manning E G, Shoja G C. Designing Meta-Interfaces for Object-Oriented Operating Systems. IEEE Pacific Rim Conf. on Communications, Computers and Signal Processing, Aug. 1997. 989～992

23 Kon F,Singhai A,Campbell R H,Carvalho D, Moore R,Balesteros F. 2K: A Reflective, Component-Based Operating System for Rapidly Changing Environments. ECOOP' 98 Workshop on Reflective Object-Oriented Programming and Systems, Brussels, Belgium, July 1998

24 McAffer J. Meta-Level Architecture Support for Distributed Objects. In:Proc. of Reflection'96, San Francisco, CA, 1996. 39～62

25 McAffer J. The CodA MOP. In:Proc. of the 8th Conf. on Object-Oriented rogramming Systems, Languages and Applications (OOPSLA'93), Workshop on Object-Oriented Reflection and Metalevel Architectures, Washington, DC, USA, ACM,1993

26 Garbinato B, Guerraoui R, Mazouni K R. Distributed Programming GARF. In: Rachid Guerraoui, Oscar Nierstrasz, Michel Riveil, eds. Object-Based Distributed Programming, LNCS 901, Springer-Verlag, 1994. 1～32

27 Ledoux T. Implementing Proxy Objects in a Reflective ORB. In: Proc. s of ECOOP'97 Workshop on CORBA: Implementation, Use and Evaluation, Jyväskylä, Finland

28 http://devius.cs.uiuc.edu/2k/dynamicTAO/architecture/

29 Blair G S,et al. The Design and Implementation of Open ORB 2. IEEE Distributed Systems Online, 2001,2(6)

30 Hayton R. FlexiNet Open ORB Framework: [Technical Report 2047.01.00]. APM Ltd., Cambridge, UK, Oct. 1997

31 Ledoux T. OpenCORBA: A Reflective Open Broker. In:Proc. of Reflection'99, number 1616 in LNCS, St. Malo, France,

(下转第192页)

查全率较低，容易遗漏相关的网页，尤其是一些包含查询项较多的长查询。于是我们进一步在Okapi搜索引擎得到的基本结果集上进行了试验，并和原来的结果作比较。我们对动态确定参数 a 进行了探索，把50个查询中的前一半拿来训练，另一半用作测试。对训练用的25个查询分成5份进行交叉检验，然后在另25个查询上做测试。并且，我们评价了不做训练(取 $a=1$)时的结果，发现性能也很好。

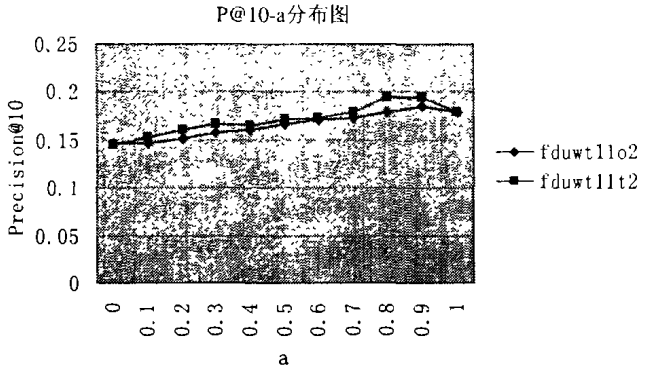

图3 参数 a 对 P@10的影响

Baseline 是搜索引擎的基本结果集，未作任何后续处理；自适应是应用交叉检验训练得到的结果。从表2中可以看出，运用这种算法可以比较明显地提高 $P@10$ 的查准率。以无参数($a=1$)为例，*fduwtllt2* 的 $P@10$ 提高了18.9%，*fduwtOkapi* 的 $P@10$ 提高了20.4%。搜索引擎的基本结果集对这种性能提高的影响不大。

表2 两组结果的P@10比较

Run	Baseline	不用参数	用参数	
		(a=1)	a=0.8	自适应
fduwtllt2	0.1510	0.1796	0.1939	0.1875
fduwtOkapi	0.2	0.2408	0.2204	0.2375

3.4.4 实验结果分析 由于 Web 本身具有的特点，通过它发布的信息极大丰富，但也包含有大量的垃圾信息。而对于用户来说，他们只希望返回的前几个网页都是非常相关的。在 Topic Distillation 任务中，采用的 $P@10$ 评价指标，也只注重返回结果前十个网页的查准率，对查全率没有什么要求。因此我们认为，Web 检索的查准率比查全率更加重要。

实验结果证明，这种方法对于 Topic Distillation 任务，效果是很明显的。相对于基本的检索结果，实验的两组结果的性能有明显的提高，尤其是提升了返回的前几个网页中的查准率。网页的标题、锚文本等结构信息对提高检索的查准率很有效，但它也依赖于基本搜索引擎的结果集，因此基本的检索结果集的查全率也是一个不能忽略的因素，不能太低。我们认为，如果语料足够大(相对于真正的 Web，18.1 GB 的语料还是很小)，我们的布尔搜索引擎得到的检索结果会比现在更好。

链接信息是非常有用的，但要和其他方法结合起来应用，对链接信息的查准率要求也比较高。如果能够准确地过滤掉网页中的无关链接，链接信息确实能有效地提高 Web 检索的性能。链接信息的结果也同样依赖于基本的检索结果集的查全率。

结论 检索的查准率对于 Web 检索的性能至关重要，因此要在不损失大量查全率的情况下，着重提高查准率。网页的结构信息，对提高 Web 检索的查准率作用很大。因此，我们还需要对网页的结构信息进一步细化，希望能系统化地处理网页中包含的结构信息。

过滤掉无关的链接，有效地应用相关的链接，有助于充分利用 Web 本身具有的特征，得到更高质量的检索结果。把网页的本身属性和链接信息结合起来，被证明确实是一条有效提高 Web 检索性能的途径。我们将对怎样把网页的结构信息和链接信息有机地结合在一起做进一步的探索。

Web 链接信息还可以应用于除 Web 信息检索以外的其他领域，如基于 Web 的文档分类、文档聚类、信息抽取、信息挖掘等。

参考文献

1 Murray B H, Moore A. Sizing the Internet. White paper, Cyveillance Inc., July 2000
2 Voorhess E M, Harman D. Overview of the Eighth Text Retrieval Conference (TREC-8). TREC 8, Gaithersburg, Maryland, Nov. 1999
3 TREC 2002 Web Track Guidelines. http://trec.nist.gov, 2002
4 Brin S, Page L. The anatomy of a large-scale hypertextual web search engine. In: Proc. of the Seventh Intl. World-Wide Web Conf. 1998
5 Kleinberg J M. Authoritative Sources in a Hyperlinked Environment. In: Proc. 9th ACM-SIAM Symposium on Discrete Algorithms, 1998
6 Weiss R, et al. Hypursuit: A hierarchical network search engine that exploits content-link hypertext clustering. In: Seventh ACM Conf. on Hypertext, March 1996
7 Lempel R, Moran S. The stochastic approach for link-structure analysis (SALSA) and the TKC effect. In: 9th Intl. World Wide Web Conf. Amsterdam, Netherlands, May 2000
8 Hawking D. Overview of the TREC-9 Web Track. In: Proc. of the 9th Text Retrieval Conf. (TREC-9), National Institute for Standards and Technology, 2001
9 Craswell N, Hawking D. DRAFT Overview of TREC-2002 Web Track. http://trec.nist.gov/, 2002
10 Wu Lide, Huang Xuanjing, Niu Junyu, Xia Yingju, Feng Zhe, Zhou Yaqian. FDU at TREC2002: Filtering, Q&A, Web and Video Tasks. In: Proc. of the 11th Text Retrieval Conf. (TREC-2002), 2002
11 Voorhees E M. DRAFT Overview of TREC 2002. http://trec.nist.gov/, 2002

(上接第167页)

Springer-Verlar, 1999. 197～214
32 Singhai A, Sane A, Campbell H. Quarterware for Middleware. In: Proc. of the 18th Intl. Conf. on Distributed Computing Systems (ICDCS'98), Amsterdam, May 1998. 192～201
33 Cazzola W, Ancona M. mChaRM: A Reflective Middleware for Communication-based Reflection: [Technical Report DISI-TR-00-09]. DISI, Universita degli Studi di Milano, Milan, May 2000
34 Schmidt D C, Cleeland C. Applying Patterns to Develop Extensible ORB Middleware. IEEE Communications Magazine Special Issue on Design Patterns, 1999, 37(4): 54～63
35 Nahrstedt K, Chu H H, Narayan S. QoS-aware Resource Management for Distributed Multimedia Applications. Journal of High-Speed Networking, Special Issue on Multimedia Networking, 1998, 7: 227～255
36 Kiczales G, Lamping J, Mendhekar A, Maeda C, Lopes C V, Loingtier J M, Irwin J. Aspect-oriented programming. In: Proc. of 11th European Conf. on Object-Oriented Programming, Finland, LNCS 1241, Springer-Verlag, 1997. 220～242
37 Barga R, Pu C. A Practical and Modular Method to Implement Extended Transaction Models. In: Proc. Intl. Conf. on Very Large Data Bases, Zurich, Switzerland, 1995. 206～217
38 Barga R, Pu C. Reflection on a Legacy Transaction Processing Monitor. In: proc. of the Reflection'96 Conference, San Francisco, 1996
39 Wang Y M, Lee W-J. COMERA: COM Extensible Remoting Architecture. In: Proc. of COOTS, April 1998