支持集成与扩展的构件接口模型研究

王 琼 杜承烈
(西北工业大学计算机学院 西安 710072)

摘 要 接口是构件与外界交互的唯一场所,接口的设计直接涉及到构件集成与扩展的难易程度。提出了一个面向软件体系结构的具有性能约束的构件接口模型,该接口模型遵守构件的封装性,允许客户了解构件的拓扑结构等信息,支持构件在接口处的集成与扩展;同时还增加了性能描述和性能保障机制,以实现对构件的性能控制。该模型在某国防基础科研项目的研究中得到了应用,达到了比较好的效果。

关键词 软件体系结构,构件,接口,集成,扩展,性能约束
中图法分类号 TP303 **文献标识码** A

Study of Component Interface Model Supported Integration and Expansion

WANG Qiong DU Cheng-lie
(College of Computer Science, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract Component interface is the only interaction between component and environment, and the design of interface affects the complexity of integration and expansion of component directly. An architecture-oriented and with performance-constraint component interface model (AOPCCIM) was proposed. This interface model complies with encapsulation characteristic of component and allows customer to understand the information of component, such as the topology of component, for integration and expansion of component in component interface. It also adds performance description and performance guarantee mechanism to control the component performance. The application of this model in some foundational projection has proved its effection.

Keywords Software architecture, Component, Interface, Integration, Extension, Performance constraint

接口是构件与外界交互的唯一场所,一方面,构件接口把客户请求与构件的实现完全隔离开来;另一方面,客户对构件的了解和使用只能通过接口来实现。接口理论支持组装设计,它允许将设计任务劈成一定数量的子任务,每个子任务由独立的人员来设计;因此可以将接口 F 精化成形如 $(F_1||F_2||\cdots||F_k)\theta$ 的接口[1]。接口的设计直接涉及到构件集成与扩展的难易程度。在软件体系结构设计层次上,构件通过接口定义了与外界的信息传递以及所承担的系统责任,构件接口包括了构件与周围环境的全部交互内容。除此之外,环境不应对构件作任何其它与接口无关的假设,例如实现细节等。因此,构件之间的集成可以在接口处定义,从而支持理想的构件集成方式[2]。但是,传统的软件构件接口部分通常只定义了其对外提供的功能,构件对外要求的功能隐藏在构件实现的细节中,因此难以根据接口处的信息定义构件的集成,阻碍了 CBSE 的发展和构件市场的形成。

以往对复用的研究侧重于功能复用,模型侧重于构件功能以及构件功能接口相关的内部和外部接口交互的描述,但是要使软件开发真正地走上工程化和产业化道路,那么各个领域都应该可以实现复用;不同的领域有各自的特点,必然导致对复用的要求不仅仅是功能,针对领域特点还会有相应的性能要求,例如军工领域、航空航天领域等的应用对时间的要求很高。因此要将软件复用真正广泛深入到各个领域中去,具有性能约束的软件复用是必然的也是必需的。

因此,定义新的构件接口模型、扩充和增强构件接口规约成为新的重要研究课题[2]。针对构件的集成与扩展,本文对文献[3]中的接口模型进行了扩展,引入了性能约束(关于性能约束方面,本文以实时性为例,引入 timer[9] 来实现),提出了一个面向软件体系结构的具有性能约束的构件接口模型 AOPCCIM(Architecture Oriented and with Performance-Constraint Component Interface Model),该模型贯通体系结构设计和 CBSE 构件组装实现,运用层次结构控制构件行为协议规约的复杂度。与文献[3]相比,AOPCCIM 增加了扩展端口和性能控制端口,支持构件的集成与扩展,同时还增加了性能约束和性能保障机制,实现了对构件的性能控制。

1 相关研究工作

关于构件接口描述理论,有不少学者提出了自己的思路,对传统接口模型作了扩展和描述。文献[4]提出了一种改进

到稿日期:2009-08-10 返修日期:2009-10-11 本文受航空科学基金(2007ZD53043),"十一五"武器装备预先研究项目(102010101),某国防基础科研项目资助。

王 琼(1973-),女,博士生,主要研究方向为分布测控与仿真,E-mail:wq78026@gmail.com;杜承烈(1970-),男,教授,博士生导师,主要研究方向为分布测控与仿真。

的软构件接口模型，该模型是基于自动计数机——一种传统有限状态机的扩展，描述了构件提供接口（提供哪些服务及什么时候提供）和使用接口（使用哪些服务和什么时候使用）；其独特之处在于接口的连接允许自适应地选择一个构件所提供的功能，以防所需求的外部构件没有全部准备好；该方法为构件的自动组装生成系统提供了可能，其实现依赖于一个负责整个提供服务接口的协议，但是该模型没有提供相应的协议负责需求服务接口，否则这种构件连接的自适应选择会更加高效。文献[3]提出了一种面向体系结构的构件接口模型ACIM，它既能表达体系结构设计的高层抽象构件，又能表达底层代码级别的实现构件，是 SA 和 CBSE 技术融合的一种探索和尝试；同时，基于 CSP，提出了 ACIM 的两级构件接口行为协议及其形式规约方法，能够简化复杂构件的行为协议规约和规约各种抽象级别构件的行为协议；但是没有提及对于性能方面的约束。文献[5]提出了一个具有性能约束的构件模型——PCCM，在该模型的接口描述中除了传统的功能接口描述，为了达到性能约束的要求，还增加了性能接口维，形成了{功能接口，性能接口}二维构件模型；该模型为领域构件性能约束的实现提供了一个思路，但是没有考虑构件的进一步集成与扩展。文献[6]开发了一种叫做 CIDER（Component Interface Descriptor）的构件接口描述语言，该语言是一个面向对象的语言，它使得软件工程师可以捕捉、集成和重用基于可重用软构件模型的构件接口。文献[7] 介绍了一种基于时序逻辑的构件交互协议规约方法；该协议规约采取互动的方法限制构件的签名要素（即属性，操作和事件），这种方法的特别之处是允许构件交互协议或构件交互约束的逐步规约，它与文献[3]提出的协议规约思想类似，但是没有明确的分层概念。

还有一些其它的研究思路，这里不一一列举。

2 面向软件体系结构的具有性能约束的构件接口模型

2.1 AOPCCIM 的概念模型

AOPCCIM 的基本概念模型如下：

（1）每个构件具有一个接口，多个构件能够实现同一接口。通过接口理解和运用构件，接口是构件提供自身信息并与外界交互的地点。

（2）构件接口包含一个或者多个端口，端口分为 4 类：定制端口、交互端口、性能控制端口和扩展端口。定制端口用于对构件自身功能进行定制，实现构件自身功能的完善，局部于定制的单个构件，不影响软件系统整体。交互端口包含输入输出，分别对应构件需要外界提供的服务和构件向外界提供的服务，用于构件与其它构件交互、通信，从而构件之间彼此协作，完成软件系统的整体功能；为了支持构件的扩展集成，交互端口还包含几个预留端口作为备用。性能控制端口利用构件的性能描述初始化构件的运行环境，调用构件等功能端口使之达到要求的性能，例如为了实现实时，利用实时控制端口规定构件运行的开始和结束时间以及设置构件运行时的线程数、线程的优先级等触发系统来达到要求的性能指标；不同的性能需求对应于不同的性能控制端口。已有的成品构件可能无法满足所有客户的需求，因此 AOPCCIM 还设计了一类扩展端口，以供客户对构件进行扩展集成；对扩展端口来说，具体的软构件仍然是黑盒的，但是构件的内部结构信息可见。

（3）定制端口包括数据参数、类型参数、函数参数或简单对象，通常由构件组装人员提供。定制端口不能包含向外提供服务元素。

（4）交互端口包括共享的数据、服务提供操作、服务请求操作、向外发出消息、能够响应的外部消息、写入数据的管道、读出数据的管道，同时能够包括其它构件接口。

（5）性能控制端口根据控制的性能不同，包含的性能信息也不同，比如实时性能控制端口包含的性能信息有截止期、任务的同步、共享资源、线程优先级等。

（6）扩展端口包含有构件内部的子构件、函数、拓扑结构以及交互端口中的预留端口等相关信息。

（7）构件接口具有两级行为协议：端口级行为协议和接口级行为协议。

AOPCCIM 各元素之间的关系如图 1 所示。

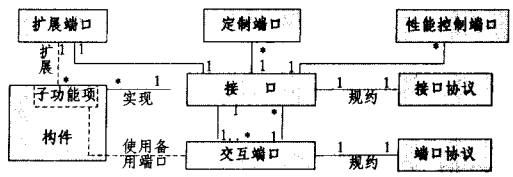

图 1 AOPCCIM 各元素之间的关系

2.2 AOPCCIM 分析

AOPCCIM 构件接口模型具有两个特点：一是不同端口的分类，二是交互端口和接口的嵌套定义。

AOPCCIM 进行端口分类，即定制端口、交互端口、性能控制端口和扩展端口。虽然它们的型构基本相同，但是逻辑功能却不相同。定制端口用于构件功能的定制和完善；交互端口用于构件与外界的交互和通信；性能控制端口类似于定制端口，但它是对性能而非功能的定制；扩展端口用于满足用户的特殊需求。定制端口和扩展端口经常在构件部署的时候设定，而交互端口能够与其它构件的端口进行动态交互，性能控制端口则是在构件与其他构件动态交互时实现。几类端口的分离符合多维关注分离的思想，易于构件理解、定制和复用，同时体现了 SA 的思想，即注重构件之间的逻辑交互关系。

SA 的研究把构件接口依据其逻辑交互功能划分为端口，形成二级结构，而 AOPCCIM 构件接口模型提出端口和接口相互嵌套定义，形成一个树型层次结构，是 SA 二级接口结构思想的深化。其遵循的思想是：复杂软件系统层次合成已经是普遍接受的原则，复杂的构件接口和交互协议同样应该能够进行层次合成和定义。由此带来的好处有：(1)控制接口规约复杂度；(2)支持 SA 的求精与实现；(3)提升构件组装的抽象级别等。

2.3 AOPCCIM 的型构规约

下面用 BNF 范式给出 AOPCCIM 的型构规约，接口协议规约和端口协议规约见 2.4 节。

```
Component Interface Specification::=
  INTERFACE interface_name
  {
    [0+ Custom_port_section]    //0 个或多个定制端口定义
    [1+ Interaction_port_section] //1 个或多个交互端口定义
    [0+ Performance_control_port_section] //0 个或多个性能
```

控制端口定义

[0+ Extend_port_section] //0 个或多个扩展端口定义

[1 Interface_level_protocol] //1 个构件接口级交互协议定义

}

Custom_port_section::=

CUSTOM_PORT portname

{

[0+ Data_Parameter] //0 个或多个数据参数定义

[0+ Type_Parameter] //0 个或多个类型参数定义

[0+ Function_Signature_Declaration] //0 个或多个函数参数

[0+ Object_Signature_Declaration] //0 个或多个对象定义

//上述诸项不能全部为空

}

Interaction_port_section::=

INTERACTION_PORT portname

{

[0+ OWN Data_Definition]

//0 个或多个交互端口提供的、供交互对方共享的数据

[0+ SHARE Data_Definition]

//0 个或多个交互端口共享的、属于交互对方的数据

[0+ PROVIDE Operation_Signature_Declaration]

//0 个或多个提供服务操作定义

[0+ REQUEST Operation_signature_Declaration]

//0 个或多个请求服务操作定义

[0+ NOTIFY Message_Signature_Declaration]

//0 个或多个消息通知的定义

[0+ RESPONSE Message_Signature_Declaration]

//0 个或多个能够响应的消息定义

[0+ WRITE Pipe_Signature_Declaration]

//0 个或多个写入数据管道定义

[0+ READ Pipe_Signature_Declaration]

//0 个或多个读入数据管道定义

[0+ INTERFACE Interface_name Instance_name]

//0 个或多个接口实例定义

[0+ DUAL INTERFACE Interface_name Instance_name]

//0 个或多个对偶接口实例定义

//上述诸项不能全部为空

[1 Port_level_protocol] //1 个端口协议定义

}

Performance_control_port_section::=

PERFORMANCE_CONTROL_PORT portname //以时间性能控制为例

{

[Deadline_Parameter]* //0 个或多个截止期参数

[Tasksynchronize_Parameter]* //0 个或多个任务同步参数

[Shareresource_Parameter]* //0 个或多个共享资源参数

[Thread_prior_Parameter]* //0 个或多个线程优先级参数

//上述诸项不能全部为空

}

Extend_port_section::=

EXTEND_PORT portname

{

[0+ Data_Parameter] //0 个或多个数据参数定义

[0+ Type_Parameter] //0 个或多个类型参数定义

[0+ Function_Signature_Declaration] //0 个或多个函数参数

[0+ Object_Signature_Declaration] //0 个或多个对象定义

[1+ Preport_Signature_Declaration] //1 个或多个预留端口定义

[0+ Subcomponent_Interface_Signature_Declaration] //0 个或多个子构件接口定义

[1 Component_level_topology_information] //1 个构件拓扑结构信息

//上述诸项不能全部为空

}

2.4 AOPCCIM 的行为协议规约

要想有效地开发、管理和使用构件，必须要有一个综合的构件接口表征。AOPCCIM 采用如图 2[8] 所示的综合框架来描述构件接口表征。其中，底层是构件的 signature，是构件与外界进行交互的最基本元素的描述，包含交互等所有必须的机制，例如属性、操作和事件等；构件接口的 signature 是构件的功能表征。第二层是对适当使用构件 signature 的约束，也就是说，构件的使用应当遵守这些约束以防止发生例外、错误、无法预知的行为。构件 signature 和它的约束一起定义了构件的全部功能。第三层是关于构件接口 signature 根据在给定使用场景中的角色进行的封装，目的是使得构件接口根据使用环境可以有不同的配置。第四层是关于构件的非功能性属性表征，比如构件质量方面有性能、可靠性、安全性等。在该构件接口框架中非功能属性有一个特殊的位置，很多需要与接口 signature 和配置交错。

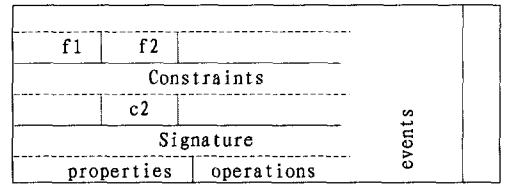

图 2 构件接口表征的综合描述框架

AOPCCIM 的行为协议涉及到描述框架的第一、二、三层，不同的操作或者行为需要不同的约束和配置。它从两个级别描述构件的行为协议：端口级行为协议和接口级行为协议。端口级行为协议描述端口中元素的正确使用顺序和其间存在的依赖关系，反映端口的交互能力和功能行为；接口级行为协议描述接口的多个端口正确使用顺序和其间存在的依赖关系，反映构件整体的交互能力和功能行为。在规约构件接口行为协议的时候，将其包含的端口作为整体对待。在规约端口级行为协议的时候，将其包含的接口元素作为整体对待。两级行为协议规约结合接口端口嵌套定义，从而实现构件行为协议的层次化规约。

在 AOPCCIM 的行为协议描述中，存在两种接口元素依赖关系，即服务依赖和顺序依赖[10]。

定义 1（服务依赖） 某一提供服务元素在执行过程中需要执行另一服务请求元素以帮助完成该服务提供的操作，则称前者依赖于后者的服务。

定义 2（顺序依赖） 若某一元素在执行之前或之后必须执行另一元素，否则导致出错或执行过程无法进行下去，那么称前者对后者有顺序依赖。

服务依赖和顺序依赖不一定是双向的，即定义 1、定义 2 中的前者对后者有服务依赖（顺序依赖），但是后者对前者不一定存在相应的依赖。只有当二者均属于上述定义中的情形时，才能说二者之间存在互相依赖关系（关于这一点，本文认

为文献[10]中的定义 8-1、8-2 不够准确)。

CSP 提供丰富的行为描述机制,有完善的理论体系和成熟的支持工具,因此,AOPCCIM 基于 CSP 描述构件接口行为协议。AOPCCIM 的特点在于基于构件接口型构的描述,确定构件的事件集合和描述构件的行为协议,从而协议规约变得简单和易于理解。数据共享、操作管道、消息是构件交互的基本机制,对于具有时间特性的构件,timer 也是交互过程不可缺少的,因此它们一起组成了构件交互端口的元素。下面给出它们的事件及事件定义,具体描述和运用方法见文献[3]。

定义 3(共享数据的事件) 设共享数据名字为 Sx,则其事件、事件含义如表 1 所列。

表 1 共享数字包字为 Sx 的事件及事件含义

属性 \ 事件	OWN Sx	SHARE Sx
Read. Sx	OE. 交互对方读数据 Sx	IE. 读交互对方的数据 Sx
Write. Sx	OE. 交互对方写数据 Sx	IE. 写交互对方的数据 Sx
Visite. Sx	简写,等价	简写等价 Write. Sx∥Read. Px

定义 4(timer 的事件) 设 timer 的超时时限为 t,则其事件、事件含义如表 2 所列(关于 timer 的详细结构及含义参考文献[9])。

表 2 timer 超时时限为 t 的事件及事件含义

属性 \ 事件	timer	timer$^{\Delta t}$(vx, P, Q). E
timer. create	信道创建时相应的 timer 也随着创建	无意义
timer. begin	信道被激活时相应的 timer 开始计时	接收到 x 后执行进程 P 然后执行进程 E
timer. end	当 timer 的 t 减到 1 时超时,timer 被销毁	仍未接收到 x 则执行进程 Q

定义 5(操作调用的事件) 设操作名字为 Ox,则其事件、事件含义如表 3 所列。

表 3 操作名字为 Ox 的事件及事件含义

属性 \ 事件	Provide Ox	REQUIRE Ox
Call. Ox	OE. 交互对方调用操作 Ox	IE. 调用交互对方的操作 Ox
Return. Ox	IE. Ox 执行结束并返回	OE. Ox 执行结束并返回
Ox	简写,等价 Call. Ox→Return. Ox	简写,等价 Call. Ox→Return. Ox

定义 6(消息机制的事件) 设消息名字为 Mx,则其事件、事件含义如表 4 所列。

表 4 消息名字为 Mx 的事件及事件含义

属性 \ 事件	SEND Mx	RESPONSE Mx
Send. Mx	IE. 发送消息 Mx	无意义
Response. Mx	无意义	IE. 响应消息 Mx

定义 7(管道机制的事件) 设管道名字为 Px,则其事件、事件含义如表 5 所列。

表 5 管道名字为 Px 的事件及事件含义

属性 \ 事件	WRITE Px	READ Px
Begin. Px	IE. 开始管道传输数据	IE. 开始接收管道数据
End. Px	IE. 管道传输数据结束	IE. 管道读取数据结束
Px	简写,等价 Begin. Px→End. Px	简写,等价 Begin. Px→End. Px

定义 8(性能控制机制的事件) 设控制的性能为 pro,pro 相应的性能表征有 x,y 和 z,则其事件、事件含义如表 6 所列。

表 6 控制性能 pro 的事件及事件含义

属性 \ 事件	ASSIGN rpo	SET enviroment
pro. x	给 pro. x 赋值	设置满足 pro. x 的执行环境
pro. y	给 pro. y 赋值	设置满足 pro. y 的执行环境
pro. z	给 pro. z 赋值	设置满足 pro. z 的执行环境

定义 9(端口执行和事件) 端口执行是端口的一次逻辑交互,从而完成其定义的功能或子功能。设端口名字为 Px,端口执行的开始是为完成该功能其包含的元素开始执行,用事件 Run. Px 表示。端口执行结束是指该相关元素执行结束,用事件 End. Px 表示。它的具体执行由该端口的端口协议决定,接口协议规约对其进行抽象。事件 Px=Run. Px→End. Px。

定义 10(接口执行和事件) 接口执行是接口的一次逻辑交互,从而完成其定义的功能或子功能。设接口名字为 Ix,接口执行的开始是为完成该功能其包含的端口开始执行,用事件 Run. Ix 表示。接口执行结束是相关端口执行结束,用 End. Ix 表示。接口中相关端口的执行具体由接口的行为协议规约,端口协议规约对其进行抽象。事件 Ix=Run. Ix→End. Ix。

结束语 基于构件的计算模型可以归纳为两个[1]:构件模型和接口模型。构件模型描述每个构件在一个任意环境中如何行为;接口模型描述构件期望从环境中得到什么。构件模型支持组装抽象以及基于构件的验证;接口模型支持组装精化以及基于构件的设计。它们都是为构件的重用服务。接口是构件与外界交互的唯一场所,接口的设计直接影响到构件集成与扩展的难易程度;因此,构件接口模型及协议规约的研究是现代软件工程领域的一个重要内容,目前已经有了一些研究成果,但是对于支持构件集成与扩展的机制的研究并不多见。而形式化方法是提高软件系统,特别是 safety-critical 系统的安全性与可靠性的重要手段,用形式化方法来研究构件的集成与扩展具有很好的前景。

针对构件的集成与扩展,本文对文献[3]中的接口模型进行扩展,引入性能约束和 timer,提出了一个面向软件体系结构的具有性能约束的构件接口模型 AOPCCIM,并研究了基于该接口模型的集成方法。AOPCCIM 模型遵守构件的封装性,通过增设扩展端口允许客户了解构件的拓扑结构等信息,支持客户根据需求对构件功能的裁剪和扩展,真正支持理想的构件集成方式;同时还增加了性能描述和性能保障机制,通过对构件运行环境的初始化来实现对构件的性能控制。该模型在某国防基础科研项目子项目——虚拟试验实时软总线的研究中得到了应用,达到了比较好的效果。

进一步的工作是继续深入研究接口模型及其协议规约,实现用形式化方法分析构件的集成与扩展及其兼容性。

参 考 文 献

[1] deAlfaroL. InterfaceTheoriesforComponent-BasedDesign[C]// Lecture Notes in Computer Science, London: Springer-Verlag, 2001:148-165

(下转第 182 页)

```
}
dos. writeUTF("0000"); //发送数据传递结束标识
bs. close();
InputStream in =cs. getInputStream();
//接收节点服务器传递的结束标识
DataInputStream din=new DataInputStream(in);
//如果节点服务器传来结束标识"stop",则计算出程序的执行时
间
if((str=(din. readUTF())). equals("stop"))
{
t=System. currentTimeMillis()-t;
System. out. println("程序所用时间为:"+t);
}
```

节点服务器中的核心代码:

```
//clientword. txt:节点服务器用来存储调度系统传递来的要处
理的数据
BufferedWriter bs=new BufferedWriter(new FileWriter("client-
word. txt"));
InputStream is=s. getInputStream();
//接收调度系统传来的待处理数据
DataInputStream ds=new DataInputStream(is);
OutputStream os=s. getOutputStream();
DataOutputStream din=new DataOutputStream(os);
//把调度系统传递来的数据保存在节点服务器中
while(true)
{
str=ds. readUTF();
if(str. equals("0000"))
//接收到数据传递结束标识就退出读取数据操作
break;
System. out. println(str);
bs. write(str);
}
```

测试结果如图 5 和图 6 所示。

图 5 数据流测试结果

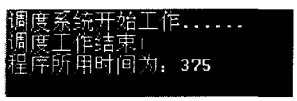

图 6 程序流测试结果

测试结果表明,系统可以正确实现,性能表现较好且稳定,并且程序流执行效率高于数据流程序的执行效率。系统运行一段时间,通过日志监测表明,系统没有发现异常错误,也没有收到任何错误报告。

结束语 本文提出了云检索在程序并行化执行模式上的改进方案——采用程序流并行执行模式,并给出云检索核心技术分布式文件系统及分布式数据库系统的架构模式。建立云检索系统理论基础及架构方案,为实现集群系统分布式数据及海量数据的检索处理提供了新的思路。

参 考 文 献

[1] 初晓博,秦宇. 一种基于可信计算的分布式使用控制系统[J]. 计算机学报,2010(1):93-102

[2] 李卫疆,赵铁军,王宪刚. 基于上下文的查询扩展[J]. 计算机研究与发展,2010,47(2):300-304

[3] 付雄,王汝传,邓松. 无线传感器网络中一种能量有效的数据存储方法[J]. 计算机研究与发展,2009,46(12):2111-2116

[4] Hsieh J W,Kuo T W,Chang L P. Efficient Identification of Hot Data for Flash Memory Storage Systems[J]. ACM Transactions on Storage,2006,2(1):22-40

[5] 杨代庆,张智雄. 基于 Hadoop 的海量共现矩阵生成方法[J]. 现代图书情报技术,2009(4):23-26

[6] 刘立坤,武永卫,徐鹏志,等. CorsairFS:一种面向校园网的分布式文件系统[J]. 西安交通大学学报,2009,43(8):43-47

[7] 张刚,谭建龙. 分布式信息检索中文档集合划分问题的评价[J]. 软件学报,2008,19(1):136-143

[8] 张海军,史树敏,朱朝勇,等. 中文新词识别技术综述[J]. 计算机科学,2010,37(3):6-10

[9] 李小龙,林亚平,胡玉鹏,等. 基于分组的分布式节点调度覆盖算法[J]. 计算机研究与发展,2008,45(1):180-188

[10] 张刚,刘悦,郭嘉丰,等. 一种层次化的检索结果聚类方法[J]. 计算机研究与发展,2008,45(3):542-547

[11] 侯东风,刘青宝,张维明,等. 一种适应性的流式数据聚集计算方法[J]. 计算机科学,2010,37(3):152-155

[12] 马亮,陈群秀,蔡莲红. 一种改进的自适应文本信息过滤模型[J]. 计算机研究与发展,2005,42(1):79-84

[13] 王鹏,陈高云,安俊秀,等. 移动搜索引擎原理与实践[M]. 北京:机械工业出版社,2009

[14] 郎皓,王斌,李锦涛,等. 文本检索的查询性能预测[J]. 软件学报,2008,19(2):291-300

[15] 黄瑞,史忠植. 一种新的 Web 异构语义信息搜索方法[J]. 计算机研究与发展,2008,45(8):1338-1345

[16] 陈全,邓倩妮. 云计算及其关键技术[J]. 计算机应用,2009,29(9):2562-2567

[17] 叶育鑫,欧阳丹彤. 语义 Web 搜索技术研究进展[J]. 计算机科学,2010,37(1):1-5

[18] 洪亮,卢炎生,陈锦富,等. 一种基于位置数据库聚类的动态适应·缓存位置信息策略[J]. 计算机研究与发展,2008,45(7):1203-1210

[19] 田晓珍,尚冬娟. Web 个性化服务[J]. 重庆工学院学报:自然科学版,2008,22(7):76-80

(上接第 140 页)

[2] 张世琨,张文娟,常欣. 基于软件体系结构的可复用构件制作和组装[J]. 软件学报,2001,12(9):1351-1359

[3] 任洪敏,张敬周,钱乐秋. 面向体系结构的构件接口模型及其形式化规约[J]. 计算机工程,2005,12:67-69

[4] Reussner R H. Enhanced Component Interfaces to Support Dynamic Adaption and Extension[C]// Proceedings of the 34th Hawaii International Conference on System Sciences. 2001

[5] 卢炎生,查虎平,徐丽萍. PCCM:具有性能约束的构件模型[J]. 计算机科学,2004,3:89-92

[6] Whittle B,Ratcliffe M. Software component interface description for reuse[J]. Software Engineering Journal,1993,1(8):307-318

[7] Han Jun. Temporal logic based specification of component interaction protocols[C]// Proceedings of the 2nd Workshop of Object Interoperability at ECOOP 2000. Cannes,France,June 2000

[8] Han Jun. A Comprehensive Interface Definition Framework for Software Components[C]// Proceedings of the 1998 Asia-Pacific Software Engineering Conference. IEEE Computer Societ,Taipei,Taiwan,December 1998:110-117

[9] Berger M. Towards Abstractions for Distributed Systems[D]. Imperial College,Department of computing,2002