

基于运行时模型的混合云管理方法*

陈星^{1,2}, 兰兴土^{1,2}, 李隘鹏^{1,2}, 郭文忠^{1,2}, 黄罡^{3,4}



¹(福州大学 数学与计算机科学学院, 福建 福州 350108)

²(福建省网络计算与智能信息处理重点实验室(福州大学), 福建 福州 350108)

³(北京大学(天津滨海)新一代信息技术研究院, 天津 300452)

⁴(高可信软件技术教育部重点实验室(北京大学), 北京 100871)

通信作者: 黄罡, E-mail: hg@pku.edu.cn

摘要: 随着云计算技术的普及,涌现出众多不同用途、不同类型的云计算平台.为了满足遗产系统整合和动态资源扩展等需求,常常需要构造混合云来统一管理不同云平台中的计算和存储资源.然而,不同云平台的管理接口和管理机制存在差异,使得开发混合云管理系统难度大、复杂度高.提出一种基于运行时模型的混合云管理方法:首先,在云平台管理接口的基础上,构造单一云平台的运行时模型;其次,根据云平台领域知识,提出一种云平台软件体系结构的统一模型;最后,通过模型转换,实现云平台统一模型到运行时模型的映射.于是,管理程序可以建立在云平台统一模型的基础上,降低了混合云管理系统开发的难度和复杂度.另外,还实现了基于运行时模型的 CloudStack 和亚马逊 EC2 混合云管理系统,并对方法的可行性和有效性进行了验证.

关键词: 混合云管理;软件体系结构;运行时模型

中图法分类号: TP316

中文引用格式: 陈星, 兰兴土, 李隘鹏, 郭文忠, 黄罡. 基于运行时模型的混合云管理方法. 软件学报, 2017, 28(7): 1881-1897. <http://www.jos.org.cn/1000-9825/5113.htm>

英文引用格式: Chen X, Lan XT, Li AP, Guo WZ, Huang G. Runtime model based approach to managing hybrid clouds. Ruan Jian Xue Bao/Journal of Software, 2017, 28(7): 1881-1897 (in Chinese). <http://www.jos.org.cn/1000-9825/5113.htm>

Runtime Model Based Approach to Managing Hybrid Clouds

CHEN Xing^{1,2}, LAN Xing-Tu^{1,2}, LI Ai-Peng^{1,2}, GUO Wen-Zhong^{1,2}, HUANG Gang^{3,4}

¹(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350108, China)

²(Fujian Provincial Key Laboratory of Networking Computing and Intelligent Information Processing (Fuzhou University), Fuzhou 350108, China)

³(Beida (Binhai) Information Research (Peking University), Tianjin 300452, China)

⁴(Key Laboratory of High Confidence Software Technologies of Ministry of Education (Peking University), Beijing 100871, China)

Abstract: With the development of cloud computing, cloud platforms of different types and purposes are emerging. Hybrid clouds are needed to manage cross-domain computing and storage resources in a unified manner in order to satisfy management requirements such as legacy system integration and dynamic resource scaling. However, there are differences in management interfaces and mechanism between different cloud platforms, which cause great difficulties and high complexity to construction of hybrid clouds. In this paper, a

* 基金项目: 国家自然科学基金(61402111, 61222203); 国家高技术研究发展计划(863)(2015AA01A202); 福建科技重大项目(2015H6013); 福建省科技平台建设项目(2014H2005)

Foundation item: National Natural Science Foundation of China (61402111, 61222203); National High-Tech R&D Program of China (863) (2015AA01A202); Major Science and Technology Project of Fujian Province, China (2015H6013); Science and Technology Platform Development Program of Fujian Province, China (2014H2005)

收稿时间: 2015-09-07; 修改时间: 2015-11-18; 采用时间: 2016-06-14; jos 在线出版时间: 2016-10-11

CNKI 网络优先出版: 2016-10-12 16:26:45, <http://www.cnki.net/kcms/detail/11.2560.TP.20161012.1626.011.html>

runtime model based approach to managing hybrid clouds is presented. First, the manageability of cloud platforms is abstracted as runtime models based on their management interfaces. Second, a unified model of cloud software architecture is provided according to the domain knowledge of current cloud platforms. Third, the synchronization between the unified model and cloud runtime models is ensured through model transformation. Thus, all the management logic can be carried out by executing programs on the unified model, which decreases the difficulty and complexity of hybrid cloud construction. The experiment on a real-world hybrid cloud, which consists of CloudStack and Amazon EC2, demonstrates the feasibility, effectiveness and benefits of the new approach in managing hybrid clouds.

Key words: hybrid cloud management; software architecture; models at runtime

云计算是一种能够通过网络以便利的、按需付费的方式获取计算资源的范型,这些资源来自一个共享的、可配置的资源池,并能够以省力和无人干预的方式获取和释放^[1].根据服务模型的维度,云计算分为基础设施即服务(infrastructure as a service)、平台即服务(platform as a service)和软件即服务(software as a service).然而,云计算平台的基础设施却通常类似,都是基于虚拟化技术将硬件资源进行分割或聚合,以实现按需缩减或扩展,并在虚拟化硬件之上辅以多种类型的基础软件,最终以服务的方式进行供给.

随着云计算技术的普及,涌现出众多不同用途、不同类型的云计算平台,包括以 AWS(Amazon Web service)^[2]为代表的各种公有云服务,以及基于 CloudStack^[3]、OpenStack^[4]、Eucalyptus^[5]等云管理系统的大量企业私有云.为了满足遗产系统整合和动态资源扩展等需求,一些大型企业常常需要构造混合云来统一管理不同云平台中的计算和存储资源.然而,不同云计算平台的管理系统存在差异,给混合云管理系统的开发带来了巨大挑战,主要来自以下两个方面.

一方面是云平台管理接口的异构性.不同云计算平台往往提供不同类型的管理接口,在功能名称、调用方法、输入参数、返回信息及管理效果等方面均存在差异,给管理系统的开发带来了极大的复杂度.

另一方面是云平台管理机制的异构性.不同云计算平台往往提供不同类型的管理机制,在资源分类、组织方式及系统特性等方面均存在差异,给管理系统的开发带来了极大的难度.

从系统实现的角度来看,混合云管理是一组管理任务的集合,每个管理任务由一组作用在一个或多个云计算平台上的管理操作构成,而每一个管理操作则是云平台自身提供的管理接口或是第三方提供的管理服务的调用.云计算平台的异构性导致不同云平台的管理操作存在较大的差异,混合云管理系统的开发需要熟悉不同云计算平台管理操作的详细功能、调用方法和数据格式,并在此基础上编写管理程序来实现多样化的管理任务.与分析 and 决策等混合云的管理逻辑相比,管理接口调用、底层数据处理等繁杂、琐碎的编程工作并不是混合云管理的核心,但它们却需要花费管理员大量的时间和精力.此外,由于管理程序建立在与特定云计算平台绑定的底层代码的基础上,其管理逻辑无法进行复用.即使管理机制类似,仍需要开发多个管理程序对不同的混合云进行管理.

混合云管理系统开发面临的主要问题是:其问题域与系统实现间存在着鸿沟,而通过硬编码实现问题域到系统实现的映射则会带来巨大的编程复杂性.软件体系结构用一组可管理的单元来表示系统的整体架构,能够扮演系统需求与系统实现之间的桥梁^[6],常用来解决需求到实现的映射过程中系统复杂性所带来的问题^[7].因此,系统的软件体系结构建模是理解系统问题域、简化目标系统开发的一种有效方法.而当前模型驱动工程的研究也支持问题域抽象到软件实现的系统级转换^[8,9].运行时软件体系结构模型用一组可管理的单元来表示系统的整体架构,通过将隐藏在系统内部的结构、状态、配置等运行时信息显示化地描述为标准的、面向管理者视角的结构化视图,能够有效地提高混合云管理系统开发的抽象层次和自动化程度^[10,11].运行时软件体系结构模型已经在学术界和工业界获得了广泛的关注.大量的研究工作证明了它在不同系统与管理方式下的重要作用^[12-14].

为了能够根据需求快速定制、开发混合云管理系统,本文将运行时模型引入到系统开发过程中,提出一种基于运行时模型的混合云管理方法,并在实际场景中验证方法的可行性和有效性.首先,为单一的云计算平台构造运行时模型,基于管理接口实现在模型层对单一云平台进行管理;其次,根据云计算平台领域知识,提出一种云平台软件体系结构的统一模型;最后,建立统一模型与单一云平台运行时模型之间的联系,通过模型转换实现

统一模型到云平台运行时模型的映射.于是,管理程序可以建立在云计算平台统一模型的基础上,极大地降低了开发混合云管理系统的难度和复杂度.

本文第 1 节概述方法的整体框架.第 2 节介绍云平台运行时模型的构造方法.第 3 节介绍云平台软件体系结构的统一模型.第 4 节介绍云平台统一模型到单一云平台运行时模型的映射方法.第 5 节介绍实例研究并对方法的可行性和有效性进行评估.第 6 节与相关工作进行比较.最后一节总结全文并展望未来的工作.

1 方法概览

图 1 是基于运行时模型的混合云管理方法的概览.方法将运行时软件体系结构模型引入到混合云管理过程中,通过模型转换实现云平台统一模型到单一云平台运行时模型的映射,使得能够面向云平台统一模型进行混合云管理程序开发.该方法主要包含 3 方面工作:

(1) 云计算平台运行时模型的构造方法;(2) 云平台软件体系结构的统一模型;(3) 统一模型到单一云平台运行时模型的映射方法.

首先,介绍一种云计算平台运行时模型的构造方法,以屏蔽云平台管理接口的异构性.云平台运行时模型是云计算平台软/硬件系统的抽象.管理员仅需要对云平台受管模块的功能信息和云平台管理接口的调用方法进行描述,构造方法就能够生成相应的云平台运行时模型,并支持运行时模型与系统状态的自动同步.于是,管理员可以在模型层对云计算平台进行管理.

其次,设计一种云平台软件体系结构的统一模型,以屏蔽云平台管理机制的异构性.混合云管理实际上是对不同类型的公有云服务或企业私有云进行协同管理,然而,不同云平台在资源分类、组织方式及系统特性等方面常常存在差异.统一模型在云计算平台领域知识的基础上,对常见云平台共有的资源类型及管理功能进行描述.于是,管理员可以通过统一视图对异构云平台进行管理.

最后,提出一种模型转换方法,以实现统一模型到单一云平台运行时模型的映射.不同类型云平台的运行时软件体系结构模型存在差异,管理员仅需对统一模型到单一云平台运行时模型的元素映射关系进行定义,转换方法就能够自动生成相应的模型转换程序,以保障模型间的同步关系.于是,管理员可以面向云平台统一模型进行混合云管理程序开发.

2 云计算平台运行时模型的构造方法

运行时模型对目标系统及其管理能力、体系结构内容与形式以及二者之间的关系分别进行建模,将隐藏在系统内部的结构、状态、配置等运行时信息显示化地描述为标准的结构化视图.为了使管理员能够通过模型读写实现系统监控,运行时模型与运行系统需要保持因果关联:运行系统的任何时间的信息均将反映到运行时模型上,而运行时模型的任何变化也会作用到运行系统上.

云平台管理接口的异构性给混合云管理系统的开发带来极大的复杂度,本文使用 SM@RT 工具^[15,16]进行云平台运行时模型的构造,并通过运行时模型实现在模型层对单一的云计算平台进行管理.SM@RT 包含一种领域特定的建模语言和一种支持运行时系统管理的代码生成器.开发人员使用 SM@RT 建模语言定义系统元模型和系统访问模型:(1) 系统元模型描述运行系统的体系结构;(2) 系统访问模型描述运行系统的管理能力.在以上两种模型的基础上,SM@RT 代码生成器能够自动生成同步引擎,支持运行时模型与运行系统的双向同

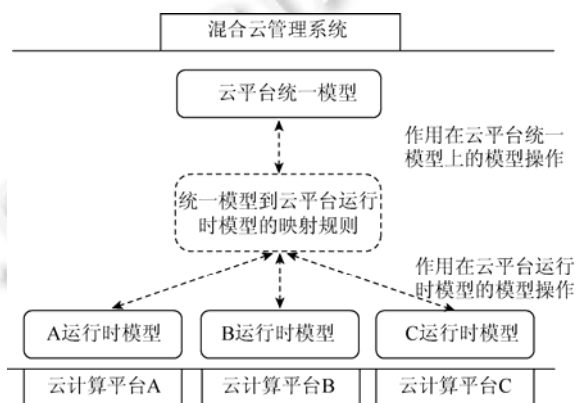


Fig.1 Overview of the runtime model-based approach to managing hybrid clouds

图 1 基于运行时模型的混合云管理方法的概览

步.于是,运行系统的任何运行时信息均将反映到运行时模型上,而运行时模型的任何变化也会作用到运行系统上.如图 2 所示,同步引擎为云平台中每一个虚拟机在运行时模型中创建一个可管理的“虚拟机”单元;而当某个“虚拟机”单元被删除时,同步引擎也能够发现运行时模型的变化,确定目标虚拟机并将其在云平台中关闭.在之前的工作中,我们对运行时模型的构造方法与支撑机制进行了深入的研究^[17,18],并对云平台运行时模型的构造方法进行了初步探讨^[19],本文不再赘述.

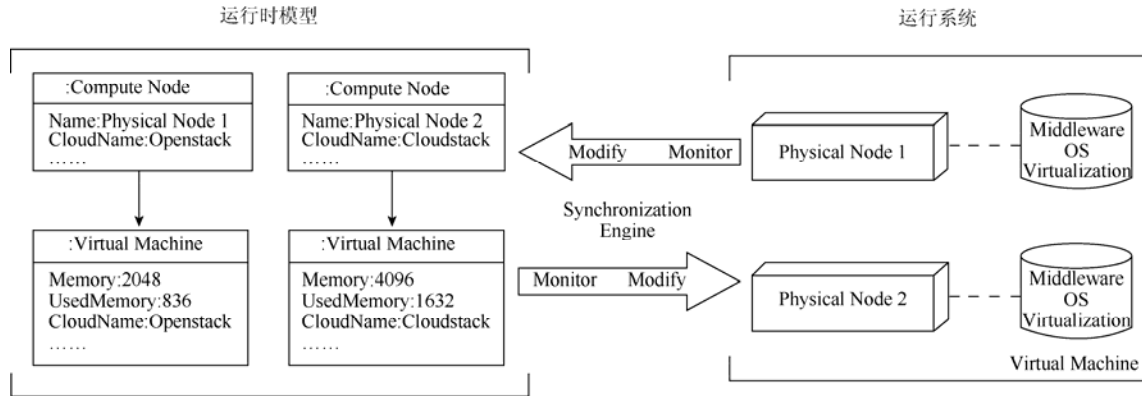


Fig.2 Synchronization between the runtime model and running system

图 2 运行时模型与运行系统的同步

3 云平台软件体系结构的统一模型

混合云管理实际上是对不同类型的公有云服务或企业私有云进行协同管理.在云计算平台领域知识的基础上,本文提出一种云平台软件体系结构的统一模型,对常见云平台共有的资源类型及管理操作进行描述;其中,云平台的资源类型及状态信息用模型元素及其属性进行表示,而云平台的管理操作则作用于对应模型元素的模型操作进行表示.统一模型包括使用者模型和管理员模型,分别表示面向使用者和管理员的受管资源及管理操作集合.

3.1 使用者模型

使用者模型能够对云平台使用者所拥有的计算、存储、网络等资源进行统一管理,其资源分配的最小单元是虚拟机(server),每个虚拟机都存在于一个项目(project)中.图 3 描述了使用者模型中的主要受管单元,包括使用者(Account)、项目(Project)、虚拟机映像(Image)、资源配置类型(ServerType)、网络子网(Network)、磁盘卷(Volume)及虚拟机(Server)等.其中,Account 表示使用者,描述使用者的基本信息,并包含该使用者所有关联项目的列表 Projects.Project 表示项目,描述项目的基本信息以及计算、存储、网络等资源的分配情况;项目允许多个使用者共享虚拟资源,因此,Project 包含该项目所有关联使用者的列表 ProjectAccounts.Image 表示一个虚拟机映像文件,是虚拟机软件系统的载体,Images 则表示项目可以使用的虚拟机映像文件的集合.ServerType 表示虚拟机资源配置类型,描述了 CPU、内存、存储等资源配置信息,ServerTypes 则表示项目可以使用的虚拟机资源配置类型的集合.Network 表示网络子网,描述该子网的基本信息,并包含该子网拥有的所有公共 IP 地址的列表 PublicIps;PublicIp 表示可供外部网络访问的 Ip 地址,为网络子网中拥有该 IP 地址的虚拟机提供外部网络的访问入口.Volume 表示磁盘卷,可为虚拟机增加额外的存储,Volumes 则表示项目中所有磁盘卷的集合.Server 表示虚拟机,描述虚拟机的映像文件(imageId)、资源配置等基本信息,其中,资源配置信息可以通过指定虚拟机配置类型(serverTypeId)或设置 CPU、内存、磁盘等属性值两种方式进行描述.同时,Server 还包含虚拟机所使用的虚拟网卡的列表 Nics;Nic 表示虚拟网卡,描述虚拟网卡的 ip 地址、mac 地址以及所在的网络子网等网络配置信息.Servers 则表示项目管理的所有虚拟机的集合.特别地,主要受管单元均包含 cloudName 和 dataCenterId 属

性,分别指出该受管单元所属的云平台及数据中心.

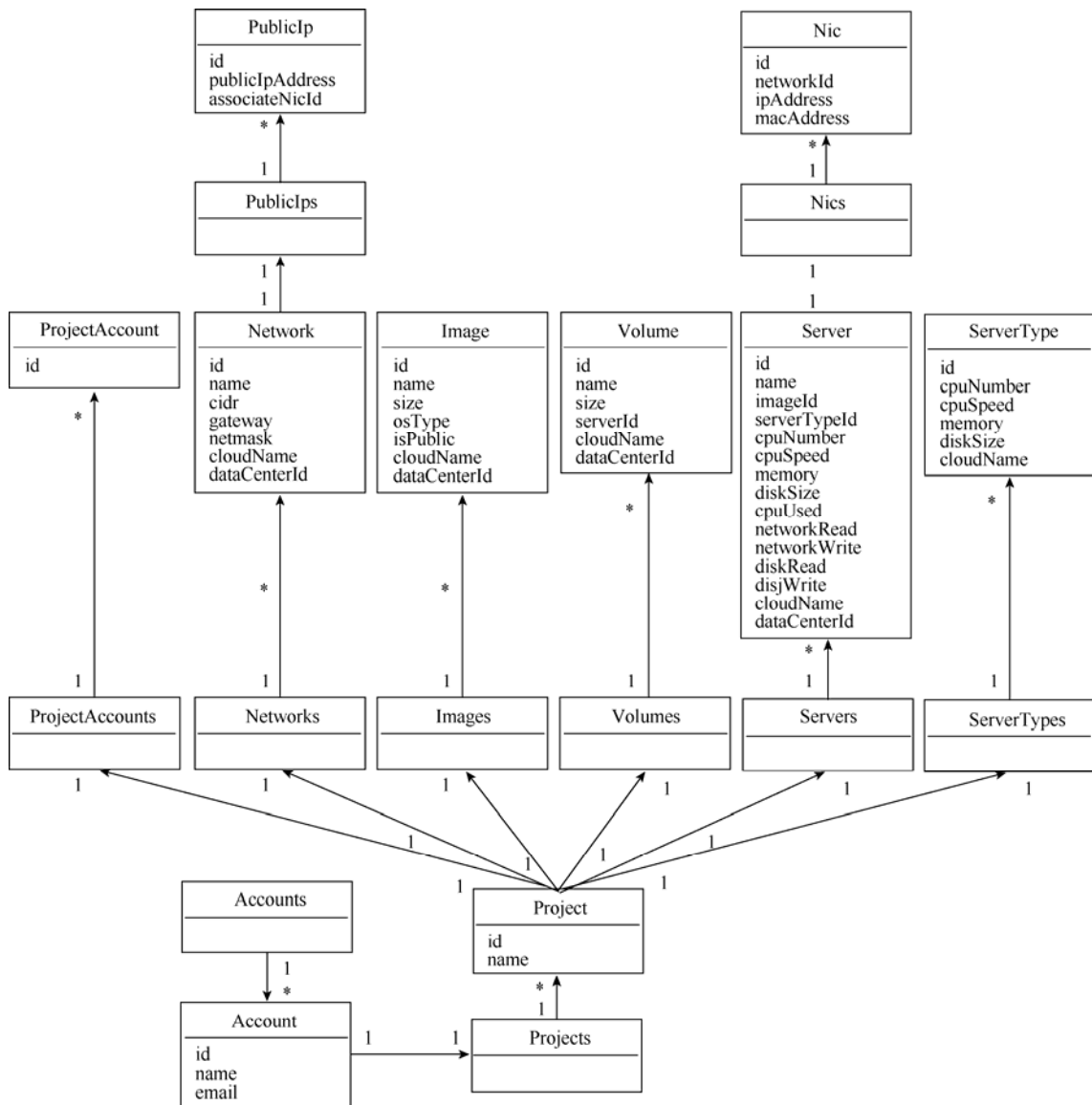


Fig.3 The user model of cloud software architecture

图 3 云平台软件体系结构的使用者模型

3.2 管理员模型

管理员模型能够为云平台管理提供一个全局视图,对计算、存储、网络等资源进行统一管理.图 4 描述了管理员模型中的主要受管单元,包括数据中心(DataCenter)、物理机(ComputeNode)、虚拟机映像文件(PublicImage)、资源配置类型(ServerType)、存储(ImageStore,VolumeStore)和网络(PublicNetwork)资源,以及项目和使用者等.CloudManagement 是管理员模型的根元素,包含数据中心列表 DataCenters、项目列表 Projects 和使用者列表 Accounts.其中,Projects 表示所有项目的集合,Accounts 表示所有使用者的集合,与项目、使用者相关的受管单元在上一节中已详细讨论,在此不再赘述.DataCenter 表示数据中心,包含物理机列表 ComputeNodes、公共虚拟机映像文件列表 PublicImages、公共虚拟机资源配置类型列表 ServerTypes、映像文

件存储池列表 ImageStores、磁盘卷存储池列表 VolumeStores、网络资源列表 PublicNetworks. ComputeNode 表示安装虚拟化软件的物理节点,为虚拟机提供运行环境,描述了物理节点的计算资源等信息. PublicImage 表示公共虚拟机映像文件,可供所有项目使用,描述了映像文件的操作系统类型等基本信息. ServerType 表示公共虚拟机资源配置类型,可供所有项目使用,描述了虚拟机 CPU、内存、磁盘等配置信息. ImageStore 表示虚拟机映像文件的存储池提供了 NFS、Amazon S3 等多种形式的存储管理. VolumeStore 则表示虚拟机磁盘卷的存储池,包含 SCSI、FC-SAN 等多种形式. PublicNetwork 表示云平台拥有的公共网络资源,包含 IP 网段列表 IpRanges,描述了可供使用的公共 IP 资源. 与使用者模型类似,主要受管单元也包含 cloudName 和 dataCenterId 属性,分别指出该受管单元所属的云平台及数据中心.

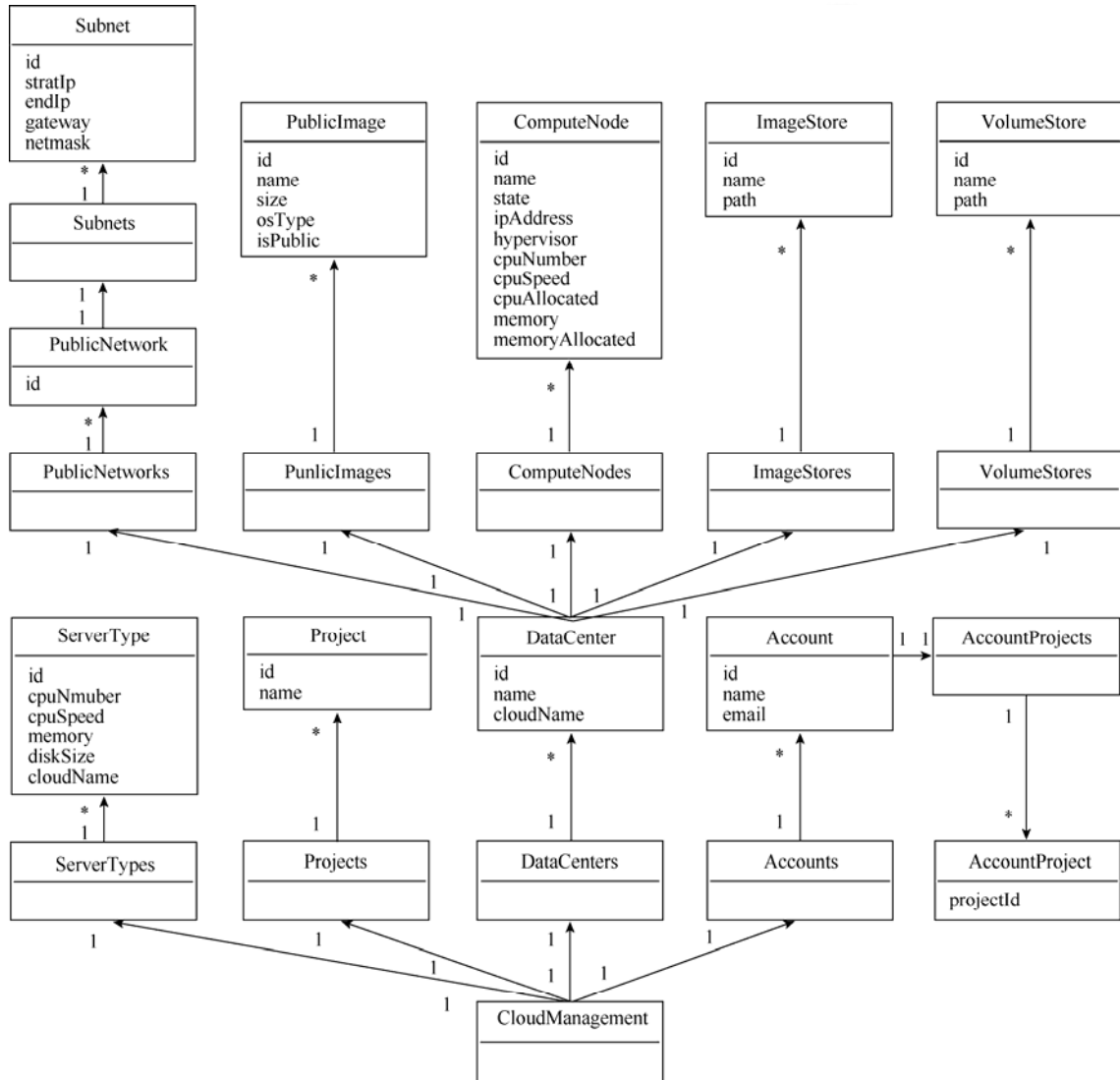


Fig.4 The administrator model of cloud software architecture

图 4 云平台软件体系结构的管理员模型

4 统一模型到单一云平台运行时模型的映射方法

统一模型为云计算平台的使用和管理分别提供了统一视图,为了进一步使使用者和管理员能够通过统一

模型对云资源进行操作,需要实现统一模型到云平台运行时模型的映射.本文提出一种模型转换方法,管理员仅需定义统一模型到云平台运行时模型的映射规则,就能够通过模型操作的转换,实现模型间的同步关系.

4.1 模型元素的映射关系

映射规则用于描述统一模型到云平台运行时模型的元素映射关系,任何一个统一模型中的元素属性与一个云平台运行时模型中相应的元素属性保持值的对应,且任何一组统一模型上的模型操作转换为的一组云平台运行时模型上相应的模型操作,以达到预期的管理效果.模型元素间存在 3 种基本映射关系,其他映射关系均可表示为以下 3 种基本映射关系的组合.

(1) 模型元素间“一对一”映射关系

统一模型中的一个元素与云平台运行时模型中的一个元素对应,特别地,统一模型中元素的属性可以在云平台运行时模型中对应的元素中找到对应的属性.它们通常是指统一模型中存在一种元素,云平台运行时模型中也存在一种元素,它们均是为了描述同一类型的事物.如图 5 所示,统一模型中的 ServerType 与云平台运行时模型中的 InstanceType 均描述虚拟机的资源配置信息,它们是“一对一”的映射关系.

(2) 模型元素间“多对一”映射关系

统一模型中的两个或多个元素与云平台运行时模型中的一个元素对应,特别地,云平台运行时模型中元素的属性在统一模型中的对应属性分布在两个或多个元素中.它们通常是指统一模型中存在两种或两种以上的元素,共同描述某一事物,而云平台运行时模型中仅用一种元素来描述这一事物.如图 5 所示,统一模型中的 Nic 与 PublicIp 均用来描述虚拟机的网络配置信息,而云平台运行时模型中仅 Nic 用来描述其信息,它们是“多对一”的映射关系.

(3) 模型元素间“一对多”映射关系

统一模型中的一个元素与云平台运行时模型中的两个或多个元素对应.它们通常是指统一模型中存在一种元素,用来描述某一类事物,而云平台运行时模型中的两种或多种类型的元素分别描述该类事物的不同子类型事物.如图 5 所示,统一模型中的 Server 用来表示虚拟机,其属性 isHA 描述虚拟机是否具备高可用功能,而云平台运行时模型中 VirtualMachine 与 VirtualMachineHA 则均用来表示虚拟机,分别描述普通虚拟机和高可用虚拟机,它们是“一对多”的映射关系.

	“一对一”映射关系	“多对一”映射关系	“一对多”映射关系
统一模型	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> ServerType id cpuNumber cpuSpeed memory diskSize cloudName </div>	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px; width: 40%;"> PublicIp id publicAddress associateNicId </div> <div style="border: 1px solid black; padding: 5px; width: 40%;"> Nic id networkId ipAddress macAddress </div> </div>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> Server id name imageId serverTypeId cpuNumber cpuSpeed memory diskSize isHA </div>
云平台运行时模型	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> InstanceType id vCpus cpuSpeed memory diskSize </div>	<div style="border: 1px solid black; padding: 5px; width: fit-content;"> Nic id subnetId privateIpAddress elasticIpAddress macAddress </div>	<div style="display: flex; justify-content: space-around;"> <div style="border: 1px solid black; padding: 5px; width: 45%;"> VirtualMachine id name imageId serverTypeId cpuNumber cpuSpeed memory diskSize </div> <div style="border: 1px solid black; padding: 5px; width: 45%;"> VirtualMachineHA id name imageId serverTypeId cpuNumber cpuSpeed memory diskSize </div> </div>

Fig.5 Three types of basic mapping relationships between model elements

图 5 模型元素间的 3 种映射关系

此外,统一模型中的元素属性值与云平台运行时模型中的元素属性值也存在映射关系,它们通常是指,同一种功能或内涵的配置参数或系统指标,在统一模型和云平台运行时模型中用不同的值表示.例如,虚拟机配置信息 CPU:2.5GHz;Memory:1G 在统一模型中表示成 ServerTypeId 的值为 small,在云平台运行时模型中则表示成

ServiceOffering 的值为 a7d50774-d553-4ea6-b03e-6a168af2de14.因此,在统一模型到云平台运行时模型的映射过程中,也需要保持其值的对应关系.

在之前的工作^[20]中,我们已经研究了两个模型间映射规则的定义方式,本文不再详述.在第 5.2 节中,我们将针对实际应用场景,举例进行说明.

4.2 模型操作的映射规则

云平台使用者和管理员通过模型操作进行管理任务的执行.云平台的管理任务本质上是云资源的增减及其属性的查改.分别对应模型元素的 List,Add 和 Remove 操作及其属性的 Get 和 Set 操作.为了实现统一模型到云平台运行时模型的映射,需要将任何一个统一模型上的模型操作转换为一个云平台运行时模型上对应的模型操作,以达到预期的管理效果.如表 1 所示,我们定义了模型操作的映射规则.

Table 1 Mapping rules of model operation

表 1 模型操作的映射规则

	One-to-One mapping rule	Many-to-One mapping rule	One-to-Many mapping rule
Example	$A \rightarrow B$ $A1.a1 \rightarrow B1.b1$	$A \rightarrow B$ $A1.a1 \rightarrow B1.b1$ $C1.c1 \rightarrow B1.b2$	$A \rightarrow B$ or C $A1.a1 \rightarrow B1.b1$ $A2.a1 \rightarrow C1.c1$
Get	$Get A1.a1 \rightarrow Get B1.b1$	$Get A1.a1 \rightarrow Get B1.b1$ $Get C1.c1 \rightarrow Get B1.b2$	$Get A1.a1 \rightarrow Get B1.b1$ $Get A2.a1 \rightarrow Get C1.c1$
Set	$Set A1.a1 \rightarrow Set B1.b1$	$Set A1.a1 \rightarrow Set B1.b1$ $Set C1.c1 \rightarrow Set B1.b2$	$Set A1.a1 \rightarrow Set B1.b1$ $Set A2.a1 \rightarrow Set C1.c1$
List	$List^*A \rightarrow List^*B$ <u>$Get A.properties \rightarrow Get B.properties$</u>	$List^*A \rightarrow List^*B$ <u>$Get A.properties \rightarrow Get B.properties$</u>	$List^*A \rightarrow List^*B$ and $List^*C$ <u>$Get A.properties \rightarrow Get B.properties$</u> or <u>$Get C.properties$</u>
Add	$Add^*A \rightarrow Add^*B$ <u>$Set A.properties \rightarrow Set B.properties$</u>	$Add^*A \rightarrow Add^*B$ <u>$Set A.properties \rightarrow Set C.properties$</u> and <u>$Set B.properties$</u>	$Add^*A \rightarrow Add^*B$ or Add^*C <u>$Set A.properties \rightarrow Set B.properties$</u> or <u>$Set C.properties$</u>
Remove	$Remove^*A \rightarrow Remove^*B$	$Remove^*A \rightarrow Remove^*B$	$Remove^*A \rightarrow Remove^*B$ or $Remove^*C$

(1) 模型元素间“一对一”映射关系

存在统一模型中元素 A 与云平台运行时模型中元素 B 对应;那么,对元素 A 的 add, remove 或 list 操作转换成对元素 B 的相同操作,对元素 A 属性的 get 或 set 操作则转换成对元素 B 对应属性的相同操作.

(2) 模型元素间“多对一”映射关系

存在统一模型中的元素 A 与云平台运行时模型中元素 B 对应,且元素 B 的某些属性同时与统一模型中 C 元素的属性关联;那么,对元素 A 和元素 C 属性的 get 或 set 操作转换成对元素 B 对应属性的 get 或 set 操作,对元素 A 的 add, remove 或者 list 操作则转化成对元素 B 的相同操作.特别地,当元素 B 被创建时,属性的初始值同时从元素 A 和元素 C 中获取.

(3) 模型元素间“一对多”映射关系

存在统一模型中的元素 A 与云平台运行时模型中元素 B 或元素 C 对应;那么,对元素 A 及其属性的操作转换成对其对应元素和属性的相同操作.特别地,对元素 A 的 list 操作则转换成同时对元素 B 和元素 C 执行 list 操作.

4.3 模型操作的描述方式

通过模型操作的转换,任何作用在统一模型上的操作将映射为作用在云平台运行时模型上的对应操作.如图 6 所示,这些模型操作将以 XML 文件的形式进行描述,同时,根据云平台基本信息,传送到目标云平台运行时模型并执行,以实现预期的管理效果.

List	Description	<pre><action node="TypeS" type="List"> <query node="TypeF" condition="Constraint" /> </action> <return> <node="TypeS" condition="Constraint1" /> <node="TypeS" condition="Constraint2" /> </return></pre>
	Pre-Condition	$\exists \text{TypeS } s, \exists \text{TypeF } f, s \in f \wedge f \text{ in condition of Constraint}$
	Illustration	Find the "TypeF" element which satisfies the constraints and list "Types" elements which are its child nodes.
Get	Description	<pre><action key="KEY" type="get"> <query node="Type" condition="Constraint" /> </action> <return key="KEY" value="VALUE" /></pre>
	Pre-Condition	$\exists \text{Type } n, n \text{ in condition of Constraint} \wedge \text{"KEY"} \in \text{Type.attributes}$
	Illustration	Find the "Type" element which satisfies the constraints and get the value of its "KEY" attribute.
Add	Description	<pre><action node="TypeS" type="add"> <query node="TypeF" condition="Constraint" /> <set key="KEY1" value="VALUE1" /> <set key="KEY2" value="VALUE2" /> </action></pre>
	Pre-Condition	$\exists \text{TypeF } f, f \text{ in condition of Constraint} \wedge \text{"KEY1"}, \text{"KEY2"} \in \text{TypeS.attributes}$
	Illustration	Find the "Type" element which satisfies the constraints and add a "TypeS" element as its child node.
Set	Description	<pre><action key="KEY" value="VALUE" type="set"> <query node="Type" condition="Constraint" /> </action></pre>
	Pre-Condition	$\exists \text{Type } n, n \text{ in condition of Constraint} \wedge \text{"KEY"} \in \text{Type.attributes}$
	Illustration	Find the "Type" element which satisfies the constraints and set the value of its "KEY" attribute to "VALUE".
Remove	Description	<pre><action node="Type" condition="Constraint" type="remove" /></pre>
	Pre-Condition	$\exists \text{Type } n, n \text{ in condition of Constraint}$
	Illustration	Find the "Type" element which satisfies the constraints and remove it.

Fig.6 Five types of model operations

图 6 5 种模型操作

5 实例研究

为了满足遗产系统整合和动态资源扩展等需求,一些大型企业常常需要构造混合云来统一管理不同云平台中的计算和存储资源.然而,目前尚不存在针对混合云管理的成熟的开源解决方案.为了验证本文方法的可行性和有效性,我们针对 Amazon EC2 公有云服务^[2]和基于 CloudStack^[3]的企业私有云,从云平台使用者的角度,构造混合云管理系统,实现了面向云平台统一模型的管理程序开发.

5.1 CloudStack 与 Amazon EC2 运行时模型的构造

CloudStack 是一种开源的云平台管理软件,常用于企业私有云管理.图 7 所示模型图(上)描述了 CloudStack

软件体系结构模型(系统元模型)中的主要受管单元,包括项目(Project)、计算配置方案(ServiceOffering)、磁盘配置方案(DiskOffering)、虚拟机映像文件(Template)、虚拟机(VirtualMachine)和磁盘卷(Volume)等资源。

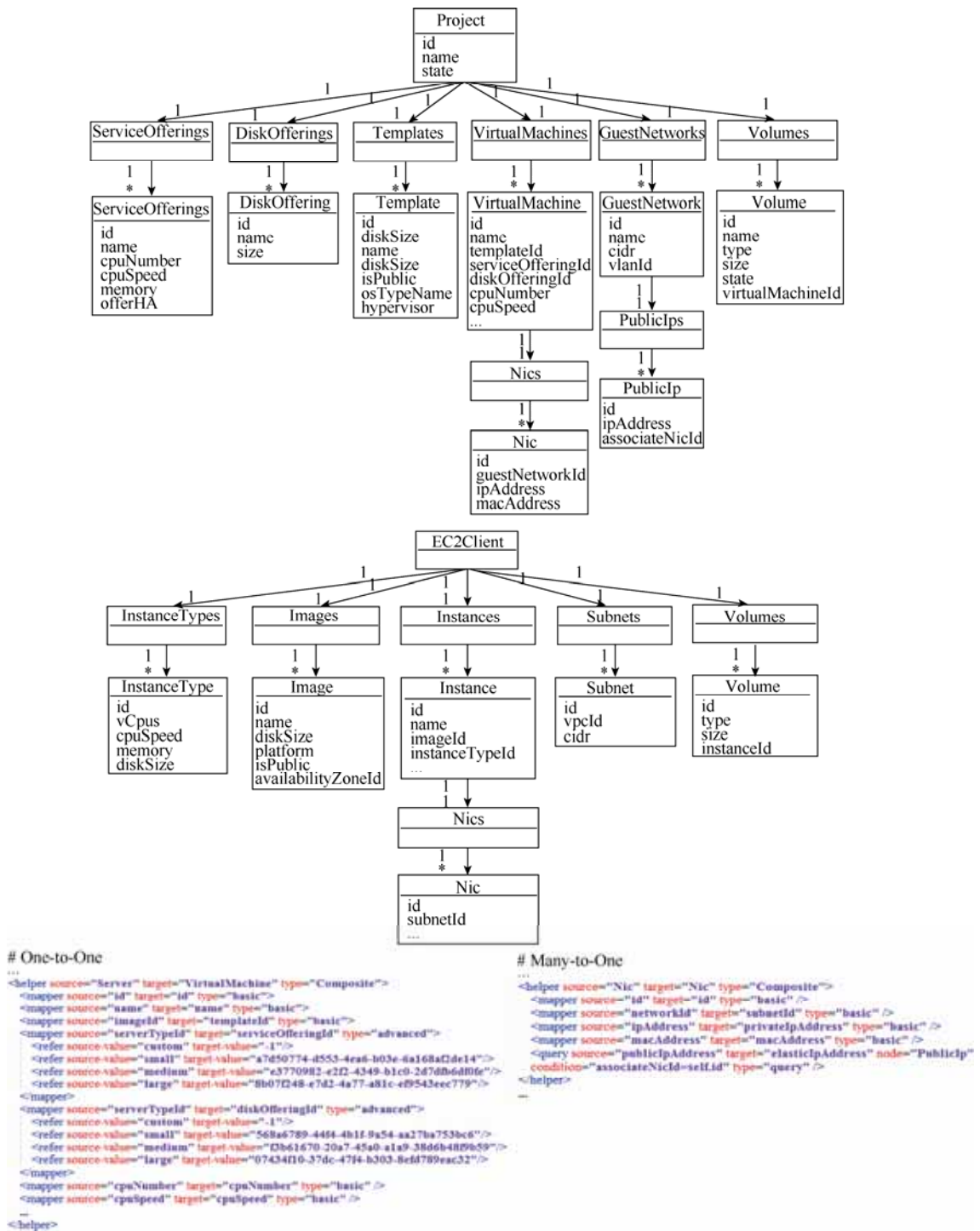


Fig.7 Models of CloudStack (up) and Amazon EC2 (down) and mapping relationships between model elements
图 7 CloudStack 模型(上)、Amazon EC2 模型(下)及模型元素间的映射关系

图 7 中,Project 表示项目,描述了项目的基本信息,包含计算配置方案列表 ServiceOfferings、磁盘配置方案列表 DiskOfferings、虚拟机映像文件列表 Templates、虚拟机列表 VirtualMachines、网络子网列表 GuestNetworks 和磁盘卷列表 Volumes.Template 表示项目可以使用的虚拟机映像文件,描述其操作系统信息.ServiceOffering 表示虚拟机计算配置方案,描述其 CPU 核数、内存大小等计算资源配置信息.DiskOffering 表示虚拟机磁盘配置方案,描述其磁盘大小等存储资源配置信息.GuestNetwork 表示网络子网,描述该子网的基本信息,并包含该子网拥有的公共 IP 地址的列表 PublicIps;PublicIp 表示可供外部网络访问的 IP 地址,为子网中拥有该 IP 地址的虚拟机提供外部网络的访问入口.Volume 表示磁盘卷,可为虚拟机增加额外的存储.VirtualMachine 表示虚拟机,描述虚拟机使用的映像文件、计算及存储配置方案等基本信息.同时,VirtualMachine 还包含虚拟机使用的虚拟网卡列表 Nics;Nic 表示虚拟网卡,描述其 IP 地址、MAC 地址及所在的网络子网等网络配置信息.

Amazon EC2(Amazon Elastic Compute Cloud)公有云服务允许使用者按需租用虚拟机来搭建应用系统.图 7 所示模型图(下)描述了 Amazon EC2 软件体系结构模型(系统元模型)中的主要受管单元,包括资源配置类型(InstanceType)、虚拟机映像文件(Image)、虚拟机(Instance)、网络子网(Subnet)及磁盘卷(Volume)等.其中,EC2Client 是模型的根元素,包含资源配置类型列表 InstanceTypes、虚拟机映像列表 Images、虚拟机列表 Instances、网络子网列表 Subnets 及磁盘卷列表 Volumes.InstanceType 表示虚拟机资源配置类型,描述了 CPU 核数、内存大小、存储空间等资源配置信息.Image 表示虚拟机映像文件,是虚拟机软件系统的载体.Subnet 表示某个网络子网,为虚拟机提供基本的网络服务.Volume 表示可定制的用于持久性数据存储的磁盘卷,可为虚拟机增加额外的存储空间.Instance 表示虚拟机,描述了虚拟机使用的映像文件、资源配置类型等基本信息.同时,Instance 包含其使用的虚拟网卡列表 Nics;Nic 表示虚拟网卡,描述虚拟网卡的 IP 地址、MAC 地址及所在的网络子网等网络配置信息.

给定 CloudStack 和 Amazon EC2 系统元模型,仍需定义其上的模型操作^[21],即访问模型.CloudStack 和 Amazon EC2 拥有大量的管理接口,我们通过定义模型操作到这些管理接口的映射规则,来对它们建模.由于篇幅限制,访问模型的具体构建过程请查阅我们之前的工作^[15,16,19].在系统元模型和访问模型的基础上,SM@RT 工具能够自动生成模型转换程序,以保障系统运行时模型与运行系统的双向同步.于是,能够在模型层对 CloudStack 和 Amazon EC2 进行使用和管理.

5.2 统一模型到云平台运行时模型的映射

如图 3 所示,统一模型(使用者模型)为云资源的使用提供了统一的全局视图.为了进一步使使用者能够通过统一模型对基于 CloudStack 的企业私有云及 Amazon EC2 公有云服务的计算、存储资源进行操作,需要实现统一模型到 CloudStack 和 Amazon EC2 运行时模型的映射.根据模型间的元素映射关系,我们定义了统一模型到云平台运行时模型的映射规则.其中,云平台虚拟机及网卡在模型间的映射关系是映射规则描述的关键点.下面以统一模型中 Server 到 CloudStack 运行时模型中 VirtualMachine 的“一对一”映射,以及统一模型中 Nic 和 PublicIp 到 CloudStack 运行时模型中 Nic 的“多对一”映射为例,详细介绍映射规则的描述方式.

1) 统一模型中 Server 与 CloudStack 运行时模型中 VirtualMachine 均表示虚拟机,它们存在“一对一”映射关系,图 7 所示映射规则描述片段中通过 helper 标签表示两个元素间的映射关系.Server 中 id、name、imageId、cpuNumber、cpuSpeed、cpuUsed 及 diskSize 等属性与 VirtualMachine 中 id、name、templateId、cpuNumber、cpuSpeed、cpuUsed 及 diskSize 等属性存在对应关系,描述片段中通过 mapper 标签表示两两属性间的映射关系.其中,当 mapper 标签的 type 为“basic”时,表示对应属性的属性值相等.例如,当 Server 的属性 cpuNumber 值为 3 时,VirtualMachine 对应的属性 cpuNumber 值也为 3.而当 mapper 标签的 type 为“advanced”时,表示对应属性的属性值需要进行转换.例如,当 Server 的属性 serverTypeId 值为“small”时,VirtualMachine 对应的属性 serviceOfferingId 值为“e3770982-e2f2-4349-b1c0-2d7dfb6df0fe”.

2) 统一模型中 Nic 表示虚拟网卡,描述了虚拟机所在网络子网、子网 IP 地址等网络配置信息,PublicIp 表示可供外部网络访问的 IP 地址,并描述了关联的 Nic 信息;而 CloudStack 运行时模型中 Nic 则描述了虚拟机所在网络子网、子网 IP 地址、外网 IP 地址等全部网络配置信息.因此,统一模型中 Nic 和 PublicIp 到 CloudStack

运行时模型中 Nic 是“多对一”映射关系.在图 7 所示的映射规则描述片段中,通过 helper 标签表示统一模型中 Nic 到 CloudStack 运行时模型中 Nic 的映射关系,通过 mapper 标签表示两两属性间的映射关系.特别地,通过 query 标签表示统一模型中与 Nic 关联的 PublicIp 属性 publicIpAddress 到 CloudStack 运行时模型中 Nic 属性 elasticIpAddress 的映射.

根据映射规则,作用在统一模型上的模型操作将转换为作用在 CloudStack 或 Amazon EC2 运行时模型运行时模型上对应的模型操作.图 8 展示了统一模型上的虚拟机创建操作转换为 CloudStack 运行时模型上对应的模型操作并加以执行的过程,虚拟机创建操作描述如下.(1) Query:查询一个 Servers 元素,其 projectId 为“2edc7933-0a09-46eb”.(2) Add:创建一个 Server 元素.(3) Set:为 Server 元素的属性赋值.

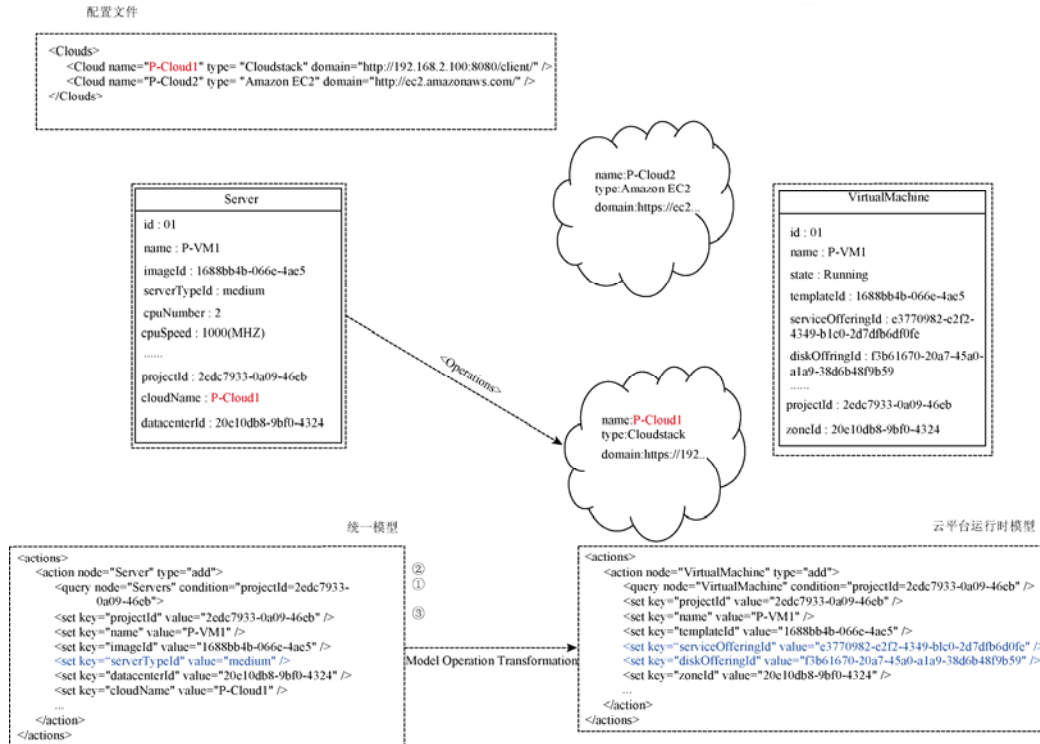


Fig.8 An example of model operation transformation

图 8 模型操作转换实例

统一模型中待创建的 Server 元素的 cloudName 属性值为“P-Cloud1”,根据配置文件可知,目标云平台类型是 CloudStack.因此,Server 创建任务将转换为 CloudStack 运行时模型上的 VirtualMachine 创建任务.根据映射规则,模型操作会逐条进行转换:统一模型中 Servers 元素的查询操作将转换为 CloudStack 运行时模型中 VirtualMachines 元素的查询操作,其 projectId 为“2edc7933-0a09-46eb”;统一模型中 Server 元素的创建操作将转换为 CloudStack 运行时模型中 VirtualMachine 元素的创建操作;统一模型中 Servers 元素属性的赋值操作将转换为 CloudStack 运行时模型中 VirtualMachine 元素对应属性的赋值操作.特别地,赋值操作的转换需要遵循属性值的映射规则.例如,统一模型中 Server 元素 serverTypeId 属性的值为 medium,那么,CloudStack 运行时模型中 VirtualMachine 元素 serviceOfferingId 属性的值为 e3770982-e2f2-4349-b1c0-2d7dfb6df0fe,diskOfferingId 属性的值为 f3b61670-20a7-45a0-a1a9-38d6b48f9b59.最后,生成的模型操作文件将传送到 CloudStack 运行时模型并执行,以实现预期的管理效果.

5.3 方法评估

我们从以下 3 方面对方法进行评估.

5.3.1 混合云运行时软件体系结构模型的构造工作评估

在单一云平台运行时模型的构造过程中,开发人员仅需使用 SM@RT 建模语言定义系统元模型和系统访问模型,SM@RT 代码生成器就能够自动生成同步引擎,支持运行时模型与运行系统的双向同步,实现在模型层对单一云平台进行管理.在前期工作中,我们研究了系统元模型的推理方法^[18],通过分析调用管理 API 的客户端代码,实现系统元模型的自动构造,该工作能够进一步降低单一云平台运行时模型的构造难度.此外,单一云平台运行时模型的构造工作是一次性的,它与管理 API 一样可以在不同的云管场景中复用.因此,从管理功能复用的角度,构造单一云平台运行时模型的额外工作是可以接受的.

在单一云平台运行时模型的基础上,开发人员仅需定义统一模型到云平台运行时模型的映射规则,就能够面向统一模型进行混合云管理程序的开发.统一模型描述了常见云平台共有的资源类型及管理操作,能够对计算、存储、网络等资源进行统一管理.为了验证统一模型对云平台管理功能的覆盖程度,我们将统一模型提供的管理功能与多个云平台的资源管理 API 进行比较.如图 9 所示,云平台共有的资源管理 API 可以覆盖各云平台的核心管理功能,且其数量均超过各云平台资源管理 API 的 70%,满足对云平台进行统一管理的需求.在前期工作中,我们研究了基于模型的多样化云资源集成管理方法^[20],通过模型转换,实现单一云平台中系统管理功能的复用和集成,该工作支持云平台个性化管理功能的集成,能够进一步增强统一模型在不同管理场景中的适应能力.因此,从管理程序开发的角度,混合云运行时模型的构造代价是可以接受的,其适用程度是满足要求的.

云平台共有资源管理API示例			
Compute	Server	CreateServer	创建虚拟机
		DestroyServer	销毁虚拟机
		UpdateServer	修改虚拟机
		ListServers	查询虚拟机
		StartServer	启动虚拟机
		StopServer	关闭虚拟机
		RebootServer	重启虚拟机
		MigrateServer	热迁移虚拟机
	ResizeServer	更改虚拟机硬件配置	
	Image	CreateImage	创建虚拟机模板映像
		DeleteImage	删除虚拟机模板映像
		UpdateImage	修改虚拟机模板映像
		ListImages	查询虚拟机模板映像
ServerType	ListServerTypes	查询虚拟机硬件配置方案	
Storage	Volume	CreateVolume	创建磁盘卷
		DeleteVolume	删除磁盘卷
		ListVolumes	查询磁盘卷
		AttachVolume	附加磁盘卷到某个虚拟机
		DetachVolume	从某个虚拟机卸载磁盘卷
		AddNicToServer	添加网卡到某个虚拟机
Network	Nic	RemoveNicFromServer	从某个虚拟机卸载网卡
		ListNics	查询虚拟网卡
	Network	CreateNetwork	创建虚拟网络
		DeleteNetwork	删除虚拟网络
		UpdateNetwork	修改虚拟网络
		ListNetworks	查询虚拟网络
		ListPublicIps	查询公共IP
	PublicIp	AssociatePublicIpWithServer	绑定公共IP到某个虚拟机
		DisassociatePublicIpFromServer	从某个虚拟机撤销公共IP

云平台共有资源管理API的覆盖程度			
	OpenStack	CloudStack	Amazon EC2
Compute	81.50%	84.20%	73.90%
Storage	70%	71.40%	75%
Network	92.90%	76.90%	79.20%

Fig.9 Statistics of common APIs for cloud resource management

图 9 云平台共有资源管理 API 统计

5.3.2 基于模型语言与通用语言的管理程序开发难度比较

为了验证本文方法,我们针对一组常见的混合云管理任务,基于模型语言 QVT^[21]和通用语言 Java 分别实现了其管理程序.图 10 展示了混合云中虚拟机 CPU 负载报警任务的 QVT 和 Java 管理程序,其中,Java 程序需要 200 多行,而 QVT 程序仅需要 3 行,与 Java 程序相比,QVT 程序的难度和复杂度都要小得多.一方面,混合云软件体系结构模型对云平台管理接口进行复用,开发人员不用处理管理接口调用及底层数据交互等编程工作;另一

方面,模型语言提供了一些模型层的复杂操作,例如,“select”用于选出符合某种条件的所有模型元素,这些复杂操作进一步降低了编程难度和复杂度.表 2 对完成相同一组混合云管理任务的 QVT 和 Java 程序进行比较.任务 1 是列出混合云中的所有虚拟机,Java 代码行数为 205 行,而 QVT 代码行数为 3 行;任务 2 是根据混合云负载,选择一个云平台创建虚拟机,Java 代码行数为 223 行,而 QVT 代码行数为 17 行;任务 3 是混合云中虚拟机 CPU 负载查询,Java 代码行数为 218 行,而 QVT 代码行数为 3 行;任务 4 与任务 5 分别是混合云中所有虚拟机磁盘使用量和网络使用量的计算.同样地,QVT 程序的代码行数也远小于 Java 程序.



Fig.10 Programs of VM monitoring in the languages of QVT and Java

图 10 分别用 QVT 与 Java 实现的虚拟机监测程序

Table 2 Comparison of LOC between the QVT and Java programs

表 2 QVT 与 Java 程序开发难度比较

管理任务	虚拟机的展示	虚拟机的创建	虚拟机 CPU 负载报警	虚拟机磁盘使用量计算	虚拟机网络使用量计算
代码量(行)	QVT 3 Java 205	QVT 17 Java 223	QVT 3 Java 218	QVT 13 Java 241	QVT 13 Java 241

5.3.3 基于运行时模型与管理接口的管理程序执行性能比较

为了比较执行性能,我们在虚拟机使用数量为 10 台、20 台和 50 台的情况下,分别执行 QVT 和 Java 管理程序,完成相同一组混合云管理任务.实验过程中,虚拟机在 CloudStack 私有云和 Amazon EC2 公有云中占比分别为 80%和 20%.如表 3 所示,QVT 程序的执行时间均会略高于 Java 程序.任务 1、任务 2 均未对运行的虚拟机进行操作,其管理接口的调用次数不随虚拟机数量的增长而发生变化,它们的执行时间也基本不变.任务 3~任务 5 均需要获取每一台运行虚拟机的属性值,其执行时间与虚拟机数量呈线性增长,QVT 与 Java 程序的执行时间差也随之增大.其主要原因是,QVT 和 Java 程序从本质上均是通过调用云平台管理接口来实现特定的管理功能;而模型方法还需要额外的操作来维护统一模型与运行时模型,以及运行时模型与底层系统间的同步机制.因此,QVT 和 Java 程序执行时间的差异与管理接口的调用次数呈线性增长.然而,与管理程序的执行时间相比,它们的差异并不大.特别地,从系统管理的角度看,这种性能上的差异是可被接受的.

Table 3 Comparison of performance between the QVT and Java programs

表 3 QVT 与 Java 执行性能比较

管理任务	虚拟机的展示	虚拟机的创建	虚拟机 CPU 负载报警	虚拟机磁盘使用量计算	虚拟机网络使用量计算
10 台	QVT 执行时间(ms)	117	118	1 148	1 183
	Java 执行时间(ms)	74	86	749	738
	API 调用次数	1	1	10	10
20 台	QVT 执行时间(ms)	160	128	2 386	2 413
	Java 执行时间(ms)	113	79	1 526	1 493
	API 调用次数	1	1	20	20

Table 3 Comparison of performance between the QVT and Java programs (Continued)

表 3 QVT 与 Java 执行性能比较(续)

管理任务	虚拟机的展示	虚拟机的创建	虚拟机 CPU 负载报警	虚拟机磁盘使用量计算	虚拟机网络使用量计算
50 台 QVT 执行时间(ms)	247	131	5 876	6 129	5 812
Java 执行时间(ms)	202	81	3 826	3 784	3 816
API 调用次数	1	1	50	50	50

6 相关工作

目前存在许多云平台管理工具用于不同类型云资源的管理,如,OpenStack^[4]、Eucalyptus^[5]用于管理基础设施层的云资源,Tivoli^[22]、Hyperic^[23]用于管理平台层的云资源.然而,这些管理工具缺乏完善的混合云支撑机制.

近年来,存在许多混合云管理的研究工作,包括混合云构造^[24-26]、混合云应用部署^[27,28]及混合云资源调度^[29-31].文献[24]提出一种混合云虚拟化基础设施的管理方法,通过 OpenNebula^[32]实现单个云平台虚拟化基础设施的管理及外部云平台核心管理 API 的集成,并提供了一套混合云虚拟化资源的调度机制.文献[25]提出一种 IaaS(infrastructure as a service)云服务的抽象模型及一组核心管理 API 的抽象接口,并在多云环境中构造了统一管理实例,文献[26]在此基础上进一步实现了一组高级管理功能.上述工作对混合云构造方法进行了初步探索,然而,方法中的统一管理接口是通过直接封装云平台核心管理 API 获得,因此,其工作量大且可扩展性差.文献[27,28]在混合云虚拟化资源管理的基础上,增加了应用部署和配置功能,实现了混合云中的应用自动部署.文献[29-31]在混合云虚拟化资源管理的基础上,面向特定应用场景研究调度策略,实现了混合云中的资源管理.本文在常见云平台共有的资源类型及管理操作的基础上,提出一种 IaaS 云服务的抽象模型,通过模型转换实现多个云平台核心管理功能的快速集成,其工作量小;基于前期工作^[20],还能够支持云平台个性化管理功能的集成,可扩展性好.此外,本文方法为混合云应用部署及资源调度提供了虚拟化资源的统一管理能力,而以模型为中心的的分析方法与支撑机制^[33]能够进一步为混合云管理程序的开发提供帮助.

运行时模型被广泛应用在不同类型的软件系统中,以支持数据操作^[34]、系统自修复^[35]和动态自适应^[36-39]等管理功能.前期工作中,我们在运行时模型理论及构造方法方面进行了研究:给定系统元模型与一组管理接口,SM@RT 工具^[15]就能自动生成代码,在保证性能的前提下实现模型到管理接口的映射;当系统元模型发生变化时,SM@RT 可以自动生成新的映射代码;文献[16]对以上内容进行了详细的论述.我们还研究了系统元模型的推理方法,通过分析调用管理 API 的客户端代码,实现系统元模型的自动构造^[18].同时,为了弥补建模语言本身的非完全形式化问题,我们在模型分析及模型容错方面也进行了研究:提出一种 MOF 元模型扩展机制^[40]以支持元模型的向上兼容,从而实现在模型集成过程中模型的自动转换.该方法在体系结构级别的系统容错实践^[41]中进一步得到了验证.我们还构建了云平台运行时体系结构模型^[19],对运行时模型的性能进行了验证,并尝试基于模型语言实现系统自适应管理.进一步地,提出一种基于模型的多样化云资源集成管理方法^[20],通过模型转换实现单一云平台中系统管理功能的复用和集成.本文方法建立在以上前期工作的基础上.

7 结束语

云平台管理接口和管理机制的异构性,给混合云管理系统的开发带来了极大的难度和复杂度.本文提出一种基于运行时模型的混合云管理方法:开发人员仅需定义统一模型与云平台运行时模型间的元素映射关系,任何统一模型上的管理操作就能够自动转换为云平台运行时模型上对应的管理操作,并最终作用到云计算系统上.于是,开发人员能够面向统一模型进行管理程序的开发,而不用处理管理接口调用及底层数据交互等繁杂、琐碎的编程工作.本文方法能够降低混合云管理系统开发的难度和复杂度.

未来工作的重点主要包含两个方面:一方面,将方法运用到遗产系统整合和动态资源扩展等混合云实际管理场景中,并完善特定场景下的支撑机制;另一方面,在方法基础上进行管理风格的研究,基于模型分析、推理等技术实现系统容错、安全监控等高级管理功能.

References:

- [1] Mell P, Grance T. The NIST definition of cloud computing. *Communications of the ACM*, 2011,53(6):50.
- [2] Amazon Web Services (AWS). Cloud Computing Services. 2017. <http://aws.amazon.com/>
- [3] Kumar R, Jain K, Maharwal H, Jain N, Dadhich A. Apache CloudStack: Open source infrastructure as a service cloud computing platform. *Int'l Journal of Advancement in Engineering Technology, Management and Applied Science*, 2014,681–684. [doi: 10.13140/2.1.1290.0483]
- [4] OpenStack. The open source cloud operating system. 2013. <http://openstack.org/projects/>
- [5] Nurmi D, Wolski R, Grzegorzczak C, Obertelli G, Soman S, Youseff L, Zagorodnov Z. The eucalyptus open-source cloud-computing system. In: *Proc. of the 9th IEEE/ACM Int'l Symp. on Cluster Computing and the Grid*. Shanghai: IEEE Press, 2009. 124–131. [doi: 10.1109/CCGRID.2009.93]
- [6] Garlan D. Software architecture: A roadmap. In: *Proc. of the 22nd Int'l Conf. on Software Engineering, Future of Software Engineering Track*. New York: ACM Press, 2000. 91–101. [doi: 10.1145/336512.336537]
- [7] Mei H, Shen JR. Progress of research on software architecture. *Ruan Jian Xue Bao/Journal of Software*, 2006,17(6):1257–1275 (in Chinese with English abstract). <http://www.jos.org.cn/1000-9825/17/1257.htm> [doi: 10.1360/jos171257]
- [8] France R, Rumpe B. Model-Driven development of complex software: A research roadmap. In: *Proc. of the 29th Int'l Conf. on Software Engineering*. Minneapolis: IEEE Press, 2007. 37–54. [doi: 10.1109/FOSE.2007.14]
- [9] Huang G, Ma XX, Tsai WT. A new software paradigm for Internet computing. *National Science Review*, 2014,1(2):168–169. [doi: 10.1093/nsr/nwt014]
- [10] Bencomo N, Blair G, France R. Summary of the workshop models@run.time at MoDELS 2006. *Models in Software Engineering*, 2006,4364:227–231. [doi: 10.1007/978-3-540-69489-2_28]
- [11] Blair G, Bencomo N, France R. Models@ run.time. *Computer*, 2009,42(10):22–27. [doi: 10.1109/MC.2009.326]
- [12] Huang G, Mei H, Yang FQ. Runtime recovery and manipulation of software architecture of component-based systems. *Automated Software Engineering*, 2006,13(2):257–281. [doi: 10.1007/s10515-006-7738-4]
- [13] Occello A, Dery-Pinna A, Riveill M. A runtime model for monitoring software adaptation safety and its concretisation as a service. In: *Models@ runtime*, Toulouse, 2008,8:67–76.
- [14] Wu YH, Huang G, Song H, Zhang Y. Model driven configuration of fault tolerance solutions for component-based software system. In: *Proc. of the 15th Int'l Conf. on Model Driven Engineering Languages and Systems*. Innsbruck: Springer-Verlag, 2012. 514–530. [doi: 10.1007/978-3-642-33666-9_33]
- [15] Huang G, Song H, Mei H. SM@RT: Applying architecture-based runtime management of internetware systems. *Int'l Journal of Software and Informatics*, 2009,3(4):439–464. [doi: 10.1145/1640206.1640215]
- [16] Song H, Huang G, Chauvel F, Xiong YF, Hu ZJ, Sun YC, Mei H. Supporting runtime software architecture: A bidirectional-transformation-based approach. *Journal of Systems and Software*, 2011,84(5):711–723. [doi: 10.1016/j.jss.2010.12.009]
- [17] Song H, Xiong YF, Chauvel F, Huang G, Hu ZJ, Mei H. Generating synchronization engines between running systems and their model-based views. In: *Models in Software Engineering (the MoDELS Workshops)*. Denver: Springer-Verlag, 2009. 140–154. [doi: 10.1007/978-3-642-12261-3_14]
- [18] Song H, Huang G, Xiong YF, Chauvel F, Sun YC, Mei H. Inferring meta-models for runtime system data from the clients of management APIs. In: *Proc. of the 13rd Int'l Conf. on Model Driven Engineering Languages and Systems*. Oslo: Springer-Verlag, 2010. 168–182. [doi: 10.1007/978-3-642-16129-2_13]
- [19] Huang G, Chen X, Zhang Y, Zhang XD. Towards architecture-based management of platforms in the cloud. *Frontiers of Computer Science*, 2012,6(4):388–397. [doi: 10.1007/s11704-012-2100-4]
- [20] Chen X, Zhang Y, Huang G, Zheng XH, Guo WZ, Rong CM. Architecture-Based integrated management of diverse cloud resources. *Journal of Cloud Computing: Advances, Systems and Applications*, 2014,3:11. [doi: 10.1186/s13677-014-0011-7]
- [21] Object Management Group. Meta object facility (MOF) 2.0 query/view/transformation (QVT). 2011. <http://www.omg.org/spec/QVT>
- [22] IBM. IBM Tivoli Software. 2011. <http://www-01.ibm.com/software/tivoli/>
- [23] SpringSource. Hyperic. 2013. <http://www.hyperic.com/>
- [24] Sotomayor B, Montero RS, Llorente IM, Foster I. Virtual infrastructure management in private and hybrid clouds. *IEEE Network Computing*, 2009,13(5):14–22. [doi: 10.1109/MIC.2009.119]
- [25] Lee BS, Yan SX, Ma D, Zhao GP. Aggregating IaaS service. In: *Proc. of the 2011 Annual SRII Global Conf*. San Jose: IEEE Computer Society Press, 2011,14:335–338. [doi: 10.1109/SRII.2011.44]
- [26] Yan SX, Lee BS, Zhao GP, Ma D, Mohamed P. Infrastructure management of hybrid cloud for enterprise users. In: *Proc. of the 5th Int'l DMTF Academic Alliance Workshop on Systems and Virtualization Management (SVM)*. Paris: IEEE Press, 2011. 1–6. [doi: 10.1109/SVM.2011.6096463]

- [27] Suzuki J, H.Phan D, Higuchi M, Yamano Y, Oba K. Model-Driven integration for a service placement optimizer in a sustainable cloud of clouds. In: Proc. of the Joint, Int'l Conf. on Soft Computing and Intelligent Systems. Kobe: IEEE Press, 2012. 301–306. [doi: 10.1109/SCIS-ISIS.2012.6505344]
- [28] Juve G, Deelman E. Automating application deployment in infrastructure clouds. In: Proc. of the 3rd IEEE Int'l Conf. on Cloud Computing Technology and Science. Athens: IEEE Computer Society Press, 2011. 658–665. [doi: 10.1109/CloudCom.2011.102]
- [29] Yue JY, Zhang Z, Fu JH, Lu SQ, Li XM, Shen YF. Extensible architecture for high-throughput task processing based on hybrid cloud infrastructure. In: Proc. of the 2011 Int'l Conf. on Electronics, Communications and Control (ICECC). IEEE Press, 2011. 1452–1455. [doi: 10.1109/ICECC.2011.6067604]
- [30] Ruben VDB, Vanmechelen K, Broeckhove J. Cost-Optimal scheduling in hybrid IaaS clouds for deadline constrained workloads. In: Proc. of the 3rd IEEE Int'l Conf. on Cloud Computing (CLOUD). Miami: IEEE, 2010. 228–235. [doi: 10.1109/CLOUD.2010.58]
- [31] Wang WJ, Chang YS, Lo WT, Lee YK. Adaptive scheduling for parallel tasks with QoS satisfaction for hybrid cloud environments. *Journal of Supercomputing*, 2013,66(2):783–811. [doi: 10.1007/s11227-013-0890-2]
- [32] Fontán J, Vázquez T, Gonzalez L, *et al.* OpenNEBula: The open source virtual machine manager for cluster computing. In: Proc. of the Open Source Grid and Cluster Software Conf. San Francisco, 2008.
- [33] Rushby J. Model Checking and Other Ways of Automating Formal Methods. *Software Quality Week*, 1995. 1–12.
- [34] Bruneliere H, Cabot J, Jouault F, Madiot F. MoDisco: A generic and extensible framework for model driven reverse engineering. In: Proc. of the 25th IEEE/ACM Int'l Conf. on Automated Software Engineering. Antwerp: ACM Press, 2010. 173–174. [doi: 10.1145/1858996.1859032]
- [35] Sicard S, Boyer F, Palma ND. Using components for architecture-based management: The self-repair case. In: Proc. of the 30th Int'l Conf. on Software Engineering. New York: ACM Press, 2008. 101–110. [doi: 10.1145/1368088.1368103]
- [36] Morin B, Barais O, Nain G, Jezequel J. Taming dynamically adaptive systems using models and aspects. In: Proc. of the 31st Int'l Conf. on Software Engineering. Washington: IEEE Computer Society Press, 2009. 122–132. [doi: 10.1109/ICSE.2009.5070514]
- [37] Li Y, Sun KW, Yang J, Liu TC, Zeng LZ. Model-Based system configuration approach for internetware. *Science China Information Sciences*, 2013,56(8):1–20. [doi: 10.1007/s11432-013-4917-3]
- [38] Chen XP, Huang G, Chauvel F, Sun YC, Mei H. Integrating MOF-compliant analysis results. *Int'l Journal of Software and Informatics*, 2010,4(4):383–400.
- [39] Li JG, Chen XP, Huang G, Mei H, Chauvel F. Selecting fault tolerant styles for third-party components with model checking support. In: Proc. of the 12th Int'l Symp. on Component-Based Software Engineering. East Stroudsburg: Berlin, Heidelberg: Springer-Verlag, 2009. 69–86. [doi: 10.1007/978-3-642-02414-6_5]

附中文参考文献:

- [7] 梅宏,申峻嵘. 软件体系结构研究进展. *软件学报*, 2006,17(6):1257–1275. <http://www.jos.org.cn/1000-9825/17/1257.htm> [doi: 10.1360/jos171257]



陈星(1985 -),男,福建永春人,博士,副教授,CCF 专业会员,主要研究领域为分布式系统,软件中间件,软件工程.



郭文忠(1979 -),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为计算智能及其在计算机网络中的应用研究.



兰兴土(1992 -),男,学士,主要研究领域为分布式系统,软件中间件,软件工程.



黄翌(1975 -),男,博士,教授,博士生导师,CCF 专业会员,主要研究领域为分布式系统,软件中间件,软件工程.



李隘鹏(1992 -),男,学士,主要研究领域为分布式系统,软件中间件,软件工程.