软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2012,23(7):1669−1687 [doi: 10.3724/SP.J.1001.2012.04105]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

W⁴H:一个面向软件部署的技术分析框架*

陈 伟[1,2,3+], 魏 峻[1,2], 黄 涛[1,2]

[1](中国科学院 软件研究所 软件工程技术研究开发中心,北京 100190)

[2](中国科学院 软件研究所 计算机科学国家重点实验室,北京 100190)

[3](中国科学院 研究生院,北京 100190)

W⁴H: An Analytical Framework for Software Deployment Technologies

CHEN Wei[1,2,3+], WEI Jun[1,2], HUANG Tao[1,2]

[1](Technology Center of Software Engineering, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[2](State Key Laboratory of Computer Science, Institute of Software, The Chinese Academy of Sciences, Beijing 100190, China)

[3](Graduate University, The Chinese Academy of Sciences, Beijing 100190, China)

+ Corresponding author: E-mail: wchen@otcaix.iscas.ac.cn, http://www.ios.ac.cn

Chen W, Wei J, Huang T. W⁴H: An analytical framework for software deployment technologies. *Journal of Software*, 2012,23(7):1669−1687 (in Chinese). http://www.jos.org.cn/1000-9825/4105.htm

Abstract: Deployment, as a post-productive activity, is an important phase of software lifecycle, in which software execution is supported through configuration, installation, activation, and other activities. In order to systematically know the state of art and technical progress of software deployment, this paper builds a multi-dimensional and fine-grained framework, W⁴H, to characterize the technologies and software systems. This framework consists of 5 aspects and 12 dimensions, covering the subject, object, scope, fashion style, and process of software deployment. Based on the W⁴H analytical framework, current representatives of the software deployment method and technique have been analyzed and summarized. The study results show that the analytical framework is capable of providing a more comprehensive analysis and significant guidance for the selection and development of software deployment methods and techniques.

Key words: software deployment; analytical framework; comprehensive analysis

摘 要: 部署是软件生命周期中的一个重要环节,是软件生产的后期活动,通过配置、安装和激活等活动来保障软件制品的后续运行.为了系统地了解软件部署的现状和最新进展,建立了一个多侧面、细粒度的分析框架——W⁴H,以对该领域的主要研究工作和系统工具进行概括分析.该框架从软件部署的概念和面对的问题空间出发,由 5 个侧面、12 个维度构成,覆盖了软件部署方法中主体、客体、适用范围、方式策略和过程支持能力等多个方面.基于 W⁴H 分析框架,对当前具有代表性的软件部署方法与技术进行分析和总结.案例研究结果表明,该分析框架能够对软件部署方法与技术进行较为全面的分析,对软件系统部署方法和技术的选择及开发具有重要的指导意义.

关键词: 软件部署;分析框架;综合分析

中图法分类号: TP311 **文献标识码**: A

* 基金项目: 国家自然科学基金(60903052, 61003029); 国家重点基础研究发展计划(973)(2009CB320704); “核高基”国家科技重大专项(2010ZX01045-001-010-4)

收稿时间: 2011-04-19; 修改时间: 2011-05-25; 定稿时间: 2011-08-09

部署作为软件生命周期中的重要阶段之一,承担软件系统运行之前的准备和初始化工作,部署对软件系统的实际意义主要体现在:(1) 部署技术影响着整个软件过程的运行效率和投入成本:IDC 的技术报告(IDC #28934)指出,软件系统部署的管理代价占到整个软件管理开销的绝大多数;(2) 软件配置过程极大地影响着软件部署结果的正确性:伯克利大学和斯坦福大学在联合承担的 Recovery-Oriented Computing(ROC)项目研究中发现,应用系统的配置是整个部署过程中的主要错误来源[1];Wang 等人[2]则发现,部署和维护中约 90%的错误与软件和部署环境的错误配置相关.

由上述内容不难看出,软件部署的直接目标是:1) 保障软件系统的正常运行和功能实现;2) 简化部署的操作过程,提高执行效率;3) 同时还必须满足软件用户在功能和非功能属性方面的个性化需求.软件部署技术研究领域中需要解决的核心问题可归纳为以下几个方面:

(1) 提高软件部署技术的通用性和灵活性:对应于如何提高软件部署技术的适用范围和扩展、定制能力,促使部署技术能够适用于更为广泛的软件类型和应用场景;

(2) 加强软件部署技术的可靠性和正确性:对应于如何降低软件部署过程中发生错误的几率,实现软件系统的正确配置,从而保障系统后续的运行;如何加强软件部署技术的优化能力,以满足用户的非功能需求;

(3) 提高软件部署技术的过程化和自动化程度:对应于如何提高软件部署技术的自动化程度,通过减少人工活动的参与以有效提高操作执行效率和降低部署成本,同时降低由于人为因素而引入错误的风险.

随着软件技术和互联网的发展,软件形态已由封闭、静态和单一向开放、动态和分布的方向发展,基于网络的大规模分布式软件系统(简称分布式系统)成为当前软件的主流形态并得到广泛应用,如服务计算[3]、web2.0[4]和正在逐渐兴起的云计算[5]都是分布式系统的典型应用模式.分布式系统基于多层分布式体系结构,通过分布式技术实现异构平台间的相互通信,以松散耦合为特征、以高度复用为目标的组件技术[6]和面向服务的体系结构(service oriented architecture,简称 SOA)[3]已成为当前分布式系统的实现基础,使软件系统具有更高的开放性和扩展能力.但是,软件系统在规模、复杂度和动态性方面发展的同时也为其部署提出了更高的要求,带来了更大的困难.部署在软件生命周期中具有重要意义,而且软件形态及其应用模式的不断发展还为软件部署带来新的困难,这些原因使软件部署技术得到了研究领域及工业界的持续关注.

本文第 1 节首先对软件部署的概念进行讨论,基于已有的软件部署定义澄清软件部署的内涵,然后提出软件部署的一般过程模型,最后对当前典型的软件部署模式进行归纳总结.第 2 节基于软件部署概念及其对应的问题空间建立面向软件部署技术的 W^4H 分析框架,覆盖软件部署的多个侧面和评价维度.第 3 节基于 W^4H 分析框架对现有的典型软件部署技术的相关工作进行比较,其中包括研究工作、产品工具和标准规范这 3 个方面的现状.第 4 节对软件部署领域值得进一步研究的问题进行分析和展望,并对本文工作加以总结.

1 软件部署概念、一般过程模型和典型模式

1.1 软件部署概念

在软件部署研究领域,当前已有的工作并未对软件部署的概念给出严格、统一的定义,主要通过描述性语言来说明“软件部署是什么,应该包含什么”,内容包括软件部署在软件生命周期中所处的位置、涉及的主要活动和执行的具体任务等.

Wolf 等人在文献[7]中对软件部署给出了较为直观和概括的定义,他们认为,“软件部署包括了一系列使得软件系统能够被直接使用的活动”.以此为基础,Wolf 等人将软件部署过程定义为包含了发布(release)、安装(install)、更新(update)、激活(activate)、钝化(deactivate)、调整(adapt)、卸载(uninstall)、退役(retire)在内的多个相互关联的活动集合.

Dearle 在文献[8]中将软件部署看作是一种“软件生产的后期活动(post-production activity)”,他更强调部署是在获得软件制品后以用户为中心(user centric)、以支持软件运行为目标的过程,重点是面向用户的定制和配置活动.Dearle 还认为,部署在整个软件生命周期中起到了承上启下的作用,将软件开发和软件执行两个阶段紧密地衔接起来.

对象管理组织(Object Management Group,简称 OMG)在发布的基于组件的分布式应用部署和配置规范(deployment and configuration of component-based distributed applications specification,简称 D&C)[9]中将软件部署描述为“由软件部署人员执行的、介于获取软件制品和软件执行之间的过程”,这一过程包括了安装(installation)、配置(configuration)、规划(planning)、实施(preparation)、启动(launch)几个阶段.

从以上几个定义可以看到,虽然形式不尽相同,但从描述内容来看,软件部署的概念内涵主要包括如下几个方面:1) 部署在软件生命周期中的位置:主要覆盖由软件制品开发完成并交付到软件系统成功运行这一时间阶段,并随着软件部署向软件运行时管理的延伸涵盖了运行时的部分时间段或时间点;2) 软件部署的参与人员:整个软件的部署过程由负责软件部署安装的软件部署和管理维护人员来操作执行,同时也可能涉及系统开发人员和领域专家等相关角色;3) 软件部署的主要活动包括:打包(package)、安装(install)、更新(update)、激活(activate)、钝化(deactivate)、卸载(uninstall)等.除了上述几种基本活动外,软件部署过程通过向软件运行时的管理维护延伸,还可能包括对运行时系统的升级(update)、再配置(re-configure)和再部署(re-deploy)等;4) 软件部署的主要目标:支持软件运行,满足用户个性化需求,使得软件系统能够被直接使用.

1.2 软件部署典型模式

软件部署过程极大地影响了部署技术和方法的可靠性与正确性,当前主要存在着两种典型的软件部署过程,即: 1) 基于“尝试-纠错”的 Ad-Hoc 方式,以及 2) 基于“规划-执行”的系统化、过程化的部署方式.两类部署方式的主要区别在于:前者依赖管理维护人员的经验积累和领域知识,完全以专业人员头脑中的知识为基础来指导部署操作的进行.正是由于系统化、规范化机制的缺失,导致了错误的不断出现;后者作为更加全面和系统的过程,明确划分出了几个典型阶段,不同阶段之间通过文档和记录作为输入、输出,具有较高的系统化和文档化能力.过程化的部署方法具有更高的可靠性,使之能够更好地满足大规模、复杂度高的软件部署需求.以“规划-执行”为典型阶段划分的部署过程,能够应用于更多的软件部署场景,尤其是对于复杂分布式系统.

从软件部署技术适用的软件类型来看,部署模式可以被分为面向单机软件的部署和面向分布式软件的部署两类,分布式软件的部署需要借助运行支撑平台及代理来更好地完成部署方案的决策和规划活动.本节以适用软件类型为基础,着重从软件部署技术的方案规划能力和对执行过程的支持能力两个角度来总结归纳过程化软件部署方式下的典型模式.如图 1 所示,根据规划和执行这两个阶段的不同实现方式,当前主要存在着如下 3 种典型的软件部署模式,图中深色部分表示各模式中软件部署对应的功能模块和组成部分.

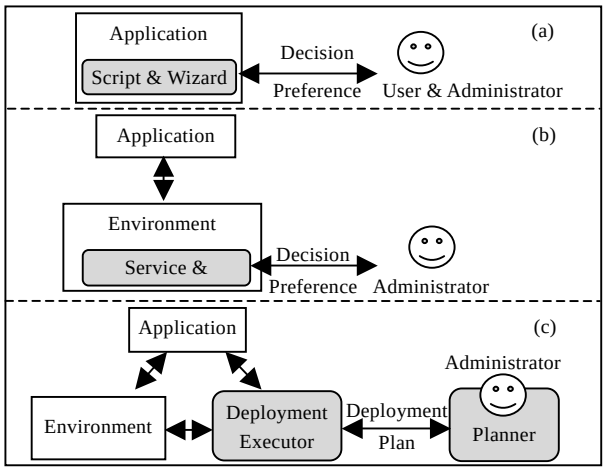


Fig.1 Patterns of software deployment

图 1 软件部署模式

1.2.1 面向单机软件的部署模式

如图1(a)所示,该部署模式将执行脚本与应用软件本身构造为自解压执行的安装包来支持部署过程中的特

定活动,主要包括安装、配置和卸载.鉴于软件本身结构单一,该模式下的规划活动较为简单,对于部署方案的决策主要取决于用户的偏好和选择,用户通过交互的方式进行简单的配置,如安装目录、所需软件模块等.在执行阶段,部署操作的执行功能主要通过脚本编程的方式来实现,以脚本语言编写的操作序列来支持诸如软件的安装、注册.该部署模式主要适用于运行在操作系统之上的、单机类型的软件,该模式下的部署方法对于软件信息和运行环境的表达能力十分有限,没有系统的规划方法能够支持少数的操作活动,该模式的典型代表包括基于 InstallShield[10],Microsoft Installer[11]等安装包制作工具生成的软件安装文档.

1.2.2 基于中间件平台的部署模式

基于中间件平台的部署模式是面向分布式软件的典型部署模式之一,随着中间件技术的出现和发展,作为应用系统运行环境的中间件平台和组件容器为软件系统提供了包括部署在内的软件生命周期多个阶段的支持.如图 1(b)所示,作为运行支撑环境的中间件平台使得部署功能与应用系统本身分离开来,而成为平台和容器中相对独立的公共基础服务,并以服务和接口的方式对外加以提供.这种分离机制大大增强了平台对于软件部署的支持能力,具体体现在包括了更多的执行活动和功能接口,如应用的安装、卸载、激活、钝化和版本管理等.在规划方面,中间件平台基于反射机制能够获取运行环境状态属性信息,为应用系统部署提供了更多的决策依据;但是中间件平台仍难以提供应用系统在部署配置过程中进行规划和决策的功能,对于部署方案的规划和构造仍需依赖维护管理人员的经验知识或其他外部系统来完成.基于中间件平台的软件部署模式主要适用于基于组件的分布式应用系统,该模式提高了对应用系统部署操作活动的支持能力,典型代表包括各类中间件平台,如基于 JavaEE 的应用服务器 WebSphere[12],JOnAS[13],OnceAS[14]等.

1.2.3 基于代理的部署模式

基于代理的软件部署模式通过对一类或多类(当前主要是基于组件的分布式应用系统)软件系统共性特征进行抽象,从而建立起更具通用性的软件部署机制,本文所提出的基于代理的软件部署模式主要适用于大规模的、复杂的分布式软件.如图 1(c)所示,在该模式下,部署代理独立于应用系统和运行环境,基于部署代理提供的执行引擎来实现部署过程中的操作活动,同时部署方案决策和规划器为应用系统部署方案的设计和生成提供支持.基于代理的软件部署模式在规划和执行两个阶段都能够为应用系统提供较好的支持,能够适用于多种形态和粒度的软件系统,在通用性、过程化和正确性保障方面都比前两类部署模式具有更强的能力,典型代表包括 SmartFrog[15],OW2 JASMINe[16]以及商业软件 IBM Tivoli[17]和 HP OpenView[18]等.

上述软件部署技术的基本模式虽然在应用场景和适用的软件类型方面有所不同,但在适用范围和部署过程的支持能力方面是逐渐增强的.需要指出的是,3 类典型模式并非正交,特别是对于后两类部署模式而言,在实际适用过程中往往是将几种模式结合使用.例如,对于当前广泛应用的分布式系统而言,将基于中间件平台的部署模式与基于部署代理的部署模式相结合,既能获取运行环境更为丰富和准确的属性信息,同时又能充分发挥部署代理的部署能力和规划能力,能够得到更为可靠的部署效果,并能更好地应对由系统规模和复杂度而带来的部署困难.

1.3 软件部署过程模型

以规划和执行作为软件部署过程的基本划分阶段,软件部署的一般过程可以归纳为以“运行环境和应用系统信息为依据进行部署方案的规划,以部署方案为依据指导各个部署活动的具体执行,整个过程还需要保障其正确性,并满足用户的需求”.基于上述归纳总结,本文提出了软件部署的一般过程模型,如图 2 所示.整个软件部署过程细化为描述(description)、规划(planning)、验证(validation)、执行(executing)和测试(testing)几个阶段,覆盖了文献[7,8]中所提出的软件部署生命周期的主要活动类型.

1) 部署信息描述:部署过程的描述阶段,包括应用系统、运行环境和用户需求以及部署策略等在内的描述信息是支持软件部署中决策和规划的基础,部署信息描述方法和机制决定了软件部署技术的适用范围,对信息的刻画能力影响了部署过程的可靠性和正确性,当前的部署信息描述机制主要是采用声明式的描述和以模型为基础的信息表达方式;

2) 部署方案设计/构造:部署过程的规划阶段,是一个规划求解的过程,以各类部署描述信息为输入,部署方

案刻画了在整个部署过程执行完毕时所期望达到的最终状态;本阶段主要采用约束求解和优化算法等方法并辅以人工操作来进行;

3) 部署约束验证和冲突消解:部署过程的验证阶段,依据施加在部署配置之上的约束信息来进行配置方案的检查和验证,从而提高软件部署方案的正确性;验证过程将部署配置中错误的发现、定位和修复提前,避免了执行阶段实际错误的发生,通过提高部署方案的正确性和可靠性来降低软件部署过程的代价和成本.本阶段主要基于逻辑语言进行部署约束的描述,并通过相关的工具来实现部署方案的约束验证,输出为进一步完善和修正的软件部署方案;

4) 部署流程的构造和执行:部署过程的执行阶段,以部署方案为输入并作为整个实施过程的参考依据和最终目标,本阶段主要采取目标驱动(goal-driven)的方法,结合具体的部署工具集合,通过部署流程构造的方式形成具体的执行流程,从而支持面向目标(部署方案)的过程实施;

5) 部署执行结果测试:在部署执行结束后,通过测试的方式来验证当前部署结果是否正确,是否能够满足预期的需求.

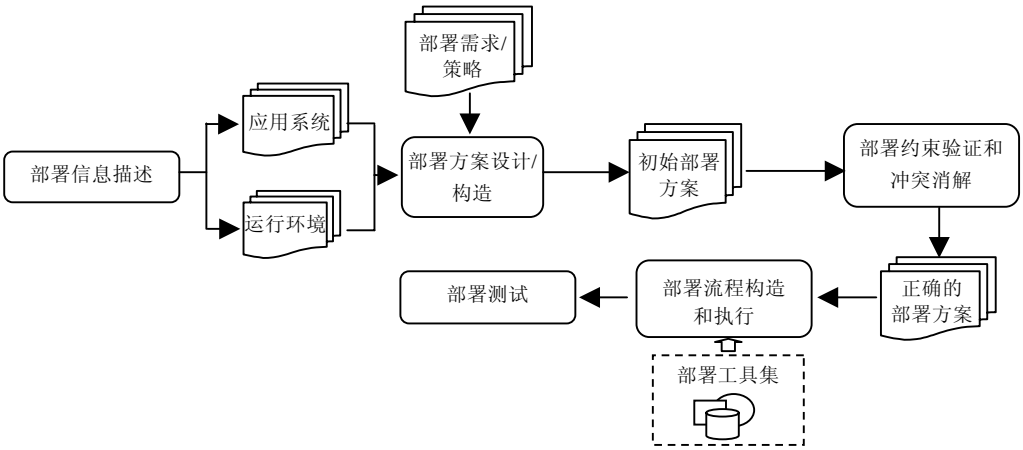


Fig.2 Generic process model of software deployment

图 2 软件部署一般过程模型

2 W4H 软件部署技术分析框架

软件类型、软件形态和应用场景的多样性决定了软件部署技术之间的差异.建立全面客观的软件部署技术综合评价体系对于更加深入、细致地分析、理解现有部署技术的能力、特点并明确未来软件部署技术的研究重点具有重要的指导意义.

从软件部署的概念、一般过程和典型模式来看,软件部署技术的差异性主要体现在适用的软件类型、参与人员、部署执行过程等诸多方面.通过对存在上述差异性的主要方面进行归纳,本文建立了面向软件部署技术的分析框架.该框架以部署技术主要面对的 3 方面问题为目标,涵盖如下 5 个分析侧面:

1) 软件部署技术的适用对象(object);

2) 软件部署技术的适用范围(scope);

3) 软件部署过程的参与人员(actor);

4) 软件部署技术采取的方式和策略(fashion);

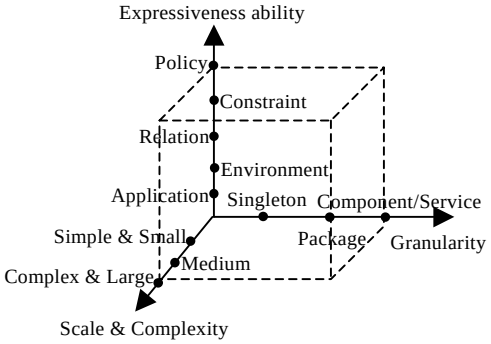
5) 软件部署技术对于部署周期的支持能力(lifecycle supportability).

分析框架的上述方面分别对应着部署技术所关注的 What(部署什么)、Where(部署的层次和范围)、Who(谁参与)、How(如何部署)、When(覆盖了部署周期的哪些活动)这 5 方面内容,因此本文将该框架定义为 W4H 软件部署技术分析框架,简称 W4H.通过上述 5 个分析侧面,W4H 同时也对应了研究领域中重点需要解决的关于通

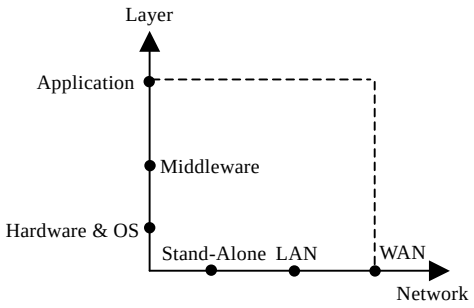
用性/灵活性、效率/自动化程度和可靠性/正确性的 3 方面问题.

2.1 适用对象

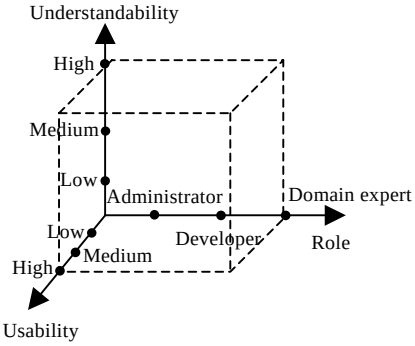
软件系统作为部署技术的作用目标是分析和评价部署技术的首要方面,如图 3(a)所示,该分析侧面进一步细化为部署对象粒度(granularity)、适用的软件规模(scale and complexity)以及部署信息的表达能力(expressiveness ability)这 3 个维度,目标是从部署技术的适用对象角度综合分析和评价部署方法的适用性.



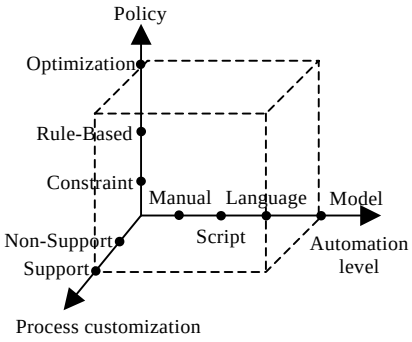
(a) W[4]H 软件部署技术分析框架之适用对象



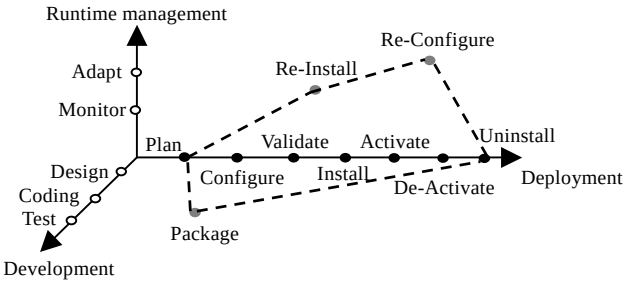
(b) W[4]H 软件部署技术分析框架之适用范围



(c) W[4]H 软件部署技术分析框架之人员角色



(d) W[4]H 软件部署技术分析框架之部署方式



(e) W[4]H 软件部署技术分析框架之部署周期支持能力

Fig.3 W[4]H framework for software deployment

图 3 W[4]H 软件部署技术分析框架

软件系统的构成形态包括单机软件(singleton)、基于包(package)和链接库的软件系统以及基于组件和服务的复杂分布式应用系统(component & service),因此,在部署对象粒度这一维度上的评价指标包括了上述 3 种形

态类型.

部署技术应对不同规模和复杂度的软件的能力各不相同,而应用系统的复杂度和规模取决于应用系统包括的组件或模块、物理节点和依赖关系的数量.以此为依据,部署技术适用的软件规模这一维度包括:1) 大规模、高复杂度(high);2) 中等规模及复杂度(medium);3) 小规模的简单系统(low).

部署技术对部署信息的刻画程度会影响部署技术的适用性和部署方案规划的可靠性,包括软件系统、运行环境和用户需求在内的部署信息,是部署方案规划的主要依据.因此,在部署技术的信息表达能力维度上,W4H主要从应用系统(application)、运行环境(environment)、应用系统和运行环境之间的关联关系(relation)、约束(constraint)和部署过程依据的策略(policy)共 5 个方面作为评价部署技术信息表达能力的指标,其中,约束包括了软件体系结构、用户非功能需求等方面的内容.

2.2 适用范围

适用范围主要从部署技术的适用场景和软件系统对应的运行环境角度进行分析,如图 3(b)所示.适用范围细化为运行环境所处的系统层次(layer)和网络环境(network)两个维度,目标是从软件的运行环境(where)角度综合评价部署技术的适用性.

部署层次主要从纵向维度来评价部署技术是否能够满足不同层次上软件系统的部署需求.根据软、硬件资源和功能的不同,该维度基于当前分布式软件系统的典型分层结构,将评价指标划分为:1) 硬件资源和操作系统(hardware & OS);2) 中间件平台(middleware)和 3)应用系统层次(application).

网络环境则主要从部署技术所能适用和应对的环境特点出发,根据应用系统所运行的网络环境不同分为:1) 单机(stand-alone);2) 局域网环境(LAN)和广域网环境(WAN).

2.3 人员角色

人员是部署过程的参与者和重要因素,该方面主要从参与人员(who)的角度来评价软件部署技术,如图 3(c)所示,包括了部署过程参与的角色(role)、部署技术的学习成本(understandability)和部署技术的使用代价(usability)这 3 个评价维度.

部署过程涉及开发人员、管理员以及领域专家等多类角色,部署技术对于部署过程的支持度越大,所涉及的角色类型也更加多样.部署技术的学习成本和使用成本主要是从人员代价的角度进行评价,在实际应用中,主要以部署技术的学习过程和使用过程需要投入的时间和操作步骤为依据进行量化分析.

2.4 部署方式

部署方式主要从部署技术采用的方式和方法(how)的角度进行分析,反映的是部署技术的行为和能力特点,如图 3(d)所示,包括了自动化程度(automation level)、部署技术依据的策略(policy)和执行流程的可定制能力(customizability)这 3 个维度.

过程自动化也是部署技术面对的核心问题之一,参考文献[19]的方法,本文在这一分析维度将自动化程度分为手动方式(manual)、基于脚本的部署方式(script-based)、基于语言的部署方式(language-based)和基于模型的部署方式(model-based).Talwar 等人在文献[19]中通过实验对上述部署方式的自动化程度进行了比较,比较的依据主要是:软件部署过程所需编写代码行数、所涉及的操作步骤数、对于配置变化的表达能力以及时间花费等定性和定量指标.实验结果表明,这几类部署方式的自动化程度按序依次升高,基于模型的部署方式通过使用模型来创建部署方案,从而将自动过程从执行阶段扩展至部署的设计规划阶段,具有最强的自动化能力.

部署策略维度主要从部署方案的规划和构造方面进行考虑,该维度主要分析和评价的目标是部署技术的可靠性和正确性.首先,部署方案构造时必须满足应用系统在资源和功能方面的基本约束(constraint);其次,在构造过程中,预定义的业务规则(rule-based)能够作为指导来引导整个方案的构造过程;最后,出于对非功能需求的考虑,部署技术还可以从部署方案的优化方面来加以完善,即是否考虑部署方案的优化(optimization).

部署技术的过程可定制能力决定了它的灵活性,进而影响了部署技术应对不同需求和场景的能力,这一维度是评价部署技术的灵活性和过程化能力的重要尺度.

2.5 部署周期支持能力

部署周期支持能力主要从部署技术对于部署过程的支持度进行分析,如图 3(e)所示.从软件生命周期的时间维度(when)来看,将其周期分为开发(development)、部署(deployment)和运行(runtime)这 3 个维度,每个维度轴上和坐标平面内的点均表示属于不同阶段的活动.就部署过程而言,包括了规划、配置、安装等多个活动.除了这些活动以外,还存在一些位于阶段重叠区域的活动,在图 3(e)中以位于坐标平面内的灰色点来表示,如打包(package)、再配置(re-configure)则分别属于部署-开发和部署-运行阶段相重叠的活动.这些相互关联的活动承担着部署阶段中对应的任务,因此,这一方面的评价标准主要依据部署技术能够支持的部署过程所包括的活动数量,能够支持的操作活动越多,那么该部署技术对于软件部署周期的支持能力也越强.

2.6 相关工作比较

Carzaniga 在 20 世纪 90 年代提出了一个面向软件部署技术特性的分析框架[20,21],是软件部署技术研究领域中关于分析框架的较早工作之一.该框架着重从软件部署研究的目标之一,即如何提高软件部署的通用性和灵活性方面入手,从 4 个方面的特征来进行框架的建立,如图 4 所示.过程覆盖性(process coverage)从部署技术能够支持的软件部署过程涉及的多类活动来衡量和评价部署技术的功能支持能力;过程可变化能力(process changeability)用以评价部署技术根据场景和需求的不同进行过程定制的能力,目标是评价其灵活性和可扩展能力;过程协同能力(inter-process coordination)则是考虑到分布式应用系统部署时不同活动和过程之间由于存在依赖和制约而必须考虑的协作和协同,即评价部署技术在操作执行时所具有的过程化能力;模型抽象能力(model abstraction)用以分析部署技术在描述部署相关的数据信息时的抽象表达能力,该评价侧面包括了软件运行时依托的环境(site)、软件系统(product)、部署过程依赖的策略(policy)这 3 方面的内容.该框架作为早期的对于软件部署技术的分析工作,从定性分析的角度着重评价部署技术的通用性和灵活性.

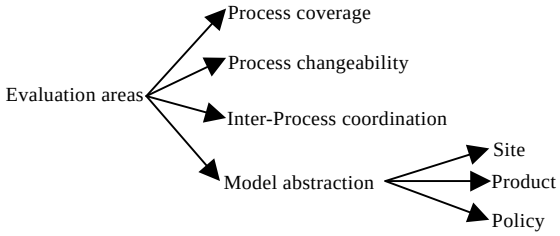


Fig.4 Areas of characterization

图 4 特征分析框架的主要分析方面

Talwar 等人在文献[19]中基于软件的管理质量(quality of manageability,简称 QoM)概念建立了基于 QoM 的软件部署技术比较框架,包括了定性和定量两类评价指标.在量化评价指标方面,主要通过:1) 实现部署过程需要编写的代码行数;2) 部署过程涉及到的操作数;3) 应用系统进行部署所花费的时间这 3 个维度来建立;而定性指标主要包括:1) 部署管理过程的自动化程度;2) 对于部署过程的正确性保障能力.该工作以部署过程的自动化程度为依据,将部署技术分为手工部署方式、基于脚本部署方式、基于语言的部署方式以及基于模型的部署方式,并在该框架下对上述 4 类部署方式进行比较,提出并验证了如图 5 所示的结论,即软件部署工具和方法的自动化程度越高,那么在该方法和工具的设计、开发和使用早期需要投入相对较高的代价,如对工具的开发和学习成本;但在后期使用中,部署过程所需投入的成本和使用代价将随着方法自动化程度的提高而大为降低,同时,应对复杂度高、规模大的系统的能力却大为提升.基于 QoM 的部署技术比较框架主要从人员成本的角度对 4 类部署技术进行分析,着重评价部署技术的过程化和自动化程度.但是,在其评价指标中,代码行数-执行时间-执行步骤几个指标之间存在着相互的关联关系,因此,在这些指标上并非完全正交的评价指标.

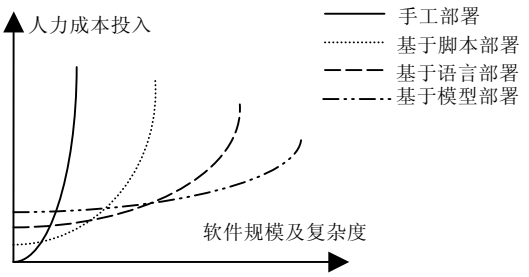


Fig.5 Comparison of deployment technologies based on QoM

图 5 基于 QoM 的软件部署技术比较

与上述工作相比,本文建立的 W4H 部署技术分析框架借鉴了上述两个工作的特点和研究成果,但同时又依据软件部署技术研究领域所面对的核心问题及其内涵,对分析框架的分析侧面和维度进行了重新划分,使之既能包含软件部署内涵的方方面面,又能够响应软件部署技术面对的核心问题(即:通用性和灵活性、可靠性和正确性、过程化和自动化).W4H 分析框架采取定性与定量分析评价相结合的方式,相比于已有工作,具有更强的针对性和更为全面的覆盖度.

3 软件部署技术相关工作概述与比较

本节以 W4H 作为软件部署技术分析工具,依据该框架对目前具有代表性的软件部署技术相关工作进行系统梳理和分析,包括了研究领域的相关成果、已有的工具系统和与软件部署相关的标准规范等工作,最后通过分析,总结目前该领域的主要研究思路和特点.

3.1 软件部署相关研究工作

3.1.1 IBM 研究机构的系列工作

包括软件部署在内的软件管理维护是IBM的研究领域之一,IBM研究机构的工作遵循了“规划-验证-执行”的部署一般过程模型,其研究成果涉及部署信息建模、方案规划、正确性验证和操作执行的几乎所有阶段.IBM研究机构采用了基于模型的软件部署方式来提高部署过程的自动化程度,采用“基于代理”的软件部署模式,目标在于提高软件部署平台的通用性和部署能力.

- 部署信息抽象和建模

IBM Watson 研究中心的 Tamar 等人在文献[22]中提出了一组与应用系统部署相关的核心配置元模型(core configuration meta-model,简称 CCM),用来刻画基于组件和服务的分布式应用系统部署的通用语法、结构和语义信息.CCM 以组件为应用系统的信息抽象和描述粒度,描述应用系统在软件体系结构层次的组成结构、互联关系以及与运行环境的关联.CCM 重点建模的实体与信息包括:部署单元《Unit》、能力《Capability》、需求《Requirement》、制品《Artifact》和关联关系《Relation》等.以 CCM 为基础,通过对元模型的扩展来描述领域相关的元素和信息,如对《Unit》的扩展可以描述特定的运行环境(数据库、应用服务器、Web 容器等)和应用系统的功能组件(Web 服务、EJB 组件等).

IBM 中国实验室的 Li 等人则提出了软件资源配置模型(software resource configuration model,简称 SRCM)[23],SRCM 主要从细粒度方面对应用软件和运行环境的参数资源配置进行抽象建模,着重对分布式应用系统部署中软件配置参数与运行环境所提供的资源之间的关系进行刻画.该模型以软件组件(software module)为中心,组件具有相应的资源(resource specification),资源的供给(provision)通过正确的配置参数(resource entry)来实现,由“供给-消费”关系(provide-consume relationship)所关联的其他组件来使用,组件部署的配置参数(configuration entry)组成了对应的参数配置模板(configuration template),用来约束组件的部署和运行.

- 部署方案规划和构造

Arnold 和 Eilam 等人以 CCM 作为系统部署方案建模的基础,通过对应用系统部署方案共性特征的抽象提

炼出面向 SOA 应用的部署模板(deployment pattern,简称 DP)[24].DP 是对同一类应用系统部署最佳实践和经验的总结,定义了部署方案的整体结构以及在组件、资源和运行环境方面的类型,体现了应用系统在特定非功能属性(如高可用性、可扩展性、安全性)方面的特点.文献[25]根据经验总结了 12 个部署方案的最佳实践,并通过抽象和建模形成对应的部署模板.Arnold 和 Eilam 等人在文献[26]中基于给定的部署模板和当前的运行环境以及应用系统实现部署模板的实例化,在满足功能性需求的同时也保证了当前应用系统具有模板本身就具备的非功能特性.该方法将模板的实例化过程抽象为子图同构问题[27],由抽象元素到实例元素的关联与实例化对应于子图同构问题中的同构映射(isomorphism mapping).部署模板抽象为带有标记的图(labeled graph),对于运行环境的配置操作被视为图的修改编辑.方法中采用图匹配(graph matching)算法来实现模板到实际运行环境的实例化过程.

- 部署约束验证和异常消解

Luo 等人提出了面向分布式应用的声明式部署约束框架[28],以 CCM 描述的部署方案为约束验证对象,通过建立约束模板、生成基于一阶逻辑语言的约束描述和约束验证几个关键步骤,来完成对部署方案模型的形式化约束描述的生成和验证.该框架定义了导航模板(navigation pattern)和逻辑模板(logic pattern)作为生成约束的基础,逻辑模板作为约束的主体框架定义了多种计算和约束逻辑,导航模板则描述了部署方案模型中模型元素之间的关系以及关系两端的模型元素.该框架通过交互的方式根据约束语义生成对象约束语言(object constraint language,简称 OCL)[29]表示的形式化描述,并基于 OCL 解析和验证工具执行验证.

Li 还在文献[23]中基于 SRCM 提出了软件资源和参数配置层面的约束验证方法.该方法以 SRCM 中的软件配置参数(configuration entry)和资源配置参数(resource specification)为约束对象,提出了资源级约束(resource specification level constraint,简称 RSLC)和配置参数级约束(configuration entry level constraint,简称 CELC)的两级划分,RSLC 用以限制所有依赖该资源的软件模块中对应参数的设置,CELC 用以限制软件模块自身的参数配置.该方法定义了一组配置规则和约束验证算法,并基于 SRCM 表示的参数配置信息和关联关系来进行约束验证,以检验是否存在违背约束的参数配置.

Kannan 等人提出了对参数配置错误引起的异常情况进行消解的方法[30],该方法基于参数配置图(configuration map,简称 CM)来表示构成应用系统的应用组件和运行环境之间在参数上的依赖关系及配置顺序.CM 在形式上表现为依赖图,通过能够到达异常节点的配置路径来表示该路径上的参数配置会引起的异常情况,从而将应用系统的参数配置与可能引起的异常类型关联起来,减少了系统部署过程出现异常时进行错误定位的代价.CM 还在异常节点上扩展出消解路径(resolution path,简称 RP),RP 定义了对于此类异常情况进行消解的方法,使用包括参数重配在内的相应方法对异常进行消解.

- 部署执行流程构造

Maghraoui 等人提出了一种目标驱动(goal-driven)的软件部署执行流程生成方法[31],借鉴了基于人工智能规划的组合服务自动构造方法[32]:1) 使用 CCM 表示的部署方案作为目标状态;2) 基于人工智能规划的方法,将操作活动映射为规划器的执行动作,操作执行前的输入和前提条件转化为前提(pre-condition),操作执行后的输出和对资源、配置所产生的影响作为效果(effect);3) 以目标状态、当前状态和执行动作集合作为输入,采用偏序规划算法[33]进行分析规划,建立各个执行操作的调用顺序,能够实现由当前运行环境的初始状态到目标部署方案的转换.使用智能规划生成部署执行工作流程的方法仅需要对运行环境、部署方案和操作模型进行声明式表达,就能够快速地生成整个执行流程.

3.1.2 基于约束的部署和管理框架

圣安德鲁斯大学(St. Andrews)的 Dearle 教授在其工作[34,35]中提出了基于约束的分布式应用系统部署和管理框架,主要应用于软件的部署和后续的演化管理.基于约束的部署和管理框架同样采用了“基于代理”的部署模式,通过自定义的声明式约束描述语言 Deladas(declarative languagefor describing autonomic systems)来描述应用组件、物理环境和软件体系结构层次的约束信息.该框架通过约束求解的方法找到满足约束需求的部署方案,并生成基于 XML 的部署描述文档(deployment description document,简称 DDD)刻画最终应用组件与部署环

境的关联关系.框架中的部署和管理引擎(autonomic deployment and management engine,简称 ADME)通过进一步分析 DDD 来抽取并生成基于脚本描述的任务列表,如初始化、安装和绑定等,由相应的部署执行引擎将任务列表对应到安装、运行、连接等具体操作上,从而驱动整个过程的执行.

3.1.3 基于体系结构的 J2EE 应用部署技术

北京大学的 Lan 等人在文献[36]中提出了基于软件体系结构模型的部署方法,主要面向基于 J2EE 的分布式应用系统.该方法将软件体系结构作为部署信息的描述模型,基于软件体系结构描述语言(architecture description language,简称 ADL)[37]对 J2EE 应用的结构进行刻画,主要包括构成 J2EE 应用系统的组件集合(主要包括 EJB,Web Module 和 Data source 等)及其互联关系和接口、配置信息以及应用组件相关的语法和语义信息.该方法采取的是典型的"基于代理"与"基于中间件平台"相结合的部署模式,CADTool 作为部署方案的规划和设计代理来支持应用系统体系结构模型的建立、组件的打包、组装和部署信息的获取;中间件系统 PKUAS[38]作为 J2EE 应用系统的主要运行支撑环境负责提供应用系统组件的加载、配置、卸载以及启动、停止等服务,同时能够通过反射机制获取运行环境中当前部分环境资源的动态数据信息,包括运行环境的当前状态、所能够提供的资源(主要指 CPU 使用率和内存占用情况).

Lan 还提出了分布式应用系统在部署过程中应遵循的一些基本原则(principle),并根据其指导目标的不同分为不同的优先等级,包括:1) 确保系统正确运行的指导原则;2) 从应用系统运行维护的投入成本方面来考虑构造部署方案;3) 从应用系统的服务质量供给方面来考虑部署方案的优化.3 个级别的部署指导原则随着目标和需求类型的不同,其优先级呈逐级递减.

3.1.4 其他研究工作

在软件部署领域中还存在其他的研究成果,分别对该领域中的某些问题点以及特定场景下的部署问题进行研究.

南加州大学 Medvidovic 的研究团队致力于软件体系结构领域的相关研究,其中包括基于体系结构的大规模分布式系统的部署技术研究.Malek 和 Medvidovic 等人提出了以用户为中心的、以提高系统服务质量为目标的分布式系统部署框架[39,40].基于"分布式系统组件在不同硬件节点上的部署极大地影响了系统能够提供的服务质量(quality of service,简称 QoS)"这一判断,该工作提出部署方案的构造和优化方法,以满足不同用户群对多维 QoS 的不同需求.Malek 在文献[39,40]中首先建立了基于软件体系结构的部署模型,包括运行环境(host)、组件(component)、网络连接(network)、交互(interaction)、服务(service)、QoS、用户(user)和参数约束(parameter constraints)等,并对多种 QoS 属性(响应时间、可用性、安全性、能耗及内存消耗等)建立了计算函数;最后将寻找最佳部署方案映射为多目标优化问题,并分别通过线性规划、非线性规划、贪吃算法和遗传算法这 4 种典型的优化算法进行问题求解,并根据软件部署领域的特征,有针对性地提出了相应的启发规则来限制算法的复杂度.与上述研究成果类似,马萨诸塞大学的 Wada 等人在文献[41]中提出了云计算环境下的面向组合服务的、基于多目标遗传算法的服务部署优化策略.该工作采用多目标遗传算法进行服务水平协议(service level agreement,简称 SLA)中多维 QoS 属性的权衡(trade-off),找到 SLA 总体最优的"帕累托"[42]最优部署配置,帮助部署人员理解并选出满足需求和偏好的部署方案.

云计算环境下,PaaS(platform-as-a-service,平台即服务)为上层应用提供了一个具有伸缩能力(scalability)和弹性(elasticity)的运行支撑环境.Google App Engine(GAE)[43]是 PaaS 平台的典型代表,采用进程隔离的方式为 Web 应用提供运行支撑,能够支持 Java 语言编写的 Web 应用系统的部署和运行.GAE 上应用系统的部署过程与普通分布式环境下 Web 应用在中间件平台上的部署过程类似,通过 Web 应用部署描述文件(web.xml)进行部署信息的识别和部署操作,但 GAE 对 JAVA EE 的标准规范进行了限制和部分支持,应用系统必须满足 GAE 所提出的 API 和编程规范才能够成功部署和运行.IaaS(infrastructure-as-a-service,基础设施即服务)层部署和管理的对象通常为虚拟机(virtual machine,简称 VM)实例,VM 包括了应用系统、运行支撑环境甚至操作系统.由于 VM 是部署、管理和资源消耗的基本单元,IaaS 通过 VM 的添加、迁移和服务器合并(server consolidation)达到提高应用系统的 QoS 属性、降低资源消耗和满足用户 SLA 等目标.Jayasinghe 提出了一种考虑结构约束的虚拟机

部署策略(structural constraint aware virtual machine placement,简称 SCAVP)[44],该方法考虑资源需求约束(demand)、通信代价(communication)和可用性约束,将 SCAVP 映射为优化问题并采用分而治之的策略加以解决.SandPiper[45]是一个基于 Xen VM[46]的 PaaS 的管理和部署系统,SandPiper 以度量指标(sand_volume)作为 VM 迁移的决策依据,sand_volume 主要衡量当前物理节点在 CPU、网络带宽和内存方面的可用资源.当需要进行 VM 迁移时,将物理节点按照 sand_volume 进行排序,从中选取能够满足 VM 资源需求且可用资源量最小的物理节点进行迁移.Singh 基于向量点积(vector dot)建立物理节点与 VM 间关联的方法[47],该方法将物理节点中已用资源向量值(resource utilization vector,简称 RUV)与 VM 资源需求向量(resource requirement vector,简称 RRV)的点积作为选择标准,点积值小的物理节点被选中.该方法的含义是,尽量选取 RUV 与 VM 的 RRV 能够形成互补关系的物理节点来部署 VM,以保证各类资源使用的相对均衡.

软件动态更新实现系统持续运行状态下在代码和体系结构等方面的变化,从而适应运行环境、用户需求等方面的变化.当前,软件动态更新的研究工作主要面向复杂分布式软件系统,采取基于软件体系结构的方式实现其在线的更新.软件动态更新的难点主要在于如何保障软件系统的状态一致性以及如何减小动态更新带来的额外开销.在状态一致性保障方面,Kramer 和 Magee 提出了组件的静止状态(quiescence)[48],并证明分布式系统的组件只有在达到静止状态时才能够进行更新并保证系统状态的一致性;而 Vandewoude 则在文献[49]中通过放松静止状态的条件提出了系统的稳定状态(tranquillity),并认为分布式系统的组件在达到稳定状态时即可进行更新,并保证其前后状态的一致.在体系结构描述语言方面,许多研究工作通过对只能描述静态属性的体系结构描述语言进行扩展,加入动态特性和约束描述而形成能够描述体系结构变化的动态体系结构语言,如 Drawin[50],dynamic-ACME[51],π-ADL[52]等.在运行支撑环境方面,OSGi[53]及 Fractal 平台[54]能够为基于其开发、运行的应用程序提供动态更新的支持.软件动态更新,尤其是基于体系结构的分布式软件动态更新技术在软件演化领域得到更多的关注和研究,并存在大量的研究工作.Godfrey 等人在文献[55]中给出了软件演化领域相关的综述和分析,本文不再赘述.

3.2 部署工具与系统

3.2.1 SmartFrog

SmartFrog(smart framework for object groups)源于 HP 的研究和开发成果,是一个开源的部署、配置和管理平台,主要面向基于组件的、Java 语言实现的分布式应用系统.作为通用软件部署代理的 SmartFrog 采用基于语言的部署方式,其主要组成元素包括:1) 一套声明式的部署描述语言和解析机制作为软件系统的描述和建模基础,并通过具有较高表达能力的标注信息来描述系统配置;2) 运行时引擎作为部署操作的代理来完成对分布式应用的安装和管理操作.SmartFrog 还从实现角度定义了组件模型,包括组件代码、位置、配置信息和基于有限状态机定义的组件生命周期,以及与应用系统部署方案相应的操作流程和生命周期管理.运行时,引擎作为部署操作的执行代理,对应于组件的生命周期提供了部署、启动和终止等操作活动,基于 RMI(remote method invocation)实现分布式应用的运行时管理和通信.除此之外,SmartFrog 还提供了一组辅助工具(如:开发工具)来简化部署方案的创建、验证和监视等.

3.2.2 JASMINe

JASMINe 是国际开源社区 OW2 提供的一个面向分布式应用系统部署、监控和管理的工具集合,其管理对象着重面向中间件集群和运行支撑平台.JASMINe 提供了基于模型的部署方案设计环境,主要针对以 JonAS, Tomcat 和 Apache HttpServer 的中间件进行应用系统运行平台部署方案的设计和配置.同时,JASMINe 还提供了面向分布式应用部署的执行代理来实现应用系统的上载、安装、验证和版本管理操作.除此之外,JASMINe 还提供了运行时的监控和管理工具,从而形成由静态的初始部署到运行时维护管理的较为全面的运行维护套件.

除上述开源系统外,IBM Tivoli 和 HP OpenView 作为商用的专业管理维护软件提供了一整套的部署和管理支持,能够支持包括硬件系统、网络环境在内的各个层次的操作和管理,能够支持部署中规划设计、验证和操作执行各个过程阶段,并通过部署策略和执行流程来提高可靠性与自动化程度.

3.3 部署相关标准规范

3.3.1 Common Information Model

Common Information Model(CIM)[56]是由国际标准化组织 DMTF(the Distributed Management Task Force)发布的面向网络环境下应用系统管理的公共信息模型.CIM 以一组核心模型(core model)为基础,通过扩展,定义出各类平台无关的领域模型(platform independent model,简称 PIM),用以建模数据库、网络环境、硬件设备和应用系统类型的实体.在描述和建模的实体类型方面,CIM 在应用系统(application model)和运行环境(network, device& system model)以及策略(policy model)方面提供了建模支持,其中,以软件元素(software element)作为部署和管理的最小单元,将软件的部署阶段划分为可部署(deployable)、可安装(installable)、可执行(executable)和运行(running)几类;在部署过程涉及的操作方面(action),CIM 定义了安装、卸载、启动和停止这几类操作;CIM 还定义了部署时对于资源需求和系统版本兼容方面的检查,以确保对需求的满足.

3.3.2 OMG D&C

D&C 定义了一套 PIM,主要从数据信息和操作管理两个维度定义了数据信息模型(data/information model)和管理操作模型(management/operation model).在数据信息模型方面,包括组件模型(component model)、部署目标模型(target model)和部署执行模型(execution model),其中,应用系统由组件构成,组件是部署的基本单元;在管理操作模型方面,通过定义诸如目标管理器(target manager)、执行管理器(execution manager)等面向不同目标的管理器模型来支持部署时的管理和操作功能;在部署过程周期方面,D&C 为部署单元定义了包括安装、配置、规划、启动等在内的多个阶段,同时还根据不同软件周期的不同阶段定义了包括在开发和部署阶段中的多类参与角色;在部署过程涉及的操作方面,D&C 定义了启动和停止操作.

3.4 分析比较

3.4.1 适用对象分析

在粒度方面,现有研究工作、系统工具和标准规范主要面向基于组件(component)的分布式应用,同时,IBM 的工作以及 SmartFrog 和 JASMINe 则明确提出了对 Web 服务(service)类型组件的支持.例如,Arnold 等人在文献[10]中所提出的 CCM 包括了部署单元(unit)元素,通过对该元素进行领域扩展来支持对 Web 服务、EJB 组件以及应用服务器、Web 容器和数据库等领域相关组件的语义描述.

在信息表达能力方面,已有工作都覆盖了应用系统(application)、运行支撑环境(environment)、关联关系(relation)这 3 个主要方面,各工作的差别在于对部署策略(policy)和部署约束(constraint)的支持.IBM 的相关工作涵盖了应用、环境、关系和约束,信息表达能力较为全面.基于体系结构的 J2EE 部署技术提出了多条部署过程需要遵循的基本原则,在部署策略支持能力方面较为突出.D&C 和 CIM 作为标准规范对应用、环境、关系、约束都有较为全面的描述,尤其是 CIM 所提供的策略模型(policy model)提高了其部署策略的表达能力.

在适用对象的系统规模方面,现有工作大多采用基于模型的部署方法,Talwar 等人的工作[19]通过实验证明了基于模型的部署方法能够支持复杂的、大规模的(high)应用部署.基于约束的部署和管理框架以及 SmartFrog 采用声明式的描述语言进行部署需求和配置的刻画,上述两个研究工作和工具更加适用于规模和复杂度稍小(medium)的应用系统.

3.4.2 适用范围分析

在网络环境方面,当前已有工作主要通过集中式的部署控制来实现局域网(LAN)和广域网(WAN)环境下的分布式应用部署.在广域网环境下,当前部署技术的可行前提是广域网环境下的可控物理节点和运行环境.当前,广域网环境下实现了对任意节点和运行环境的部署,难点在于难以建立起集中式的部署控制.同时,对于广域网节点的访问控制和安全认证也是一个难题.

在系统层次方面,现有工作都能够实现对应用层(application layer)系统部署的支持.而 IBM 的相关工作以及 SmartFrog 和 JASMINe 从应用层扩展到中间件层(middleware layer),能够支持多种组件容器和服务器的部署配置,具有多层次系统部署支持的能力.

3.4.3 人员角色分析

在参与角色方面,除了系统部署和管理人员(administrator)之外,IBM 的研究工作、基于体系结构的 J2EE 部署技术以及 D&C 都需要获取组件部署包及软件制品的详细信息,需要开发人员(developer)的参与.进一步地,Arnold 等人在文献[24]中提出了基于模板的部署方案规划方法,需要领域专家(expert)对典型应用系统的通用部署模板进行总结和归纳.

在学习成本方面,SmartFrog 和 Dearle 提出的基于约束的部署和管理框架需要学习声明式语言的语法和编程模式来支持部署方案、执行过程和约束的定义,其他部署技术则通过可视化的建模环境来降低部署人员的学习成本,使之仅仅需要了解部署的语义信息而省去了学习语言的代价.两者相比较,以 IBM 的研究工作为代表的部署方法具有较低的学习成本.

在可用性和易用性方面,软件部署技术的使用成本从操作步骤、代码编写行数以及时间开销这几个方面进行定量分析.在本文所概述的部署技术中,基于模型的部署方法由于其直观性和较高的重用性能够减少操作步骤和代码编写量,与 SmartFrog 和基于约束的部署和管理框架相比较,在面对相同的应用系统时,使用更为简单,代价较小.

3.4.4 部署方式分析

在自动化程度方面,本文参考了 Talwar[19]的工作,以人工参与的程度为依据分为手动部署(manual)、基于脚本的部署(script)、基于语言的部署(language)和基于模型的部署(model).SmartFrog 和基于约束的部署及管理框架以声明式的描述语言来定义部署信息和部署过程;而本文概述的其他工作则通过提供可视化的建模环境来支持上述工作,属于基于模型的部署方式.

在部署策略方面,已有工作主要考虑了对于部署需求和部署约束(constraint)的满足,并分别通过模型层面的约束建模、部署方案的约束验证来提高部署方案的正确性.基于体系结构的 J2EE 部署技术则提出了软件部署中的一些通用规则(rule-based).

在执行流程可定制能力方面,IBM 的 Maghraoui 等人在文献[31]中明确提出了目标驱动的、基于服务组合的执行流程构造技术,能够实现执行流程的生成和变化,具有流程定制能力(process customization).

3.4.5 部署周期支持能力分析

软件部署过程包括了规划、配置、安装等活动,同时以打包、再配置、再安装作为与软件开发和软件运行两个阶段相交叉的活动.本文概述的部署方法都能够支持配置(configuration)、安装(installation)、激活(activation)、钝化(de-activation)等基本操作,尤其是 SmartFrog 和 JASMINe 不仅能够支持部署约束的验证(validation),还能够通过多版本共存等机制来支持运行时的系统再配置(re-configure).

3.4.6 小 结

基于上述分析比较内容,图 6 对本文中所概述的部署技术进行了总结,表中的缩写字母对应表下方的注释:“-”表示不涉及相关内容;“*-”表示部分支持,其中,“*”为表中的缩写字母.从上述分析内容和图 6 可以看出,目前主流软件部署技术工作和相关成果呈现出以下特点:

(1) 从部署技术的应用对象来看,当前的应用对象主要集中在基于组件和服务的大规模分布式应用系统上;

(2) 在部署模式方面,基于代理的部署模式占据主流,将软件部署功能从运行环境和应用系统中分离出来,并辅以中间件平台自身的管理功能,使其更加专业化、通用化,并且具备更强的能力和更为丰富的功能;

(3) 在部署技术的自动化程度方面,基于模型的软件部署方式将自动化的能力由执行阶段延伸至设计阶段,从而成为面向复杂分布式应用系统的主流方式;

(4) 在部署技术的信息抽象和表达能力方面,各个工作基本都覆盖了软件系统和运行环境两个基本方面,部分工作还支持对于应用组件、运行平台之间在功能和资源方面的依赖关系和部署策略、操作的建模;

(5) 在部署对象的支持层次方面,分布式应用的典型分层结构决定了大多数工作都能够支持包括中间件层次在内的应用系统部署.除此之外,对于部分商业软件,还支持包括硬件资源和网络设备在内的更底层次的安装

和配置;

(6) 在部署操作过程的灵活性和过程性方面,当前的部署技术逐渐形成以部署方案为目标的 goal-driven 的操作流程构造以及流程驱动的操作过程执行,从而进一步提高了部署技术的灵活性和定制能力;

(7) 在部署过程生命周期的支持方面,配置(configuration)、安装(installation)、激活(activation)、钝化(de-activation)和卸载(un-install)作为部署阶段的基本操作能够被多数工作支持,对于和软件运行时管理、演化相交叠的再配置、再安装、升级等操作也得到关注和研究,以提高部署技术在软件动态特性方面的支持力度.

		Object (What)			Scope (Where)		Fashion (How)		
	Name	Granularity	Expressiveness ability	Scale & Complexity	network	Layer	Automation Level	Process Customization	Policy
Research Works	IBM Research[22,23,24,26,28,30,31]	C/S	A/E/R/C	Complex & Large	L/W	M/A	M	Y	C
	Constraints based framework[34-38]	C	A/E/C	Medium	L/W	A	L	N	C-
	Arch. based deployment[39]	C	A/E-/P-	Complex & Large	L/W	A	M	N	C/R-
Systems and Tools	SmartFrog[15]	C/S	A/E/R	Medium	L/W	M/A	L	N	-
	OW2 JASMINE[16]	C/S	A-/E/R	Complex & Large	L/W	M/A-	M	N	-
Standards & specifications	CIM[54]	C-	A/E/R/C/P	Complex & Large	L/W	H&O/M/A	-	-	C-
	D&C[9]	C	A/E/R/C/	Complex & Large	L/W	H&O /M/A	-	-	C-

Granularity: **Sin**-singleton **P**-Package, **C**-Component, **S**-Service network: **S**-Stand-alone, **L**-LAN, **W**-WAN
Policy: **C**-Constraint, **R**-Rule based, **O**-Optimization Layer: **H&O**-Hardware **&** OS, **M**-Middleware, **A**-Application
Expressiveness ability: **A**-Application, **E**-Environment, **R**-Relation, **C**-Constraint, **P**-Policy Scale & Complexity: Simple & Small, Medium, Complex & Large
Automation Level: **S**-Script, **L**-Language, **M**-Model Process Customization: **N**-Non-support, **Y**-Support

		Actor (Who)			Lifecycle Supportability (When)
	Name	Role	Understandability	Usability	
Research Works	IBM Research[22,23,24,26,28,30,31]	A/D/E	H	H	PL/C/V/I/A/DA
	Constraints based framework[34-38]	A	M	M	PL/C/I/A/DA
	Arch. based deployment[39]	A/D	M	H	C/I/V/A/DA
Systems and Tools	SmartFrog[15]	A	M	M	C/V/I/A/DA/RC/U
	OW2 JASMINE[16]	A	H	H	C/V/I/A/DA/U/RC
Standards & specifications	CIM[54]	-	-	-	I/U/A/DA
	D&C[9]	A/D	-	-	I/C/P/A/DA

Role: **A**-Administrator, **D**-Developer, **E**-Domain Expert Understand ability: **H**-High, **M**-Medium, **L**-Low Usability: **H**-high, **M**-Medium, **L**-Low
Lifecycle: **PL**-Plan **P**-Package, **C**-Configure, **V**-Validate, **I**-Install, **A**-Activate, **DA**-Deactivate, **U**-Uninstall, **RC**-Re-configure, **RI**-Re-install

Fig.6 Comparison of deployment technologies based on W⁴H

图 6 基于 W⁴H 的软件部署技术比较

4 研究趋势

软件形态、规模及其应用模式的发展,为软件部署及其相关领域的研究带来了新的困难和挑战.基于 W⁴H,对当前软件部署技术研究热点和主要工作进行了分析,本文认为,软件部署技术的研究趋势和研究内容包括以下几个方面:

(1) 软件部署正确性和可靠性保障技术研究

当前的研究热点集中在软件部署技术的通用性、灵活性和自动化、过程化两个方面,而关于如何提高软件部署技术的正确性与可靠性的研究相对薄弱.已有工作主要集中在基于一阶逻辑语言进行部署方案的静态验证方面,如文献[23,28].当部署方案中存在相互冲突的配置或者用户需求本身存在冲突时,进行需求冲突的权

衡、消解以及配置修正的策略和机制还需要进一步加以研究;而从动态方面(即部署执行过程)来看,如何保证部署操作执行过程的可靠性和容错性以及降低操作对于已有环境和系统带来的影响,都是需要进一步研究的问题.

(2) 软件动态更新和再配置技术研究

从软件部署的概念来看,应用系统运行时会随着运行环境、应用需求的变化而产生再配置和应用升级的需求,即应用系统在运行过程中并非一成不变,而是存在着动态演化[57],这使得软件部署与软件演化、自治计算等研究领域产生交叉和融合.当更加强调应用系统运行时的动态演化需求时,部署技术在部署方案建模时需要更多地考虑对动态变化信息的描述以及与之相应的再配置、再部署策略,需要更多地借鉴和引入软件在线演化和管理领域的研究成果,从模型转换正确性、应用系统升级和再配置的时机等方面给予更多的考虑,以提高软件再部署、再配置的安全性和可靠性.Leger 等人在文献[58]中基于事务的概念进行了提高软件系统再配置可靠性的探索.

(3) 新的应用模式下软件部署技术研究

云计算作为一种新兴的服务模式和应用模式,采取集约化的方式按需为相应的用户群体提供资源和服务.从应用服务的提供方来看,软件开发方作为 SaaS(software-as-a-service,软件即服务)的提供者同时又是 IaaS 和 PaaS 的需求方,基于上述场景,应用系统的开发方必须从平台资源使用成本、应用系统不同用户群体的服务需求以及平台提供方声明的 SLA 等多个方面进行综合考虑和权衡,来构造部署成本和服务质量综合最优的部署方案,这使得部署方案的决策和优化更为复杂,因此也决定了难点在于如何描述用户对于应用系统的非功能性需求、如何建立优化目标函数和模型以及相应高效的优化算法.如文献[59]以数据传输可靠性(data transmission reliability,简称 DTR)和传输开销(communication overhead,简称 CO)非功能属性建立多目标优化问题,并基于蚁群优化算法进行优化部署方案求解;从平台和基础设施服务提供方来看,作为运行环境和计算/存储资源的提供者,如何给数量众多且类型不同的应用系统提供便利的部署和配置支持,并能够进一步提高安全性和灵活性以及运行环境的后期扩展机制.

与此类似,移动计算和无线网络等也因其各自的特点带来了相应环境下应用系统部署的问题,如移动计算环境下在资源的有限性、网络互联的断续性和网络节点的动态变化性方面都将为应用系统的部署带来困难.因此,在包括云环境、移动环境在内的应用场景下,由于新的环境特点而引起的软件部署带来的新问题,也是值得研究的方面.

5 总 结

本文首先明确了软件部署概念的内涵及其对应的问题空间,并以此为基础建立了一个多侧面、细粒度的分析框架 W^4H.该框架由 5 个侧面、12 个维度构成,覆盖了软件部署中主体、客体、适用范围、方式策略和过程支持能力多个方面,能够对软件部署方法进行较为全面和综合的分析.W^4H 在分析内容方面紧扣软件部署概念的内涵,在分析维度方面围绕软件部署针对的问题和期望的目标展开.与已有工作相比,W^4H 的分析内容更加全面,分析粒度更为细致,分析目标更为突出.本文还根据部署技术的发展历程归纳总结了 3 种典型的软件部署模式和面向软件部署的一般过程模型,并基于 W^4H 对当前具有代表性的软件部署方法与技术进行分析和总结.案例研究结果表明,该分析框架能够对软件部署方法与技术进行较为全面的分析,对软件系统部署方法和技术的选择和开发具有重要的指导意义.本文提出的 W^4H 分析框架侧重于多方面的定性分析,在定量分析方面能力有限,在今后的工作中将建立更加全面和细粒度的定量分析指标,增强 W^4H 分析框架的定量分析能力.

References:

[1] The Berkeley/Stanford recovery-oriented computing (ROC) project. http://roc.cs.berkeley.edu

[2] Wang YM, Verbowski C, Dunagan J, Chen Y, Wang HJ, Yuan C, Zhang Z. STRIDER: A black-box, state-based approach to change and configuration management and support. In: Proc. of the 17th Conf. on Systems Administration (LISA 2003). 2003. 159–172.

[3] Papazoglou MP, Georgakopoulos D. Service-Oriented computing. Communications of the ACM, 2003,46(10):24−28.

[4] Tim O. What is Web 2.0: Design patterns and business models for the next generation of software. Int'l Journal of Digital Economics, 2007,65:17−37. http://mpra.ub.uni-muenchen.de/4580/

[5] Armbrust M, Fox A, Griffith R, Joseph AD, Katz R, Konwinski A, Lee G, Patterson D, Rabkin A, Stoica I, Zaharia M. A view of cloud computing. Communications of the ACM, 2010,53(4):50−58. [doi: 10.1145/1721654.1721672]

[6] Crnkovic I, Chaudron M, Larsson S. Component-Based development process and component lifecycle. In: Logistics I, ed. Proc. of the Int'l Conf. on Software Engineering Advances. Washington: IEEE Computer Society, 2006. 44−44. [doi: 10.1109/ICSEA.2006.28]

[7] Hall RS, Heimbigner DM, Wolf AL. Evaluating software deployment language and schema. In: Khoshgoftaar TM, Bennett K, eds. Proc. of the 14th IEEE Int'l Conf. on Software Maintenance. Washington: IEEE Computer Society, 1998. 177−185. [doi: 10.1109/ICSM.1998.738507]

[8] Dearle A. Software deployment: Past, present and future. In: Briand LC, Wolf AL, eds. Proc. of the 29th Int'l Conf. on Software Engineering 2007 Future of Software Engineering. Washington: IEEE Computer Society, 2007. 269−284. [doi: 10.1109/FOSE.2007.20]

[9] OMG. Deployment and configuration of component-based distributed applications, version 4.0. 2006. http://www.omg.org/cgi-bin/doc?formal/06-04-02.pdf

[10] InstallShield. http://www.flexerasoftware.com/products/installshield.htm

[11] Microsoft installer. http://www.microsoft.com/download/en/details.aspx?id=25

[12] Websphere. http://www-01.ibm.com/software/websphere/

[13] JOnAS. http://wiki.jonas.ow2.org/xwiki/bin/view/Main/WebHome

[14] Huang T, Chen NJ, Wei J, Zhang WB, Zhang Y. OnceAS/Q: A QoS-enabled Web application server. Journal of Software, 2004, 15(12):1787−1799 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/15/1787.htm

[15] SmartFrog. http://wiki.smartfrog.org/wiki/display/sf/SmartFrog+Home

[16] JASMINe. http://wiki.jasmine.ow2.org/xwiki/bin/view/Main/WebHome

[17] IBM Tivoli. http://www-01.ibm.com/tivoli

[18] HP OpenView. http://h20427.www2.hp.com/program/enterprise/cn/zh/technology/openview.asp

[19] Talwar V, Wu QY, Pu C, Yan WC, Jung GY, Milojicic D. Comparison of approaches to service deployment. In: Proc. of the 25th Int'l Conf. on Distributed Computing Systems. Washington: IEEE Computer Society, 2005. 543−552. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=1437116&contentType=Conference+Publications&sortType%3Dasc_p_Sequence%26filter%3DAND%28p_IS_Number%3A30952%29%26pageNumber%3D3 [doi: 10.1109/ICDCS.2005.18]

[20] Carzaniga A, Fuggetta A, Hall RS, Heimbigner D, van der Hook A, Wolf AL. A characterization framework for software deployment technologies. Technical Report, CU-CS-857- 98, University of Colorado, 1998.

[21] Carzaniga A. A characterization of the software deployment process and a survey of related technologies. Technical Report, 97-84, University of Colorado, 1997.

[22] Eilam T, Kalantar MH, Konstantinou AV, Pacifici G. Reducing the complexity of application deployment in large data centers integrated network management. In: Clemm A, Festor O, Pras A, eds. Proc. of the 9th IFIP/IEEE Int'l Symp. on Integrated Network Management. Washington: IEEE Computer Society, 2005. 221−234. [doi: 10.1109/INM.2005.1440790]

[23] Li Y, Qiu J, Sun KW, Chen Y. Modeling and verifying configuration in service deployment. In: O'Conner L, ed. Proc. of the 2006 IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society, 2006. 238−248. [doi: 10.1109/SCC.2006.73]

[24] Arnold W, Eilam T, Kalantar M, Konstantinou A, Totok A. Pattern based SOA deployment. In: Krämer BJ, Lin KJ, Narasimhan P, eds. Proc. of the 5th Int'l Conf. on Service-Oriented Computing. LNCS 4749, Berlin, Heidelberg: Springer-Verlag, 2007. 1−12. [doi: 10.1007/978-3-540-74974-5_1]

[25] Redlin C, Carlson-Neumann K. WebSphere process server and enterprise service bus deployment patterns. Technical Report, IBM, 2006. http://www.ibm.com/developerworks/websphere/library/techarticles/0610_redlin/0610_redlin.html#author1

[26] Arnold W, Eilam T, Kalantar M, Konstantinou A, Totok A. Automatic realization of SOA deployment patterns in distributed environments. In: Bouguettaya A, Krüger I, Margaria T, eds. Proc. of the 6th Int'l Conf. on Service-Oriented Computing (ICSOC 2008). LNCS 5364, Heidelberg: Springer-Verlag, 2008. 162−179. [doi: 10.1007/978-3-540-89652-4_15]

[27] Tsai W, Fu K. Error-Correcting isomorphisms of attributed relational graphs for pattern recognition. IEEE Trans. on System, Man, and Cybernetics, 1979,SMC-9(12):757−768. [doi: 10.1109/TSMC.1979.4310127]

[28] Luo J, Li Y, Qiu J, Chen Y. Declarative constraint framework for SOA deployment and configuration. In: Werner B, ed. Proc. of the 2008 IEEE Int'l Conf. on Web Services. Washington: IEEE Computer Society, 2008. 637−644. [doi: 10.1109/ICWS.2008.33]

[29] OMG. Object constraint language, version 2.0. 2006. http://www.omg.org/spec/OCL/2.0/PDF

[30] Kannan K, Narendra NC, Ramaswamy L. Managing configuration complexity during deployment and maintenance of SOA solutions. In: O'Conner L, ed. Proc. of the 2009 IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society, 2009. 152−159. [doi: 10.1109/SCC.2009.35]

[31] Maghraoui KE, Meghranjani A, Eilam T, Kalantar M, Konstantinou AV. Model driven provisioning: Bridging the gap between declarative object models and procedural provisioning tools. In: Steen MV, Henning M, eds. Proc. of the ACM/IFIP/USENIX 7th Int'l Middleware Conf. Middleware. LNCS 4290, Heidelberg: Springer-Verlag, 2006. 404−423. [doi: 10.1007/11925071_21]

[32] Rao JH, Su XM. A survey of automated Web service composition methods. In: Cardoso J, Sheth A, eds. Proc. of the 1st Int'l Workshop on Semantic Web Services and Web Process Composition. LNCS 3387, Heidelberg: Springer-Verlag, 2005. 43−54. [doi: 10.1007/978-3-540-30581-1_5]

[33] Minton S, Bresina J, Drummond M. Total-Order and partial-order planning: A comparative analysis. Journal of Artificial Intelligence Research, 1994,2(1):227−262.

[34] Dearle A, Kirby GNC, McCarthy A. A framework for constraint-based deployment and autonomic management of distributed applications. In: Werner B, ed. Proc. of the 1st Int'l Conf. on Autonomic Computing. Washington: IEEE Computer Society, 2004. 300−301. [doi: 10.1109/ICAC.2004.1301386]

[35] Dearle A, Kirby GNC, McCarthy A, Carballo JCD. A flexible and secure deployment framework for distributed applications. In: Emmerich W, Wolf AL, eds. Proc. of the 2nd Int'l Working Conf. on Component Deployment (CD 2004). LNCS 3083, Heidelberg: Springer- Verlag, 2004. 219−233.

[36] Lan L, Huang G, Ma LY, Wang M, Mei H, Zhang L, Chen Y. Architecture based deployment of large-scale component based systems: The tool and principles. In: Heineman G, Crnkovic I, Schmidt HW, Stafford JA, Szyperski C, Wallnau K, eds. Proc. of the 8th Int'l Symp. (CBSE 2005). LNCS 3489, Heidelberg: Springer-Verlag, 2005. 93−129. [doi: 10.1007/11424529_9]

[37] Medvidovic N, Taylor RN. A classification and comparison framework for software architecture description languages. IEEE Trans. on Software Engineering, 2000,26(1):70−93. [doi: 10.1109/32.825767]

[38] Huang G, Wang QX, Mei H, Yang FQ. Research on architecture-based reflective middleware. Journal of Software, 2003,14(11): 1819−1826 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/14/1819.htm

[39] Malek S, Medvidovic N, Mikic-Rakic M. An extensible framework for improving a distributed software system's deployment architecture. IEEE Trans. on Software Engineering, 2012,38(1):73−100. [doi: 10.1109/TSE.2011.3]

[40] Malek S. A user-centric approach for improving a distributed software system's deployment architecture [Ph.D. Thesis]. Los Angeles: University of Southern California, 2007.

[41] Wada H, Suzuki J, Oba K. Queuing theoretic and evolutionary deployment optimization with probabilistic SLAs for service oriented clouds. Congress on Services—I, 2009. 661−669.

[42] Godfrey P, Shipley R, Gryz J. Algorithms and analyses for maximal vector computation. The VLDB Journal, 2006,16(1):5−28. [doi: 10.1007/s00778-006-0029-7]

[43] Google AppEngine. http://code.google.com/appengine/

[44] Jayasinghe D, Pu C, Eilam T, Steinder M, Whalley I, Snible E. Improving performance and availability of services hosted on IaaS clouds with structural constraint- aware virtual machine placement. In: Jacobsen HA, Wang Y, Hung P, eds. Proc. of the 2011 IEEE Int'l Conf. on Services Computing. Washington: IEEE Computer Society, 2011. 72−79. [doi: 10.1109/SCC.2011.28]

[45] Wood T, Shenoy P, Venkataramani A, Yousif M. Black-Box and gray-box strategies for virtual machine migration. In: Proc. of the 4th USENIX Symp. on Networked Systems Design & Implementation. USENIX, 2007. 229−242. https://www.usenix.org/conference/nsdi-07/black-box-and-gray-box-strategies-virtual-machine-migration

[46] Xen. http://xen.org/

[47] Singh A, Korupolu M, Mohapatra D. Server-Storage virtualization: Integration and load balancing in data centers. In: Proc. of the 2008 ACM/IEEE Conf. on Supercomputing. Piscataway: IEEE Press, 2008. 1−12. http://dl.acm.org/citation.cfm?id=1413370.1413424&coll=DL&dl=ACM&CFID=85482705&CFTOKEN=73026936 [doi: 10.1109/SC.2008.5222625]

[48] Kramer J, Magee J. The evolving philosophers problem: Dynamic change management. IEEE Trans. on Software Engineering, 1990,16(11):1293−1306. [doi: 10.1109/32.60317]

[49] Vandewoude Y, Ebraert P, Berbers Y, D'Hondt T. Tranquillity: A low disruptive alternative to quiescence for ensuring safe dynamic updates. IEEE Trans. on Software Engineering, 2007,33(12):856−868. [doi: 10.1109/TSE.2007.70733]

[50] Magee J, Kramer J. Dynamic structure in software architectures. In: Garlan D, ed. Proc. of the 4th ACM SIGSOFT Symp. on Foundations of Software Engineering. New York: ACM Press, 1996. 3−14. [doi: 10.1145/250707.239104]

[51] Garlan D, Monroe R, Wile D. ACME: An architecture description interchange language. In: Johnson JH, ed. Proc. of the '97 Conf. of the Centre for Advanced Studies on Collaborative research. IBM Press, 1997. 1−15.

[52] Oquendo F. π-ADL: An architecture description language based on the higher-order typed π-calculus for specifying dynamic and mobile software architectures. ACM SIGSOFT Software Engineering Notes, 2004,29(3):1−14.

[53] Zhang S, Huang LP. Dynamic service evolving based on OSGi. Journal of Software, 2008,19(5):1201−1211 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/19/1201.htm [doi: 10.3724/SP.J.1001.2008.01201]

[54] Bruneton E, Coupaye T, Leclercq M, Quema V, Stefani JB. An open component model and its support in Java. In: Crnkovic I, Stafford JA, Schmidt HW, Wallnau KC, eds. Proc. of the 7th Int'l Symp. on CBSE 2004. LNCS 3054, Berlin, Heidelberg: Springer-Verlag, 2004. 7−22. [doi: 10.1007/978-3-540-24774-6_3]

[55] Godfrey MW, German DM. The past, present, and future of software evolution. In: Proc. of the Frontiers of Software Maintenance 2008 (FOSM 2008). Washington: IEEE Computer Society, 2008. 129−138. http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4659256&contentType=Conference+Publications&sortType%3Dasc_p_Sequence%26filter%3DAND%28p_IS_Number%3A4659234%29 [doi: 10.1109/FOSM.2008.4659256]

[56] Common information model. http://www.dmtf.org/standards/cim

[57] Oreizy P, Medvidovic N, Taylor RN. Runtime software adaptation framework, approaches, and styles. In: Schäfer W, Dwyer MB, Gruhn V, eds. Proc. of the 30th Int'l Conf. on Software Engineering Companion. New York: ACM Press, 2008. 899−910. [doi: 10.1145/1370175.1370181]

[58] Léger M, Ledoux T, Coupaye T. Reliable dynamic reconfigurations in a reflective component model. In: Grunske L, Reussner R, Plasil F, eds. Proc. of the 13th Int'l Symp. on CBSE 2010. LNCS 6092, Berlin, Heidelberg: Springer-Verlag, 2010. 74−92. [doi: 10.1007/978-3-642-13238-4_5]

[59] Aleti A, Grunske L, Meedeniya I, Moser I. Let the ants deploy your software—An ACO based deployment optimization strategy. In: Ceballos S, ed. Proc. of the 24th IEEE/ACM Int'l Conf. on Automated Software Engineering. Washington: IEEE Computer Society, 2009. 505−509. [doi: 10.1109/ASE.2009.59]

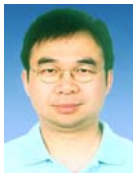
附中文参考文献:

[14] 黄涛,陈宁江,魏峻,张文博,张勇.OnceAS/Q:一个面向 QoS 的 Web 应用服务器.软件学报,2004,15(12):1787−1799. http://www.jos.org.cn/1000-9825/15/1787.htm

[38] 黄罡,王千祥,梅宏,等.基于软件体系结构的反射式中间件研究.软件学报,2003,14(11):1819−1826. http://www.jos.org.cn/1000-9825/14/1819.htm

[53] 张仕,黄林鹏.基于 OSGi 的服务动态演化.软件学报,2008,19(5):1201−1211. http://www.jos.org.cn/1000-9825/19/1201.htm [doi: 10.3724/SP.J.1001.2008.01201]

陈伟(1980－),男,江西高安人,博士生,助理研究员,主要研究领域为网络分布式计算,软件工程.



魏峻(1970－),男,博士,研究员,博士生导师,主要研究领域为网络分布计算,软件工程.

黄涛(1965－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为网络分布计算,软件工程.