



基于架构的软实时软件的自适应框架的研究^{*}

包晓安,张 娜,桂 宁,宋瑾钰

(浙江理工大学信息学院 杭州 310018)

摘要

随着移动设备的普及和普适计算的发展,用户需求多样化,实时系统在动态运行环境中得到了广泛应用。然而,现有自适应框架通常无法处理资源竞争和系统更改问题。本文提出了一种基于结构的自适应软件框架,为管理实时组件的互相依赖性和生命周期提供一种有效而统一的方法,为云计算服务提供实时计算方面的支持;提出了一种实时组件的运行服务以掌控全局,控制组件的整个生命周期,并能保持现有的实时组件承诺的需求,以应对运行时的变化。这一框架易于用约束描述语言进行扩展并集成其他约束解析政策。最后,本文通过一个模拟控制应用程序对本框架从适应性和性能两方面进行了测试。

关键词 实时系统;云计算;自适应框架;实时组件

1 引言

传统实时系统,如过程控制系统,通常工作在封闭和高度可预知的环境。这些系统也只能提供预先定义的、固定不变的服务。只要满足这些假设和条件,这种方法能够高度保证硬实时需求得以满足。然而,越来越多的系统被应用于动态的环境中,如在云计算系统中,服务器的配置不断地发生改变。在这样的动态环境中,传统的基于瀑布模型的实时应用程序开发模式不能有效工作^[1]。提供有效的实时软件的重配置支持,提供整个系统的生命周期中系统配置演变支持,已经成为研究的热点问题。

这种开放和不断演变的执行环境需要新的办法来建立和管理实时应用程序。但是,由于系统资源的缺乏,这些已安装的应用程序通常在一定程度上是互相依存的,不同

的应用程序会相互竞争 CPU、内存、网络或其他系统资源。这种依赖性可能会导致违背实时程序不能满足亟需满足的实时特性指标^[2]。

传统的基于实时操作系统和实时调度的解决方案无法应对这样复杂的依赖性。系统作业调度器本身不可能理解这种高级别的应用程序的相互依赖性。在某种意义上,实时操作系统离应用程序级的问题太“遥远”,因此无法正确决定自适应策略,如选择哪些组件、预留哪些资源。对于云计算来说,这就意味着系统的调度程序无法很好地使用后台的服务器资源。

由设计自适应实时系统的经验可知,为了支持动态变化的实时应用程序,任何一个有效的解决方案应该有以下3个重要方面^[3]。第一,跨应用程序的资源需求管理。第二,组件依赖的依赖必须显式化,包括显式功能的依赖和由于争夺有限的可用资源而产生的隐含功能的依赖。第三,系统高层模型应该可以被用于驱动相应实时任务的较低级别的管理。

本文从不同的角度解决这些问题,引入一种基于结构

^{*} 国家自然科学基金资助项目(No.61202050),浙江省自然科学基金资助项目(No.LY12F02041),浙江省科技厅一般面上资助项目(No.2011C33G2060108),浙江省钱江人才计划基金资助项目(No.2012R10009),嘉兴市科技局重点和面上资助项目(No.2011BY11003)

的管理框架,即声明实时组件的框架(DRCom 框架),跨越安装应用程序域。通过从它的元数据及其组件管理接口引出每个实时组件的不同属性,可以为当前实时系统构建一个准确的总体结构模型。为了保证实时应用程序的某些性能,可根据不同的适应策略对该模型进行映射以调整低级实时任务的属性和状态。得益于面向服务的体系结构,该框架易于用其他依赖解析策略进行扩展,以处理不同的系统需求。本文定性和定量地证明了本体系结构的有效性,可以用于云计算的资源调度管理。

2 软件架构

为了支持动态变化的实时应用程序,本文提出了一种基于架构的自适应软件设计模式,即声明实时组件框架(DRCom 框架),用以生成自适应实时应用程序。

2.1 架构设计目标

为了在提供实时支持的同时,应对高度动态的运行环境,本文的软件架构提供了 3 种功能支持,即实时组件动态支持、实时组件依赖性支持、可自定义的全局自适应支持,实现了将系统生命周期的所有请求当作一个整体进行控制管理,能快速得到当前系统配置,并能管理软硬件各组件间两种类型的依赖,即关于业务逻辑的功能性依赖和关于 QoS 需求的非功能性依赖。利用当前系统配置以及组件间的功能性和非功能性依赖的相关描述定义自定义的适应策略,并根据资源的动态可用性和自适应策略,采用不同的约束策略。

2.2 系统需求

本模型使用可扩展的元数据文件中声明 DRCom 的各个实时属性。由于不同的实时应用程序具有不同的特定的

域和功能,不可能提出一个万能的自适应逻辑,因此系统的体系结构必须适应不同的环境。

本框架的设计可以使用不同种类的描述语言和适应逻辑。用 3 个关键模块,即描述模块、推理模块和动作模块,满足各种需求。描述模块进行依赖关系和配置说明,推理模块进行推理说明,动作模块进行相应的动作描述。参考文献[4]和现有的大多数研究,都是在运行时静态地集成这 3 个模块,与此相反,本框架通过运用面向服务的模型,将这 3 个模块当作服务提供者来执行,各服务提供者通过服务接口松散耦合。这种设计使得未来复杂自适应模块可以容易地融入本框架中。

2.3 结构框架

图 1 所示为本文所述结构框架的示意。该框架由 3 个主要部分组成,分别对应于第 2.2 节所述的 3 个模块,其他部分提供辅助性服务。

其中,资源依赖性表示机制(自定义语言解析服务,作为描述模块)允许开发人员用自定义的语言描述组件依赖性;自定义约束解析服务(作为推理模块)允许用户或管理员指定特定域系统的自适应策略;描述性实时组件运行时执行环境 DRCom (动作模块)是本框架的核心,它管理 DRCom 实例并执行解析服务推理出的动作,负责监测由资源管理服务和 OSGi 系统发出的更改事件,并由 DRCom 维护的 DRCom 实例注册表来保存已安装的实时组件实例的信息。

本框架设计了一个完整的组件生命周期模型和一个相应的实时组件管理界面。根据自定义约束解析服务做出的决策,DRCom 完全控制组件的生命周期。自适应过程由组件状态更改和外部事件源的通知(如资源管理器或简单

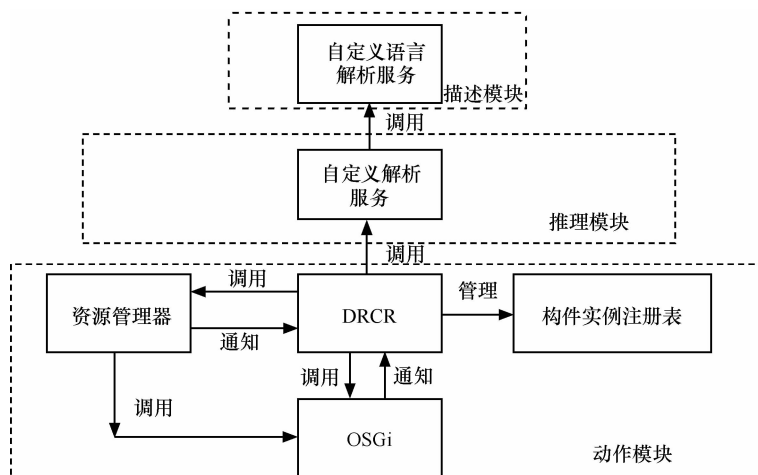


图 1 结构框架



的周期时间)触发。

3 架构组件

实时组件说明在元数据文件中定义,该文件描述了特定实时组件的主要特性。此外,本文使用支持复杂描述语言的混合实时组件模型^[5]。

在 DRCom 实现中,实时组件配置表示组件实例的功能需求和非功能需求。如上所述,框架应能够支持不同的描述语言。但从实现的角度看,纯粹分离的方法意味着事先不了解组件元数据格式,这将导致相当大的实现复杂性和不必要的性能开销。与复杂的非功能需求相比,组件的功能需求部分有较好的结构和稳定性。为了在性能和复杂度之间达到一个平衡点,使用两相模式描述和处理组件的功能性和非功能性依赖。其中,功能部分由 XML (extensible markup language, 可扩展标记语言) 定义,由 DRCR 解析;非功能部分的组件资源的先决条件描述,由外部的自定义解析服务进行解析,并对已解析的数据进行处理,然后将结果发送到 DRCR。

图 2 所示为元数据文件的一个片段,其中包含了一个能够根据需要返回用户感兴趣的区域(一幅图像数据的子集)的智能相机组件的功能和非功能需求。

其中,name:被用作组件调用名,因此必须是全局唯一的;enabled:控制组件是否启用,其默认值为真。组件的 implementation 元素定义组件实现类的名称,DRCR 通过引用此属性创建组件实例,因此是组件必不可少的元素。组件的 inport 和 outport 元素定义通信方法,通过它们可进行组件间数据共享。组件可能有 0 个或多个输入、输出端口。outport 指定实时组件所需输出的内容,它包含一个用作通信调用的名称属性。而除了名称属性外,输入端口还具有以下属性。interface:通信接口,它可以是共享内存、实时 FIFO 导管或邮箱。type:传输的数据类型,可以是整型或字节。size:数据类型的基数。与 outport 相比,inport 有一个额外属性,

即 optional 属性,该属性显示此端口依赖性是否是组件执行所必需的。此外,组件描述也可以有许多属性,可以通过管理接口来利用这些属性元素配置实时组件实例。

4 约束关系解析

本文实时组件框架的一个关键过程是约束关系解析,它负责识别实时组件之间以及资源之间的关系,并根据当前系统环境推断出系统可能的反应。

当系统配置更改时,例如,有新的实时组件的出现或者资源管理服务组件提示有明显的资源变化,DRCR 将被激活。首先它将检查已有的 DRCom 约束和依赖;如果功能部分成功通过检查并满足要求,DRCR 将判定是否能在不危害其他组件实时合同的有效性的条件下满足组件的非功能需求,如果能够满足,这一新组件将被激活。

为了与两相 DRCom 配置一致,运用两相约束解析。一个是功能依赖解析相,用于在发生系统更改时检查各个实时组件功能依赖;另一个是自定义约束关系解析相,由于本框架使用 OSGi 面向服务的模型,因此系统管理员可以通过注册自定义的解析服务改变系统解析行为,以适应特定的系统配置。

4.1 功能依赖解析

功能依赖解析主要包括以下的两个步骤。

(1) 循环检测

应用程序中由实时组件构成,通常可以建模为一个组件联通图,该图通常是有向无环图。在任务路径上,组件上的任务具有一组输出,这组输出是该路径上下一组件上的任务的一组输入。一组组件描述可创建一个循环依赖,此时至少有一个组件的配置不能得到满足,当 DRCR 试图满足组件配置时,它将检测循环引用。这时,DRCR 运行时将记录一条错误消息,且不激活任何可能创建循环依赖的组件配置。

(2) 端口兼容性检测

为了满足功能性依赖,所有已启用和激活的组件输入

```
<? xml version="1.0" encoding="UTF-8"?>
<dr:component name="camera" desc="this is a smart camera controller"
enabled="true">
<implementation bincode="ua.pats.demo.smartcamera.RTComponent"/>
<outport name="images" interface="RTAI.SHM" type="Byte" size="400"/>
<inport name="windposition" interface="RTAI.SHM" type="Integer" size="1"
optional="false" />
<property name="window size" type="Integer" value="40" />
<service-specific language="UA.PATS.Language.SRDF">
<periodictask cpuid="1" ram="20MB" priority="1"
executiontime="1000ns" period="10000ns" deadline="2000ns"/>
</dr:component>
```

非功能化的元数据

图 2 组件元数据的格式实例

端口应有相应的输出端口。这意味着它们的端口名称、接口类型、数据类型和数据大小应完全匹配。图 2 中的智能相机端口需要一个“windowposition”输入端口。为了满足端口的兼容性检查,名称为“ua.pats.trajectory.windows”的已激活的组件实例和按如下定义的输出端口 <output name="windowposition" interface="RTAI.SHM" type="Integer" size="1"/>必须已经在注册表中部署和启用。

4.2 自定义解析过程

在实际环境中,根据资源的动态可用性和系统特定需求,需要不同的资源描述方法。因此将自定义解析服务设计为面向服务的模型,以提供可自定义的支持。

通过使用不同的自定义解析服务提供者,系统行为可以在不更改应用程序主要功能逻辑的情况下轻易地改变,这缓解了重新编译和加载整个应用程序的需要。一个基本的解析策略是,必须在激活组件配置之前满足所有基本的资源需求。不管怎样,运用特定的上下文或依赖性知识,可以进行更加灵活、复杂的解析,例如,通过停止低优先级任务或具有可选端口需求的任务,给高优先级实时组件释放更多资源。此外,某些规则引擎,如 Jess^[6],也可以集成到解析服务中。

多个自定义解析服务提供商可以共存于本系统中,DRCR 会选择一个与当前系统环境相匹配的候选服务提供者。在当前实现中,系统上下文是一个由管理员管理的全局属性,下面笔者利用上下文推理策略,获取更准确、灵活的上下文信息,以便使用更适当的自定义解析服务。

4.3 自适应配置时序图

图 3 所示为自适应软件框架的动态调整时序。本方案中,组件 B 需要用组件 A 的输出满足其功能约束。由于组件 A 配置已经存在实例(ComAInstance),因此 DRCR 将成功解析 B 的功能约束。然后访问内部解析服务和外部自定义服务,如果两个服务均返回积极结果,DRCR 将根据组件 B 的配置创建并激活它的实例(ComBInstance)。如果 ComAInstance 被停止,DRCR 将得到通知,并再次请求其内部和外部自定义服务以检测新的满足或不满足条件的组件实例,当 DRCR 发现 ComBInstance 目前尚未满足,那么 ComBInstance 将被禁用。

5 系统仿真

本文使用基于场景的仿真,测试基于框架的自适应软件的动态调整模式,并定性和定量地验证所提的框架的特性。

5.1 系统实现

系统使用 OSGi^[7]框架作为实现平台。OSGi 技术一个是通用中间件平台,其开放标准简化了新功能和服务的引入。笔者的基本开发平台是 Equinox,它是 Eclipse 组织开发的免费开放源 OSGi 平台。组件实现中,使用 DRCom 模型,该模型最初是为动态可配置的反射实时系统的建模而设计的,它有一个基本的管理界面,能够统一管理组件的生命周期和属性,每个组件都与一个记录其抽象概念(如通信接口、属性以及引用约束)的元数据文件相关联。这种方法提供了一种元数据级的反射和自检,每个组件都标记

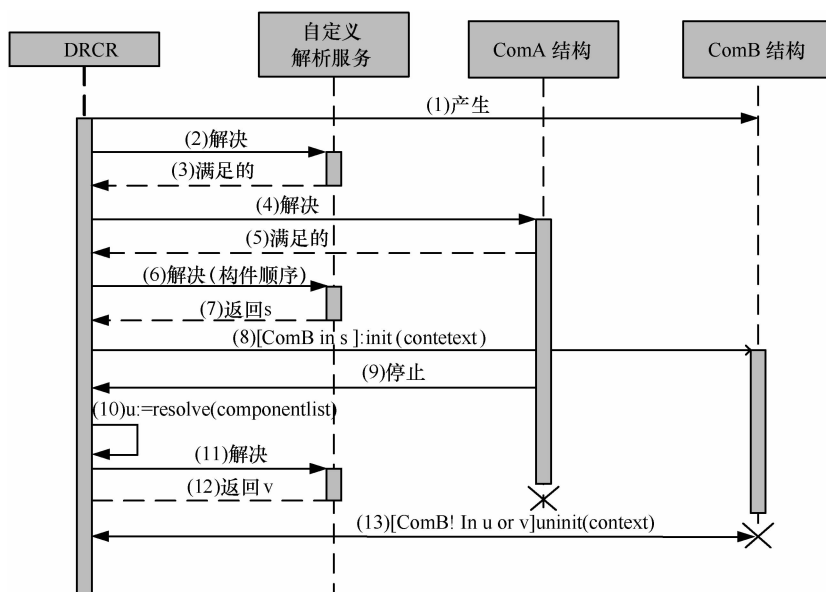


图 3 自适应配置时序



了任务相关的配置和验证信息。

5.2 模拟配置

由一个仿真的机顶盒系统进行模拟。该机顶盒系统运行 2 个应用软件,一个是电视应用程序,一个是视频文件转换工具。各个软件的核心模块(解码器和转码器)的执行组件都是以周期任务的形式执行。电视应用程序的解码组件由一个周期为 33.37 ms 的实时任务实现,其优先级为 2、执行时间约 8 ms、截止时间为 12 ms;转码器组件由一个周期为 50 ms 的实时任务实现,其优先级为 1、执行时间约 1 ms、截止时间为 30 ms。其中,RTAI 系统使用的计划策略是 FIFO。为了显示这两个组件之间的相互作用,其他组件只占用少量的 CPU 时间进行实现。此外,传输方法是异步的(共享内存),这样就能够重点研究两个编码模块的性能。

由于视频解码执行时间会随着视频流内容的变化而改变,为了更加清晰地表明不同应用程序之间的相互影响,用一个在每轮计算中使用了同等恒定 CPU 时间的计算函数代替解码函数。

5.3 仿真结果

对电视应用程序中解码组件的执行时间进行 5 000 次观测,表 1 显示了有和没有自适应框架支持时组件的执行时间。为了消除冷启动和自适应过程带来的影响,在系统启动并能稳定执行后,才开始记录测量数据。执行时间用微秒(μs)表示。

正常情况下,解码器的执行时间大多为 8 ms 左右,有自定义解析服务(它会禁用转码模块)时最大执行时间约 8.13 ms;没有自定义解析服务时,系统会同时运行这两个应用程序,解码器任务执行时间的抖动很大,其平均偏差是 3 563.2 μs ,远远大于有框架解析机制时的 5.23 μs ,这最终可能导致视频解码任务的瞬态时间问题,包括帧缺失、数据溢出等。

表 1 执行时间的比较

上下文/执行结果	有自适应框架	无自适应框架
平均执行时间(μs)	8 134.387	10 987.205
平均偏差(μs)	5.23	3 563.2
最小执行时间(μs)	8 125.3	8 129.7
最大执行时间(μs)	8 163.4	17 236.9
溢出(μs)	0	1 080

6 结论和进一步工作

本文描述了一种支持运行时自适应的软件设计模式,以应对复杂的实时系统的动态演变和连续调度问题。其中,实时合同由组件元数据指定,组件实例由依赖解析服务和实时合同系统管理,DRCR 服务对当前系统配置进行全局管理。这种系统能够推理出系统配置更改的原因,能在遵守指定的实时组件实时合同的前提下执行某些操作,并使得各实时组件不必监视系统状态,就能调用其他可用的动态实时组件。虽然本文所述框架是基于 OSGi 中间件的,但研究结果将会普遍运用于其他基于结构的管理系统。

虽然本文描述了一个为基于组件的应用程序提供实时属性的完整方法,但仍有许多地方需要进一步研究。其中之一就是如何使用元数据来描述那些组件特有的自适应接口和功能,进而使系统具有更灵活、更强大的自适应能力。

参考文献

- 1 Hassan Gomaa, Koji Hashimoto, Minseong Kim, *et al.* Software adaptation patterns for service-oriented architectures. Proceedings of ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, 2010:462~469
- 2 Gui Ning, Florio V D, Sun Hong, *et al.* A framework for adaptive real-time applications: the declarative real-time OSGi component model. Proceedings of the 7th Workshop on Adaptive and Reflective Middleware(ARM), Leuven, Belgium, 2008:35~40
- 3 Gui Ning, Florio V D, Sun Hong, *et al.* A hybrid real-time component model for reconfigurable embedded systems. Proceedings of the 2008 ACM Symposium on Applied Computing, Fortaleza, Brazil, 2008:1509~1596
- 4 Santambrogio M D, Rana V, Beretta I, *et al.* Operating system runtime management of partially dynamically reconfigurable embedded systems. Proceedings of the 8th IEEE Workshop on Embedded Systems for Real-Time Multimedia, Scottsdale, AZ, United states, 2010:1~10
- 5 Colmenares J A, Kim K H, Zhang Z, *et al.* Real-time-component based software architecture for QoS-adaptive networked multimedia applications. Proceedings of the 13th IEEE International Symposium on Object, Component, and Service-Oriented Real-Time Distributed Computing (ISORC), Carmona, Seville, Spain, 2010:133~142
- 6 Jess: the rule engine for the Java platform. [http:// www.jessrules.com/jess/index.shtml](http://www.jessrules.com/jess/index.shtml)
- 7 OSGi service platform core specification. <http://www.osgi.org>, 2011

[作者简介] 包晓安, 浙江理工大学副教授, 硕士生导师, 主要研究方向为服务组合、自适应软件系统、智能信息处理; 张娜, 浙江理工大学副教授, 主要研究方向为服务组合的可靠性分析; 桂宁,

博士, 浙江理工大学讲师, 主要研究方向为自适应软件系统、基于服务架构、中间件技术、设计模式; 宋瑾钰, 浙江理工大学讲师, 主要研究方向为形式化方法、模型驱动开发方法和自适应软件系统。

An Architecture-Based Adaptation Framework for Soft Real-Time Applications

Bao Xiaolan, Zhang Na, Gui Ning, Song Jinyu

(School of Information & Electronics, Zhejiang Sci-Tech University, Hangzhou 310018, China)

Abstract Driven by the proliferation of mobile and pervasive computing, real-time systems are increasingly used in dynamic changing environments with variable user needs, hosting real-time applications ranging in number and nature over time. However, existing adaptation frameworks normally are incapable to cope with other competing and changing concerns. This paper proposes an architecture-based framework for managing real-time components' dependence and lifecycle in an effective and uniform way. A real-time component runtime service is proposed here to maintain the global view, control the whole lifecycle of the components and keep existing real-time components' promised contracts in the face of run-time changes. This framework is designed to be easily extended with other constraint resolving policies as well as dependence descriptions languages and can be used to provide real-time computing services for cloud computing. At end of this paper, the framework is tested from both adaptation and performance aspects by a simulated control application.

Key words real-time system, cloud computing, adaptation framework, real-time component

(收稿日期: 2012-08-15)

· 简讯 ·

HID Global 推出 OMNIKEY 非接触式 双频读卡器 可集成到打印机及 单点登录解决方案中

2012 年 9 月 3 日, 深受信赖的安全身份验证解决方案引领者 HID Global 宣布, 推出新的 OMNIKEY 5427 CK 读卡器, 该产品支持 CCID (chip card interface device, 芯片接口装置) 标准及读卡器—键盘信号转换, 使系统集成商能设计免驱动安装的非接触式读卡器, 避开了现场复杂的软件周期管理问题。这类非接触式读卡器常用于打印机和复印机中的安全打印验证、单点登录解决方案 (single sign on solution, 此用户身份验证方案允许用户以单一登录方式来访问多个应用程序)、门禁系统和网络。

OMNIKEY 5427 CK 的用户利用读卡器—键盘转换功能, 可以简便地通过读卡检索凭证卡内的数据, 并利用按键模拟功能将卡片数据直接输入应用程序, 而无需用户手

工输入凭证卡数据。针对嵌入式应用, HID Global 还提供实现相同功能的 OMNIKEY 5127 CK 读卡器模板。

为了简便、快速地支持完整读卡器和纯读卡器电板的集成, HID Global 还提供了 OMNIKEY 开发人员工具组件 (developer tool kit, DTK), 以帮助节省集成时间, 并加快产品上市。

HID Global 产品营销高级总监 Helmut Dansachmüller 表示: “越来越多的企业客户将凭证卡应用于门禁以外, 更要求采用一张卡片满足多种应用需求, 例如安全打印验证和单点解决方案。新的 OMNIKEY 5427 CK 不仅支持这类综合应用, 其功能还有助于合作伙伴加速产品集成, 使合作伙伴能更快将产品推向市场。”

OMNIKEY 5427 CK 可同时支持高频和低频标准技术, 如 iCLASS、iCLASS SE、HID PROX、MIFARE Classic 和 MIFARE DESFire EV1 技术, 使用户借助多技术的混合凭证卡, 在不同读卡技术之间进行转换。