基于BPMN的Web服务并发交互机制

江东明 薛锦云
(武汉大学软件工程国家重点实验室 武汉 410073) (江西师范大学高性能计算中心 南昌 330022)
(武汉大学计算机学院 武汉 410073)

摘 要 如何抽象描述复杂业务流程的交互是服务计算研究的重要问题。大量的国内外研究虽然关注业务流程建模,但却忽略了建模过程中组件之间的并发交互描述和实现。针对Web服务并发交互,提出一种基于BPMN的Web服务并发交互的形式化模型。首先,采用BPMN描述业务流程中的并发交互模式;其次,将BPMN并发交互模式映射到Orc语言;最后,Web服务实例表明,所提方法可有效支持Web服务并发交互抽象建模。

关键词 BPMN,Orc,Web服务交互
中图法分类号 TP393 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2014.08.010

Web Service Concurrent Interaction Mechanism Based on BPMN

JIANG Dong-ming XUE Jin-yun
(State Key Laboratory of Software Engineering, Wuhan University, Wuhan 410073, China)
(Key Laboratory of High Performance Computing, Jiangxi Normal University, Nanchang 330022, China)
(School of Computer Science, Wuhan University, Wuhan 410073, China)

Abstract Business Process Modeling (BPM) aims to describe and abstract the interaction of complex systems associated, is an important part of Service-oriented computing (SOC). While many researches focus on BPM, they ignore the crucial problem how to describe the concurrent interaction between the constituted components. Therefore, this paper presented a formal model of concurrent interaction of Web services based on BPMN. First, we used BPMN to describe the concurrent interaction among Web components. Secondly we maped the concurrent interactive model of BPMN to Orc language, a function language to describe concurrent orchestration of Web service. Finally, we illustrated our approach by mean of an example.

Keywords BPMN, Orc, Web service interaction

1 引言

随着网络技术的快速发展以及市场竞争的日益激烈,软件开发者和用户都迫切需要一种新型、高效的面向Web的软件组件集成开发范式。作为一种新型分布式计算范型,Web服务自从提出以来,就得到工业界和学界的广泛关注和研究应用。借助于Web载体平台,Web服务目标是实现动态、平台无关的服务资源的无缝组合和复用,而Web服务组合的本质就是多个Web服务之间的交互协作,以满足用户多样化需求。在开放的Web服务网络环境中,多个Web服务在逻辑层次上同时工作,即并发执行。由于资源限制或控制流依赖等原因,Web服务之间形成错综复杂的交互关系。例如大量用户希望购买火车票,同时向中国铁路客户服务中心网站并发请求,结果造成系统因突发海量并发处理而导致性能恶化。

因此,如何正确设计和开发并发分布式程序日益成为服务计算中的一个关键问题。解决方法一般从两个方面入手,一是采用基于云计算的资源均衡与调度机制;二是从程序实现角度入手,采用并发分布式语言开发机制。作为工业界业务流程建模标准,BPMN得到广泛的研究和应用。然而,BPMNR[2]等建模语言缺乏精准语义,虽然可以描述Web服务静态交互,却不能高效验证Web服务交互的正确性和相容性;另一方面,目前面向BPMN的形式化模型关注于验证,却没有考虑在业务流程实现阶段如何引导设计者正确处理Web服务并发交互。

为此,本文提出一种面向BPMN的并发交互模式的转换方法。首先,我们应用BPMN描述Web服务的交互模式;然后,由于Orc语言[6]具有描述分布式并发Web服务组合的特性,我们采用Orc语言表示基于BPMN的Web服务并发交互模式,即将Web服务交互机制映射到Orc语言。

本文的主要贡献在于以下2个方面:首先建立从BPMN到Orc语言的映射关系,并用Orc的连接子实现了BPMN中网关的路由功能和泳道的分组功能。其次,从Web服务交互

到稿日期:2013-06-11 返修日期:2013-07-15 本文受国家自然科学基金重大国际合作项目(61020106009)资助。
江东明(1975—),男,博士生,工程师,CCF会员,主要研究方向为服务计算和面向构件的软件开发,E-mail:jiangdm@aliyun.com;薛锦云(1947—),男,博士生导师,主要研究方向为服务计算和形式化方法。

模式出发，设计开发了 Web 服务交互模式所对应的 Orc 表达式，用以帮助开发者复用交互模式。

2 背景知识

2.1 BPMN

业务流程指一类为了实现价值增值目标的多个简单/子任务的组合，而业务流程设计和实现是服务计算中的重要研究领域。为了协助用户、IT 开发者共同对业务流程建模，BPMI (Business Process Management Initiative) 2004 年提出业务流程建模符号 BPMN (Business Process Modeling Notation)，其目标是为服务组合语言如 BPEL 提供图形化建模符号，使业务流程设计和实现有机衔接。一般而言，企业先从业务需求出发，通过 BPMN 描述和设计系统业务流程模型。而模型的主要作用在于协助不同参与部门、组织深入分析，并理顺业务流程专家和 IT 专家的沟通协作和决策机制。然后 IT 专家将 BPMN 转化为可执行的 BPEL 程序，从而实现业务流程的设计和无缝连接。

BPMN 图形元素(如图 1 所示)分为 4 类：流对象、连接对象、泳道、工件(artifact)。泳道负责组织和分类业务流程模型中各个组件；工件负责为模型提供注释。流对象由活动对象、事件和网关 3 类组成，用于定义业务流程的各个子任务。其中活动对象是对业务行为的抽象；网关描述业务流程的路由关系，如分支、归并(如图 2 所示)。连接对象表示流对象(即子任务)之间关联，例如顺序流表示流对象之间顺序控制流关系；消息流表示流对象之间数据流依赖关系。

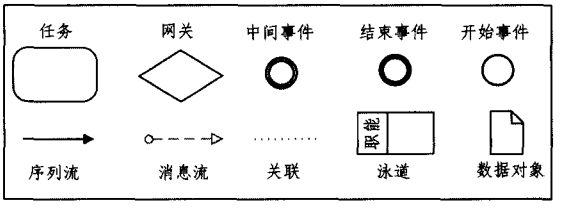

图 1 BPMN 基本元素

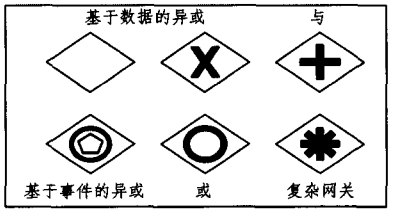

图 2 BPMN 网关类型

根据网关的入度和出度，网关可分为判断网关和归并网关；依据逻辑关系则可分为异或、或、与和复杂网关 4 类。异或网关分为基于事件、基于数据两组，两组的区别在于判定条件不同。归并网关也有两类：并行分裂和并行归并。

例 1 客户在京东的网购过程以及对应的 BPMN 业务流程模型如图 3 所示。客户查询商品，对中意商品下订单；而销售部验证订单是否合格，与客户、仓库协调；仓库负责存储和直接邮商品给客户。

由于采用半形式化描述方式，BPMN 没有明确清晰的语义[9]，因此 BPMN 不能规范和形式化 Web 服务的交互行为，也不能形式化验证其结构和行为层次的正确性。

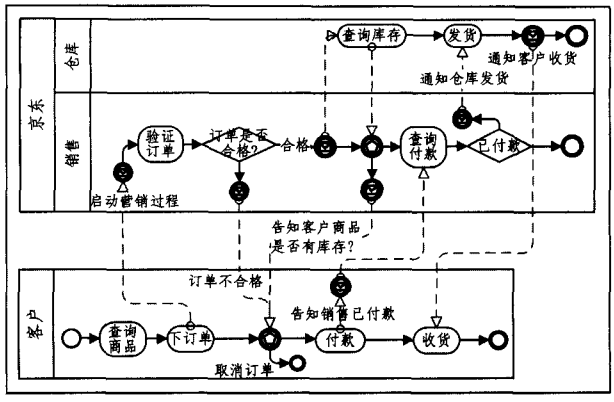

图 3 京东网购流程的 BPMN 模型

2.2 Web 服务交互

模式(Patterns)泛指解决某一类问题的方法论，即通过观察重复出现的事件，抽象其呈现规律，提出解决问题的一般性方法。例如 OOP 设计模式(design pattern)帮助开发者更快、更好地运用已成功的软件设计思路和体系结构，同时也帮助理解系统设计思路，便于软件维护和演化。结合工作流和设计模式思想，Van der Aalst 从控制流角度提出了工作流模式(workflow patterns)[7]。工作流模式可分为控制(Control)、资源(Resource)、数据(Data)和异常处理(Exception Handling) 4 部分，包括顺序、同步、多选、多合并、鉴别器(discriminator)等多种模式。

随着软件规模增大，软件系统内组件交互关系呈现多样化和复杂化，组件之间的交互关系不仅有双方关系，还有多方服务交互形态。文献[4]提出服务交互模式(Service Interaction Patterns)，用以描述各种 Web 服务之间的交互协作关联。服务交互模式与工作流模式的主要区别在于交互模式是从消息流角度分析 Web 服务之间的依赖关系，而工作流模式则是依据控制流视角。当 Web 服务组合中组件属于不同企业时，基于严格同步消息传递或控制流的模式不再适用。而对于基于异步消息通信方式的 Web 服务，服务交互模式[3]是解决服务交互场景的有效方法。

例 2 用 4 个例子说明 Web 服务交互模式的特点，并用 BPMN 描述。

1)消息竞争模式：多个参与者向信息接收方发送一组信息，接收方按照预定规则从中选择并处理一个信息(见图 4)。

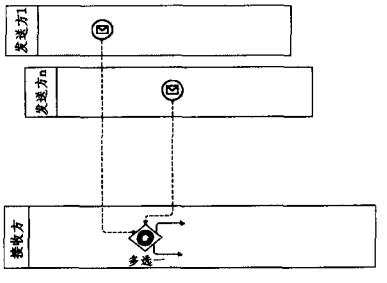

图 4 消息竞争模式

2)一对多消息发送模式：发送方向多个业务流程参与者发送信息(见图 5)。

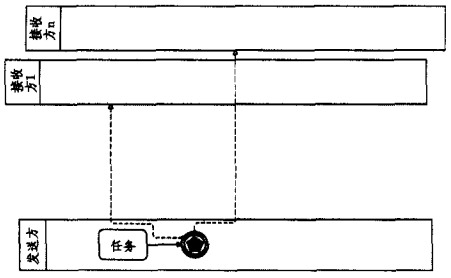

图 5　一对多消息发送模式

3)一对多消息接收模式:多个参与者向信息接收方发送逻辑相关的信息,接收方汇总所有接收消息,并形成单个逻辑请求。要求消息在规定时限内到达,否则逻辑请求无效(见图 6)。

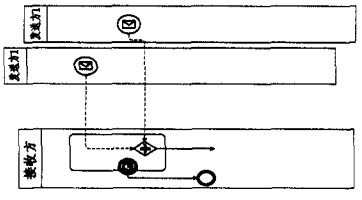

图 6　一对多消息接收模式

4)引用性请求模式:发送者 A 先发送请求给 B,B 然后将 A 的请求分解处理,分别交予 $C_1,\cdots,C_n$ 处理。最后,$C_1,\cdots,C_n$ 直接与 A 交互,发送信息(见图 7)。

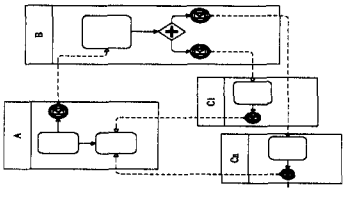

图 7　引用性请求模式

2.3　Orc

Orc 语言[6]分为 Orc 演算和 Orc 编程语言两部分:Orc 演算属于进程演算,是语言核心部分,功能等同于顺序编程语言中的 λ 演算;Orc 编程语言是一种函数式语言,是 Orc 演算的语法糖衣。作为一种描述服务组合的脚本语言,Orc 由 3 种类型构成:Site、连接子和 Site 定义。Site 是服务的构件化表现形式,提供组装、绑定功能。Site 调用方式是严格调用方式,即只有 Site 所需的所有参数被赋值,才可调用 Site,而在调用机制上,Site 采用传值调用,而表达式是传名调用。Orc 有 4 类组合子,用以分离 Web 服务计算功能和连接功能:

1)并行连接子 $f|g$:表达式 f,g 并发执行,并发布任意表达式的结果。

2)顺序连接子 $f>x>g$:先执行表达式 f;然后将 f 所求值与变量 x 的实例相绑定;对于任意变量 x 的实例,分别执行 g 所对应的实例。

3)剪裁连接子 $f<x<g$:同时执行表达式 f,g;然后将 g 所发布的第一个值与变量 x 相绑定;再中止 g(变量 x 可能出现在表达式 f 中)。

4)否则连接子 $f;g$:先执行 f;当 f 中止且没有发布任何结果时,才执行 g。

Orc 表达式定义为:$def\ E(x)=F$,其中 E 是表达式标识符,x 是表达式参数,F 是表达式的实现部分。Orc 语法部分如图 8 所示。

```
f,g∈Expression ≙
    stop Basic Expression
    |e(x̄) Site call
    |f|g Parallel Combinator
    |f>x>g Sequential Combinator
    |f<x<g Pruning Combinator
    |f;g Otherwise Combinatior
    |D#f prefixing declaration
D∈Declaration ≙
    def E(x̄)=f Site Definition
    |val x=f Val declaration
    |def class E(x̄) =f Class declaration
Program ≙f
```

图 8　Orc 语法部分

例 3

1)Fork-Join 同步:$Tuple(x,y)<x<f<y<g$

由于 $Tuple(x,y)$ 采用严格调用方式,使得表达式 f,g 同步。

2)求一个列表中的所有子集,子集之和等于指定数值:Orc 程序采用动态规划方法并行处理。

```
def parsum(0,[])=[]
def parsum(n,[])=STOP
def parsum(n,x:xs)=parsum(n−x,xs)>ys>x:ys|parsum(n,xs)
```

3　Web 服务交互模式的 BPMN

3.1　BPMN 的 Orc 表示

模型驱动体系架构(MDA)的核心思想是分离业务规约与实现技术,并以业务模型驱动系统软件开发过程。在业务流程建模邻域,BPMN 已广泛应用于 MDA 的企业级解决方案。我们将 BPMN 映射到 Orc 是因为 Orc 有 3 点特性:1)Orc 语法简洁,并具有严密的语义模型,可推导 Web 服务交互的行为正确性;2)Orc[1]是一种并发分布式 Web 服务组合的脚本描述语言,适于描述开放环境中的 Web 服务复杂交互关系;3)Orc 分离 Web 服务组件的计算和交互功能,可支持所有的工作流模式和服务交互模式。因此我们选择 Orc 为目标语言,建立从 BPMN 到 Orc 的映射关系。

如上节所述,BPMN 包括 4 组:流对象、连接对象、工件和泳道。工件的作用是注释业务流程,它对应于 Orc 的注释部分。文献[5]提出 BPMN 的 Orc 描述方法,主要用于描述网关路由作用(详细内容见文献[5])。

1)顺序流:Orc 表示为 Orc 的顺序连接子>。

2)并发分裂:用 Orc 并行算子|表示为等价 Orc 代码:$R>(F|G)$。

3)并发归并:$(F,G)>(x,y)>R$,其中 x,y 分别是 F,G 所发布的值。

4)任务循环:$Loop(g,F)=g>b>\text{If } Cond(b,F)\gg Loop(g,F)$。

但是文献[5]只考虑 BPMN 子集与 Orc 语言的映射关系,并没有涵盖 BPMN 2.0。例如,文献[5]没有给出触发器所对应的 Orc 表达式,也没有考虑条件顺序流和默认顺序流;另外,BPMN 中消息流和泳道是描述不同参与者之间的交互的重要方式,而文献[5]没有映射消息流和泳道,不能描述多

泳道的交互关系。本文从触发器、连接对象和泳道3个方面将BPMN扩充到Orc的映射。

触发器类型与事件类型相关，可细分为3类：开始事件相关触发器、中间事件相关触发器和结束事件相关触发器。限于篇幅，我们只描述BPMN开始事件相关触发器与所对应的Orc程序，如表1所列。

表1 开始事件相关触发器的Orc代码

触发器类型	描述	对应Orc代码段
消息	业务流参与者消息触发而启动流程	Email(a,x)≫F，其中a是接收方，x是消息内容，F是待启动的任务
计时	在指定时间内启动任务	Rwait(t)≫F，其中t表示所指定的时间
规则	满足规则所定义的条件时启动任务	Ift(R)≫F，其中R表示规则所定义的条件
链接	链接另一过程的结束事件，前过程结束才启动后一过程	前一过程：F>x>mail(a,x)；后续启动过程：G(x)，其中x为G中的一个待绑定参数

BPMN连接对象分为3类：顺序流、消息流和关联。而顺序流也可细分为3种：普通顺序流、条件顺序流和默认顺序流，如表2所列。文献[5]只用Orc的顺序连接子<描述普通顺序流，并没有考虑其他几种类型连接对象。

表2 连接对象的Orc代码

连接	描述	对应Orc代码段
条件顺序流 ⟼	在满足条件时才启动目标任务	Ift(R)≫F|Iff(R)≫G，若条件R为真，启动任务F，否则启动任务G
默认顺序流 ⟼	用于网关的默认路径	采用优先级方式处理：(Ift(x)≫F|Rwait(t)≫G)
消息流 ⇢	表示消息的接收和发送	发送方行为：Email(a,x)，其中a是接收方，x是消息内容；接收方：F(x)，F是待启动任务

BPMN泳道用于分类BPMN中模型元素，其中池(pool)表征业务流程的参与组织；道(lane)表示池的若干组成部分。利用泳道和消息流可以为服务计算中的多方建立交互模型。由于Web服务的松耦合特性，不同泳道(组织)或同一泳池中不同道之间的交互均采用消息传递方式；而在同一泳道内，服务组件交互关系以控制流为主。本文用Orc的定义和通道来封装泳道。

val in=channel()

val out=channel()

def pool(P,in,out)=((F>x>out,put(x))|(in.get()>y>G(y))≫pool(P,in,out)

其中，P为泳池，x为所接收的消息流，in，out表示P向协作的Web服务组件发送或接收消息的通道，其中通道*channel*()是Orc内置的site。

3.2 Web服务交互模式Orc表示

如同设计模式和工作流模式，Web服务交互模式[4]可以帮助IT专家挖掘业务流程模型中的交互共性形态，有利于Web服务构件的重用。

例4 本文将例2的交互模型例子用Orc表述

1)消息竞争模式

$val\ in=Table(n,lambda(\_)=BoundChannel())$

发送方：

$def\ send_i=F_i>x>in(i).\ put(x),i\in[1,n]$

接收方：

$def\ recv=G(x)<x<(in(1).\ get()|\cdots|in(n).\ get())$

2)一对多消息发送模式

$val\ out=Table(n,lambda(\_)=BoundChannel())$

发送方：

$def\ send=F>x>(out(1).\ put(x)|\cdots|out(n).\ put(x))$

接收方：

$def\ recv_i=out(i).\ get()>x>F(x),i\in[1,n]$

3)一对多消息接收模式

$val\ in=Table(n,lambda(\_)=BoundChannel())$

发送方：

$def\ send_i=F>x>in(i).\ put(x),i\in[1,n]$

接收方：

$def\ recv=((in(1).\ get()>x_1|\cdots|in(n).\ get()>x_n)>(x_1,\cdots,x_n)>y)|Rtime(t)>SKIP)>(Ift(y)>F|Iff(y)>G)$

4)引用性请求模式

先建立通道：

$val\ in=channel()$

$val\ out_1=Table(n,lambda(\_)=BoundChannel())$

$val\ out_2=Table(n,lambda(\_)=BoundChannel())$

发送方A：

$def\ send=F>x>in.\ put(x)>Rtime(t)>(out_2(1).\ get()>z_1|\cdots|out_2(n).\ get()>z_n)>(z_1,\cdots,z_n)$

中介方B：

$def\ mid=in.\ get()>x>(x_1,\cdots,x_n)>(out_1(1).\ put(x_1)|\cdots|out_1(n).\ put(x_n))$

引用方：

$def\ ref_i=out_1(i).\ get()>x>G(x)>y>out_2(i).\ put(y),i\in[1,n]$

4 实例

本节以图3为例说明BPMN转化为Orc的过程。首先利用消息传递和泳池建立京东销售、仓库和客户之间的通信过程模型，如图9所示。

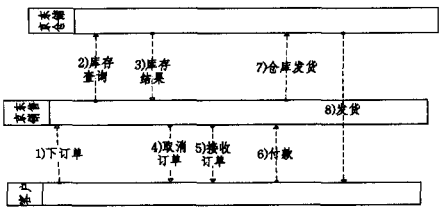

图9 京东网购的BPMN交互过程模型

分析个泳道直接的交互模式，我们发现消息传递1)到5)都属于一对多接收或者发送模式，而消息传递6)到8)属于引用性请求模式。先建立通道，分别用于客户与销售、仓储3方异步通信。

再用Site封装每个泳道：

客户端Site：

$def\ Client(c(1),c(3))=$

$find\_good>x>c(1).\ put(x)>Rtime(t)>c(1).\ get()>y>(Ift(y)>;STOP)>payoff>z>c(1).\ put(z)>Rtime(t)>c(3).\ get()>k>receive(k)>STOP$

京东销售端Site：

$def\ Saler(c(1),c(2))=$

$c(1).get()>x>c(2).put(x)>Rtime(t)>c(2).get()>y>c(1).put(y)>(Ift(y)>Rtime(t)|Iff(y)>STOP)>(Rtime(t)>STOP|c(1).get()>z)>(Ift(z)>k>c(2).put(k);STOP)$

京东仓储端 Site：

$def\ Store(c(2),c(3))=$

$c(2).get()>x>check\_store(x)>y>c(2).put(y)>(c(2).get()>z;STOP)>c(3).put(z)>delive(x)>STOP$

在 Internet 中，由于多个京东网购业务流程的实例并发执行，若简单并行执行 3 个泳道所对应的 Site：(*Saler*|*Client*|*Store*)，会导致业务流程出现死锁情况。其原因是在 BPMN 泳道分割，即水平分割，从控制流角度考虑业务运行，当业务流程跨越多个企业或控制域时，可执行业务流程则因交互顺序不匹配而导致死锁。

为了避免交互顺序不匹配情况，本文从 Web 服务交互模式设计角度，对多泳道的交互过程进行细化。换而言之，我们还考虑到服务交互模式垂直分割泳道的问题，这样不但可保证 Web 服务交互顺序的一致性，而且可利用已有的好的交互模式。另外，垂直分割将 Web 服务中计算部分和连接交互部分分离，因而利于设计开发并发性、扩展性的 Web 服务系统。因此，本文从 Web 服务交互思路重新分割和组织 Site，策略是在同一泳道中一个交互对应一个 Site。

1)下订单时，客户与销售的交互：

$def\ client_1(c(1))=find_good>x>c(1).put(x)$仓库

2)查询、销售与仓库管理人员的交互

$def\ sale_1(c(1),c(2))=c(1).get()>x>(c(2).put(x)|Rtime(t)>STOP)$

$def\ store_1(c(2))=c(2).get()>x>check\_order(x)>y>c(2).put(x)$

3)销售将是否有货的信息告知客户

$def\ sale2(c(1),c(2))=c(2).get()>x>c(1).put(x)>(Ift(x)>Rtime(t)|Iff(x)>STOP)$

4)客户自己判断是否购买或取消

$client_2(c(1))=c(1).get()>x>(Ift(x)>x;STOP)$

5)若购买，客户付款，并与销售联系，然后销售引荐仓储与客户直接联系收货事宜。由于该场景属于例 4 引用性请求模式，我们可以直接应用引用性请求模式的 Orc 表示。

结束语 并发分布式计算是云计算、服务计算的重要基石之一，如何描述和抽象 Web 服务的并发交互机制是当前一个重要的课题。本文应用 Orc 语言和 BPMN，讨论概念层次上 Web 服务交互模式，并从结构和功能层面分析和提炼出业务流程的各节点任务。下一步将从 BPMN[8] 和 Orc 语义层次细化工作。

参 考 文 献

[1] 游珍，薛锦云，应时. Apla 语言中并发分布式机制的研究[J]. 计算机科学，2012，39(1)：104-109

[2] Chinosi M，Trombetta A. Bpmn：An introduction to the standard [J]. Comput. Stand. Interfaces，2012，34(1)：124-134

[3] Decker G，Barros A. Interaction modeling using bpmn[C]// Proceedings of the 2007 International Conference on Business Process Management (BPM'07). Berlin，Heidelberg：Springer-Verlag，2008：208-219

[4] Decker G，Puhlmann F，Weske M. Formalizing service interactions[C]// Proceedings of the 4th international conference on Business Process Management (BPM'06). Berlin，Heidelberg：Springer-Verlag，2006：414-419

[5] Goel N，Shyamasundar R K. An executional framework for bpmn using orc[C]// Services Computing Conference (APSCC)，2011 IEEE Asia-Pacific. 2011：29-36

[6] Misra，Jayadev，Cook，et al. Computation orchestration：A basis for wide-area computing[J]. Software and Systems Modeling (SoSyM)，2007，6(1)：83-110

[7] Van Der Aalst W M P，Ter Hofstede A H M，Kiepuszewski B，et al. Workflow patterns[J]. Distrib. Parallel Databases，2003，14(1)：5-51

[8] Wong P Y，Gibbons J. A process semantics for bpmn[C]// Proceedings of the 10th International Conference on Formal Methods and Software Engineering (ICFEM '08). Berlin，Heidelberg：Springer-Verlag，2008：355-374

[9] Wong P Y H，Gibbons J. Formalisations and applications of bpmn[J]. Sci. Comput. Program.，2011，76(8)：633-650

(上接第 24 页)

[29] Wang Ying-xin，Cui Yan，Tao Pin，et al. Reducing Shared Cache Contention by Scheduling Order Adjustment on Commodity Multi-Cores[C]// 2011 IEEE International Parallel & Distributed Processing Symposium. 2011：984-992

[30] Yan J，Zhang W. WCET analysis for multi-core processors with shared L2 instruction caches[C]// Proc of the 14th IEEE Real-Time and Embedded Technology and Application Symposium. 2008：1179-1186

[31] Zhang W，Yan J. Accurately estimating worst-case execution time for multi-core processors with shared direct-mapped instruction caches[C]// Proc of the 15th IEEE Embedded and Real-time Computing Systems and Application. 2009：455-463

[32] Lv Ming-song，Wang Yi，Nan Guan，et al. Combining Interpretation with Model Checking for Timing Analysis of Multi-core Real-Time Software[C]// the 31st IEEE Real-Time Systems Symposium (RTSS). 2010：1176-1183

[33] Chen Fang-yuan，Zhang Dong-song，Wang Zhi-ying. Static analysis of run-time Inter-thread interference in shared Cache multi-core architecture based on instruction fetching timing[C]// 2011 Proceedings of IEEE International Conference on Computer Science and Automation Engineering. 2011：208-212

[34] 陈芳园，张东松，林聪，等. 基于取指执行时序范畴的多核共享 Cache 干扰分析[J]. 计算机研究与发展，2013，50(1)：206-217

[35] Liang Y，Suhendra V. Timing analysis of concurrent programs running on shared cache multi-cores[C]// Proc of the 30th IEEE Real-Time System Symposium. 2012：57-67