动态工作流挖掘模型及算法研究

梁俊锋 张祖平 龙 军
(中南大学信息科学与工程学院 长沙 410083)

摘 要 工作流挖掘是工作流再设计与分析的一项关键技术,目的在于从企业已有信息系统的日志记录中提取与实际应用相关的业务流程,从而提高工作流建模的客观性。从实际应用的角度出发,提出基于动态工作流网的挖掘模型,并设计了可行的算法。该算法从零日志开始,在工作流执行期间柔性地构建工作流模型和生成日志记录,动态生成工作流网,在很大程度上解决了常规挖掘方法在挖掘复杂结构效率低、准确性差、依赖完整的日志记录和不支持柔性工作流等方面的问题。算法的实用性在实际应用系统中得到了很好的证明。

关键词 动态工作流,工作流挖掘,工作流网

Model and Algorithms for Dynamic Workflow Mining

LIANG Jun-feng ZHANG Zu-ping LONG Jun
(School of Information Science and Engineering, Central South University, Changsha 410083, China)

Abstract Workflow mining is a key technology in workflow re-design and analysis field. It is generated to improve the objectivity of workflow modeling by extracting the actual business-related process from records of the running information systems in enterprises. However, most of Workflow Mining methods are based on records of information systems running in companies. On the other hand, they have bad performance in mining complex tasks and flexible workflow. To solve those problems, we proposed a Dynamic Workflow Mining model. In this model, there is no record from the beginning, workflow nets and records were dynamically generated. The given example was used to demonstrate improvements on model accuracy and description ability of the proposed algorithms.

Keywords Dynamic workflow, Workflow mining, Workflow net

1 引言

工作流是利用计算机技术实现业务流程的重组、管理与自动化的计算模型。在工作流计算模型中,业务人员在计算机管理环境下按照一定的规则互相协作,互相传递信息、文档和任务,从而实现某一目标。工作流的生命周期包括工作流设计、工作流配置、工作流执行和工作流诊断4个阶段[1]。在工作流设计阶段,建模者根据对企业流程的业务逻辑的理解,从中提取可以计算机化的部分并描述成可执行的业务流程模型。因此,建模的客观性完全依赖于建模者的主观理解,使得工作流建模变成一个复杂的过程,而且结果往往与企业实际的业务逻辑有很大的偏差。

工作流挖掘(Workflow Mining)又称为过程挖掘(Process Mining)或工作流模式挖掘(Workflow Schemas Mining),它的目标是倒转过程,收集和利用已有信息系统的数据,从而支持工作流的设计和分析[1]。工作流挖掘是为了增加人工建模的客观性,从企业已有的信息系统提取实际发生过的业务过程。自Agrawal与Gunoptllos首次将过程挖掘的思想应用到工作流领域以来,提出了一系列工作流的挖掘算法,其中较为典型并具有实际应用价值的有如下4个:

(1) Agrawal提出基于活动依赖性有向图的挖掘算法[2]。该算法假定日志中每一个活动都是有一定的先后关系,即没有并行结构,从中挖掘出活动之间依赖关系,最后用有向图表示这种依赖关系。

(2) Aalst等提出基于工作流网的α算法[3]。α算法用启发式方法去掉日志中的噪音,从无噪音的日志中挖掘活动间的关系,并用基于Petri网的工作流网(WF_net)表示挖掘的结果。α算法可以探测日志的变化,有利于柔性实现,是目前最为流行的工作流挖掘算法。

(3) Herbst等提出基于随机活动图的归纳式算法[4]。这种算法在随机行为图的基础上,用随机模型去掉日志中的噪音数据,通过ADONIS模型把并行、选择结构呈现给用户。

(4) Schimm等提出基于块状结构模型的多阶段挖掘方法[5]。这种算法可以把活动间的关系通过4块结构(即顺序结构、并行结构、选择结构和循环结构)来表示。

这些算法的研究主要集中在从日志记录的事件的二元顺

到稿日期:2010-02-05 返修日期:2010-04-21 本文受国家自然科学基金项目(60873081,60970095,M0921005),湖南省自然科学基金(07JJ6122)资助。

梁俊锋(1984—),男,硕士生,主要研究方向为数据库、工作流,E-mail:liang_csu@163.com;张祖平(1966—),男,博士,教授,博士生导师,主要研究方向为信息融合与信息系统、参数计算与生物计算、网络容错路由算法及协议,E-mail:zpzhang@mail.csu.edu.cn(通信作者);龙 军(1972—),男,博士,副教授,主要研究方向为网络资源管理与服务技术。

序关系中挖掘一些启发式规则，再基于这些规则从日志中提取过程模型。这些算法在实际应用中有如下弱点：

(1) 这些算法普遍依赖于已有系统的日志记录，要求业务必须有完整而正确的日志。

(2) 工作流模式中普遍存在的复杂结构，如重复任务(duplicate tasks)、隐藏任务(hidden tasks)、隐式库所(implicit places)和非自由选择结(non-free choices)等，很难被挖掘出来。

(3) 柔性工作流要求在工作流执行的过程中能够动态变更工作流模式，而这些算法很难适应动态变更的日志，所以不能满足柔性工作流的要求。

上述缺点使得模式挖掘的实用性大大降低。针对以上问题，提出了一系列的改进算法：文献[6]提出了能发现角色的基于角色活动图的挖掘算法；文献[7]提出了能发现隐含任务的基于可覆盖工作流网(S-Coverable Workflow Nets)的挖掘算法；文献[8]提出了能挖掘重复任务的改进 α 算法；文献[9]提出了支持并发模式的挖掘算法。为了满足实际应用的需求，本文在前人研究的基础上提出了基于动态工作网的挖掘模型。本文主要思想分为两个部分：

(1) 采用动态工作流网作为工作流的形式化描述工具。

(2) 将动态挖掘算法分为两个阶段：在动态构建阶段，从空日志记录开始，在工作流执行期间柔性地构建工作流模式和生成日志，在模式中记录下每一条路径被执行过的次数，并把这些模式按照主题分类保存在模式库中；在动态挖掘阶段，按主题在模式库中搜索匹配的模式，在规范的动态工作流网结构上减去不可信的分支，挖掘出具有标准化结构的工作流模式。

本文第 2 和第 3 节给出了相关的定义和形式化的描述、构建与算法；第 4 节给出了一个应用实例。

2 相关定义和形式化描述

为了适应动态工作流挖掘的要求，本文采用动态工作流网 DWF-net(Dynamic WF-net)来形式化描述工作流模式。工作流网 WF-Net(Workflow net)是 Aalst 在 Petri 网的基础上定义的。Petri 网作为一种从过程的角度出发描述和分析复杂系统的模拟工具，具有形式化的语义定义、直观的图形表达和严格的数学基础等优点，很适合描述具有并发、异步、分布和包含不确定性因素的复杂系统。因此，基于 Petri 网的工作流建模越来越广泛应用于流程的业务模型描述中。

一个 Petri 网 $PN=(P,T,F)$[11]，其中

(1) P 是有限个库所的集合。

(2) T 是有限个变迁的集合且 $P\cap T=\Phi$。

(3) F 是弧的集合 $(P\times T)\cup(T\times P)$。

PN 称为工作流网，当且仅当它满足下面的两个条件：

(1) PN 的两个特殊库所 i 和 o。库所 i 是一个起始库所，没有前驱节点，即 $\cdot i=\Phi$。库所 o 是一个终止库所，没有后继节点，即 $o\cdot=\Phi$。工作流网必须有一个起始节点和一个终止节点。

(2) 如果在 PN 中加入一个新的变迁 t，使 t 连接库所 o 与 i，即 $\cdot t=\{o\}$，$t\cdot=\{i\}$，这时所得到的 PN 是强连接的。工作流网中没有孤立状态节点，即所有节点都必须在起始节点和终止节点的通路上。

定义 1 动态工作流网 DWF-net$=\{S,E,DT,DP,F\}$。其中 S 为源库所，即开始节点；E 为汇集库所，即结束节点；DT 为动态变迁集；DP 为动态库所集；F 为连接 DT 和 DP 之间的弧。

定义 2(动态变迁集 DT) 表示工作流过程中的任务节点，动态变迁用矩形表示。动态变迁具有一个三元组属性 $(state,dt\_entry(),dt\_body())$，其中

(1) $state$：变迁的状态，$state=\{$就绪，等待，完成$\}$。

(2) $dt\_entry()$：变迁的入口函数，规定了变迁就绪的条件 $DP\rightarrow\{DP$ 的布尔表达式$\}$。例如变迁 dt 的入口函数 c：$cod(c)=\{x\mid x\in DP,x\in\cdot dt\}$，$dom(c)=\{$true,false$\}$，其中 $cod(c)$ 表示 c 的定义域，$dom(c)$ 表示 c 的值域。当 $c=$true 时变迁 dt 转换为就绪状态，$c=$false 时 dt 变迁为等待状态。

(3) $dt\_body()$：变迁执行函数，规定了变迁 dt 发生后如何将托肯输出到后继库所集中：$DP\rightarrow\{$控制变量赋值语句$\}$。例如变迁 dt 的执行函数 r：$cod(r)=\{x\mid x\in DP,x\in dt\cdot\}$，$dom(r)=\{$ture,false$\}$。

定义 3(动态库所集 DP) 表示变迁被激活或者变迁执行的条件集。动态库所用圆圈表示，根据出现在变迁的前与后分别分为输入库所和输出库所。

定义 4(F 为连接库所和变迁的弧)

$dom(F)\cup cod(F)=DT\cup DP$

$dom(F)=\{x\mid\exists y:(x,y)\in F\}$

$cod(F)=\{y\mid\exists x:(x,y)\in F\}$

F 的属性是个四元组 $F=(DT,DP,D,W)$。其中 DT 是弧连接的变迁；DP 是弧连接的库所；D 是弧的方向，$D=\{$in,out$\}$，in 表示弧的方向是从库所指向变迁，当前库所作为输入库所，out 表明弧的方向是从变迁指向库所，当前库所是作为输出库所；W 为弧的权重，也就是该分支被执行的次数，作为弧的置信度。

3 模型构建与算法

3.1 模型的构建

如图 1 所示，模型的主要组成部分是工作流模式库和动态修改接口，分别描述模型的静态结构和动态行为。

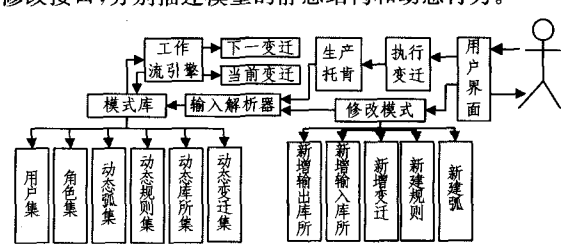

图 1 模型组织图

3.1.1 模型的静态结构

模式库中存储所有按主题分类的动态工作流网。模式库的组成包括：

(1) 动态变迁集(DT)，存储了已创建的变迁集。在动态构建的过程中约定变迁一定要可达，即每个变迁一定要有输入库所。变迁表示工作流实例中的任务或者活动。

(2) 动态库所集(DP)，存储已创建的库所集，库所表示工作流实例中的任务发生条件和任务发生后的结果。为了保证构建的连续性，在动态构建的过程中约定每个非结束库所

必须有后继变迁。

(3) 动态规则集(DR),存储各个变迁的入口函数和执行函数。

(4) 动态弧集(DF),存储变迁和库所之间的连接和被执行的次数,作为挖掘时的置信依据。

(5) 角色集(R),存储工作流参与者的抽象,与变迁存在一对多的映射关系,与参与者之间也存在一对多的映射关系。

(6) 参与者集(U),是角色的实例,是变迁的执行者和模型动态构件者。

3.1.2 *模型的动态行为*

模型的动态行为分为两个阶段:工作流模式动态构建阶段和动态挖掘阶段。

在动态构建阶段,按照柔性工作流的要求,支持用户在流程执行过程中能动态构建和修改变迁、库所、规则和角色,详见算法 1。在此阶段中有如下操作:

(1) 新增变迁 $ADD\_T(dt)$:$DT=DT\cup\{dt\}$。

(2) 新增库所 $ADD\_P(dp)$:$DP=DP\cup\{dp\}$。

(3) 新增弧 $ADD\_F(f)$:$F=F\cup\{f\}$。其中,$f=(dt,dp,d,w)$,dt 为现有变迁集中的某一变迁,dp 为现有库所集的某一库所,d 为弧的方向,w 为该弧的权重,初始值为 0。

(4) 添加规则 $ADD\_R(dr)$:$DR=DR\cup\{dr\}$,dr 包括两种类型:变迁 dt 的入口函数 $dt\_entry()$,变迁 dt 的执行函数 $dt\_body()$。

(5) 执行变迁的执行函数 $dt\_body()$:产生托肯到 dt 的输出库所集 P 中,其中 $p\subseteq DP\wedge\cdot p=dt\wedge p\subseteq dom(dt\_body())$,同时增加托肯流过弧的权重,将这些弧的置信权重加 1,即 $F.W++$。

(6) 触发就绪变迁 $activate()$:搜索变迁集,执行变迁的入口函数 $dt\_entry()$,将 $dt\_entry()==ture$ 的变迁的 state 设置为就绪状态。

在挖掘阶段,即工作流模式的提取和标准化阶段,根据输入的主题的和置信粒度,输出完整的工作流网。具体算法见算法 2。

3.2 算法描述

每一个模式的一个执行实例,就是一个工作任务单(以下简称工单)从工作流网开始节点到结束节点的一次流转。每一个工单的当前步骤处理者,可以根据需要采取相应的动态构建模式的行为。

算法 1 柔性构建工作流

输入:工单主题 f_theam,工单号 f_no,当前变迁 dt_c,可信粒度 δ。

输出:下一变迁。

Step1 判断 dt_c 是否为第一步变迁(即 $if(dt\_c==S)$),如果是则转算法 2,否则转 Step2。

Step2 搜寻模式库中的弧集 F,查找 dt_c 的可信输出库所集 DP_j,其中:

$\forall p\subseteq DP\_j:\cdot p=dt\_c$

$\wedge\exists f(DT,DP,D,W):f.DT=dt\_c\wedge f.DP=p\wedge f.D=out$

$\wedge f.W>=\delta$

即输出库所 $p\_out\in DP\_j$ 与 dt_c 构成的弧 $f(DT,DP,D,W)$ 中:$f.DT=dt\_c$,$f.DP=p\_out$,说明弧变迁端为 dt_c,库所端为 p_out;$f.D=out$,表明 p_out 为 dt_c 的输出库所;$f.W>=\delta$,表明弧的置信权重必须大于等于 δ。

Step3 用户判断 DP_j 是否能满足要求,如果能则转 Step4,否则转 Step5。

Step4 执行变迁 dt_c:

1)调用 dt_c 的函数体 $dt\_body()$,产生托肯到 DP_j;

2)增加 f 的可信度,$f.W++$;

3)调用 $activate()$ 触发所有满足条件的变迁为就绪状态。

Step5 添加当前变迁 dt_c 的输出库所:

1)查找 DP,如果有合适的 dp_i 就继续,否则就转 Step7;

2)在弧集 F 中添加弧 $f(dt\_i,dp\_i,out,0)$。

3)添加 dt_i 执行后果规则 $ADD\_R(dr)$,在规则库中添加 $dt\_body()$,添加 dt_c 产生托肯到 dp_i 的赋值语句。

Step6 工作流引擎查找出下一变迁集 dt_i,用户判断 dt_i 是否满足要求,如果是转 Step9,否则转 Step8。

Step7 新增库所 $ADD\_P(dp)$:$DP=DP\cup\{dp\}$,查询 DT 是否有适合作为 dp 的后继和前驱变迁,如果有则添加该库所的后继和前驱变迁,否则执行 Step8。

Step8 新增变迁 dt_j:

1)遍历 DP,查找是否有满足建立该变迁的输入库所和输入出库所,如果是则继续,否则转 Step7;

2)添加弧 f,其中 $f.DT=dt\_j$,$f.DP=dpi$,$f.d=out$;

3)新增 dt_j 的入口函数 $dt_entry()$和体函数 $dt\_body()$。

Step9 结束。

算法 2 根据输入主题和可信粒度输出当前的工作流网

输入:工单主题,可信粒度 δ。

输出:工作流网。

Step1 搜索模式库,找到匹配输入主题的工作流网 WF_NET_i,如果没有找到到则转 Step2,否则转 Step3。

Step2 初始化工作流网 DWF-net=$\{S,E\}$。

Step3 遍历工作流网,剪掉所有可信度小于输入可信度的分支弧,如果 $f.C<\delta$,$F=F-\{f\}$。

Step4 遍历库所集,剪掉所有孤立节点。对于非起始节点和非终止节点 p,如果 $p\cdot=\emptyset$ 或者 $\cdot p=\emptyset$,$DP=DP-\{p\}$。

Step5 遍历变迁集 DT,剪掉所有孤立的节点。对于变迁节点 dt,如果 $dt\cdot=\emptyset$ 或者 $\cdot dt=\emptyset$,$DT=DT-\{dt\}$。

Step6 遍历规则集合,去掉所有包含被去掉库所的逻辑条件。

Step7 重复 Step4-Step6,直到所有库所和变迁都为不为孤立节点为止。

3.3 模型和算法的分析

3.3.1 *正确性的分析*

一个基于工作流网建模的处理过程是正确的当且仅当[7]:

(1) 从状态 i 开始可达到的每一个状态 M,都存在着一个变迁序列可以使状态 M 到达终态 o。形式化如下:$\forall M(i\xrightarrow{*}M)\Rightarrow(M\xrightarrow{*}O)$。

(2) 状态 o 是从 i 开始的唯一的具有至少一个令牌的可达状态,形式化表示为 $\forall M(i\xrightarrow{*}M\wedge M\geqslant O)(M=O)$。

(3) 在(PN,i)中没有死的变迁。形式化表示为:$\forall t\in T\exists M,M':i\xrightarrow{*}M\xrightarrow{t}M'$。

其中,第一条规定了从初始状态(状态 i)开始,总可能达到库所 o(状态 o)中有一个标记的状态;第二条规定了一个标记放入库所 o 的时刻,所有其他的库所都应该是空的。这两条也称为"正确结束"。最后一条规定了从初始状态 i 开始不存在死变迁(任务),即对案例不存在任何作用的变迁。

定理 1 DWF_NET 可以转换成事实上的 WF_net。

(1) 当 DWF_net 的弧 F 的置信度满足挖掘算法指定的置信度时，即 $F.C \geqslant \delta$ 时，F 可以转换成普通工作流网 WF_net 的弧。

(2) 动态变迁，动态库所对应 WF_net 的库所和变迁。

(3) 通过去掉孤立节点，保证了所有节点都是可达的，从而保证了 DWF_net 与 WF_net 结构一致性。定理得证。

从结构的角度，DWF-net 可以被转化为 WF-net。文献[10,11]给出了工作流网 WF-net 合理性的定义、定理以及验证算法，并通过 Petri 网的相关理论进行了证明。

3.3.2 效率分析

一个共有 L 个不同主题工作流网的模式库，平均每个工作流网有 M 个变迁、N 个库所。遍历某个变迁的库所时，要遍历所有的库所表，所以算法的时间复杂度为 $O(L*M*N)$。可以看出，在工作流模式库中挖掘，时间复杂度只与模式数有关而与日志数无关，从而大大提高了挖掘的性能。

4 应用实例

本模型已经在长沙移动流程自动化管理系统中得到应用。下面以一个 IP 地址申请的流程来说明。IP 地址申请流程的基本任务如表 2 所列，基本转换条件如表 1 所列，基本分支如表 3 所列。该流程在一开始时需要动态创建：

(1) 第一个用户进来(第一次新建工单)，系统首先初始化工作流网 DTset={S,E}，T={}，如图 2 所示，S 为起始节点，E 为结束节点。

图 2 初始化工作流网

(2) 第一步，用户选择新建节点：$T1$ 发送工单，同时创建 $T1$ 的输入输出库所(见图 3)：

$P1$={发工单=true}，$S\bullet=T1$，$T1\bullet=P1$。

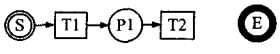

图 3 新建变迁和库所

有了 $P1$ 就要创建它的后继变迁 $T2$(班组长审核)，输出库所为 $P2$={IP 类型=公网 IP}，$P3$={IP 类型=私网 IP}，$T2$ 的前置条件 $dt\_entry()$={$P1$}，即 $P1$ 中有托肯。$T2$ 为或分支(OR_split)，如图 4 所示，它的执行函数 $dt\_body()=P2 \parallel P3$，只能选择其一作为输出库所。

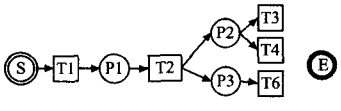

图 4 OR_split

接着后面变迁的用户动态地创建下去，直到得到如图 5 所示的工作流网。至此，构建了整个工作流网，同时记录下了每个分支被执行过的次数。变迁、库所、路径和条件可以在流程运行时动态修改，实现了工作流的柔性。当需要标准化时，可以按照输入的置信粒度，挖掘出对应的工作流网。

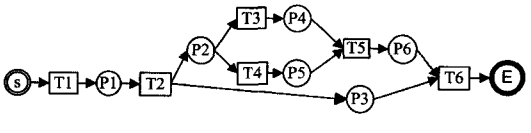

图 5 模式图

表 1 动态库所 DP

WF_NO	P_NO	Value
1	S	开始
1	P1	发工单=true
1	P2	IP 类型=私网
1	P3	IP 类型=公网 IP
1	P4	本部门主任审核=同意
1	P5	网络部主任审核=同意
1	P6	P4&&P5=true
1	E	结束

表 2 动态变迁 DT

WF_NO	T_NO	T_DESC	dt_entry()	dt_body()
1	T1	发工单	S	P1
1	T2	班长审核	P1	P2||P3
1	T3	本部门主任审核	P3	P4
1	T4	网络部主任审核	P4	P5
1	T5	自动运行同步器	P4||P5	P6
1	T6	网调中心配置地址	P2||P6	E

在系统运行一段时间后，申请私网 IP 地址的工单有 100 张，申请公网 IP 地址的工单有 20 张。该工作流网的弧记录了被走过的次数，如表 3 所列。在动态挖掘阶段，假设用户选择的粒度是 δ=90(最常用的)时，选择的主题是“申请 IP 地址”，生成的工作流网如图 6 所示。从中可以看出常用的 IP 地址申请流程很简单，无需复杂的审批过程。通过不同的置信粒度可以给用户不同的可见度，大多数用户只需关心与他相关的流程分支，这使得流程使用简单灵活。还可以做进一步的权限控制、粒度细分和角色挖掘，这将是我们下一步的研究工作。

表 3 动态弧集 DF

Wf_no	F-no	T	P	D	W
1	F1	T1	S	in	120
1	F2	T1	P1	out	120
1	F3	T2	P1	in	120
1	F4	T2	P3	out	100
1	F5	T2	P2	out	20
1	F6	T3	P2	in	20
1	F7	T4	P2	out	20
1	F8	T3	P4	out	20
1	F9	T4	P5	out	20
1	F10	T5	P4	in	20
1	F11	T5	P5	in	20
1	F12	T5	P6	out	20
1	F13	T6	P6	out	20
1	F14	T6	P3	out	100
1	F15	T6	E	out	120

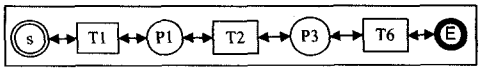

图 6 δ=90 情况下的流程模式图

结束语 本文从应用的角度出发，提出基于动态工作流网的动态工作流挖掘模型和算法，在很大程度上降低了企业内部建模的复杂性，提高了建模的柔性和挖掘的动态性，避免了流程挖掘在日志不完整、不能挖掘复杂结构和不适应柔性工作流等缺陷。但是在非自由选择结构中需要用户填写的规则还是不够人性化。提高系统的智能性，减少用户参与，是我们下一步要解决的问题。

参 考 文 献

[1] van der Aalst W M P, van Dongen B F. Workflow Mining: a Survey of Issues and Approaches[J]. Data and Knowledge Engineering, 2003, 47(2): 237-267

[2] Agrawal R, Gunopulos D, Leymarm F. Mining prooess models

from workflow logs[C] //Proceedings of the Sixth Intenational Conference Extending Database Technology. 1998:469-483

[3] vander Aalst W M P, Weijters A J M M, Maruster L. Workflow Mining:discovering process models from event logs[J]. IEEE Transactions on Knowledge and Data Engineering, 2002, 53(3): 245-264

[4] Herbst J, Karagiannis D. Workflow mining with InWoLve [J]. Computers in Industry, 2004, 53(3): 245-264

[5] Schim G. Mining exact models of concurrent workflow [J]. Computers in Industry, 2004, 53(3): 265-281

[6] Zhao Weidong, Dai Weihui. Role-Activity Diagrams Modeling Based on Workflow Mining [C]//2009 WRI World Congress on Computer Science and Information Engineering. Volume 4, 2009: 301-305

[7] She Jianchun, Yang Dongqing. Process Mining: Algorithm for S-Coverable Workflow Nets [C]//Second International Workshop on Knowledge Discovery and Data Mining(WKDD 2009). Jan. 2009: 239-244

[8] 李嘉菲，刘大有，于万钧. 过程挖掘中一种能发现重复任务的扩展 α 算法[J]. 计算机学报, 2007, 30(8): 1436-1441

[9] 吕静，陈未如，刘俊，等. 并发分支模式挖掘[J]. 计算机科学, 2004, 31(10): 270-276

[10] 陈翔，夏国平，李涛. 基于 Petri 网的工作流模型合理性研究[J]. 北京理工大学学报, 2004, 24(12): 1074-1078

[11] 袁崇义. Petri 网原理与应用[M]. 北京：电子工业出版社, 2005: 213-259

(上接第 165 页)

和 pingObj64 方法的参数则是不同长度的复杂参数；纵坐标表示远程服务方法调用的平均时间，单位为 μs。测试结果显示 StarOSGi 在简单类型数据调用上与 Apache CXF 的 D-OSGi 相比具有较明显的性能优势，这是由于 Apache CXF 的 D-OSGi 是采用 SOAP 协议来实现远程互操作，相对于 StarOSGi 采用 IIOP 协议，SOAP 协议带来了额外编码和 XML 解析开销。在复杂参数传递过程中，StarOSGi 与 Apache CXF 的 D-OSGi 性能相差不大。

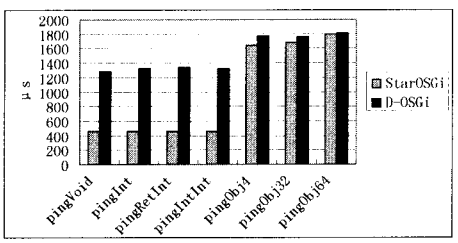

图 6 StarOSGi 与 D-OSGi 远程服务方法调用性能对比

结束语 OSGi 分布式扩展是 OSGi 在企业计算领域面临的重要问题。本文提出了一种基于 CORBA 的 OSGi 分布式扩展模型，基于该模型设计实现了一个支持 OSGi 分布式扩展的中间件 StarOSGi，与现有的 Newton，R-OSGi 以及 D-OSGi 相比，其特点在于能够保持 OSGi 面向服务的编程模型和轻量级特点，可以将集中式的 OSGi 应用透明地转变为分布式应用，并支持 OSGi 应用与 CORBA 应用的互操作，而且在性能上具有一定的优势。基于 StarOSGi 开发了体感鼠标等应用，并进行了对比测试，验证了 StarOSGi 的能力及有效性。下一步我们将对 StarOSGi 进行进一步的完善和改进，研究通用的服务发现协议，进一步扩大 OSGi 分布式扩展中服务发现的通用性。

参 考 文 献

[1] OSGi Alliance. OSGi Service Platform-Release 4[R]. 2005

[2] Gruber O, Hargrave B J, McAffer J, et al. The Eclipse 3.0 Platform: Adopting OSGi Technology[J]. IBM Systems Journal, 2005, 44(2)

[3] Sadtler C, Ganci J, Griffith K, et al. IBM Webshpere Product Overview. Redbook paper[R]. IBM, 2003

[4] Pratistha D, Nicoloudis N, Cuce S. A micro-services framework on mobile devices[C]// Conference on Web Services. Nevada, USA, 2003

[5] Fleury M, Reverbel F. The JBoss Extensible Server[C]// ACM/IFIP/USENIX International Middleware Conference. Riode Janeiro, Brazil, June 2003

[6] Walls C, Breidenbach R. Spring in action[M]. Manning Publications Co., Greenwich, CT, 2007

[7] OSGi Alliance. RFC 119 Specification[R/OL]. http://www.osgi.org/ Specifications/HomePage, 2009

[8] Paremus. The Newton Project[CP/OL]. http://newton.codecauldron.org, 2006

[9] Rellermeyer J S, Alonso G, Roscoe T. R-OSGi: Distributed Applications Through Software Modularization[C]// Proceedings of the ACM/IFIP/USENIX 8th International Middleware Conference. 2007

[10] Alliance A. Apache CXF Project[CP/OL]. http://cxf.apache.org/ distributed-osgi.html, 2009

[11] Object Management Group. The Common Object Request Broker: Architecture and Specification. Second Edition[R]. 1995

[12] Baker S. CORBA distributed objects: using Orbix [M]. New York: Addison-Wesley Publishing Co., 1997

[13] Wang Huai-min, Wang Yu-feng, Tang Yang-bin. StarBus+: Distributed Object Middleware Practice for Internet Computing[J]. 计算机科学技术学报：英文版, 2005(4)

[14] Marples D, Kriens P. The open services gateway initiative: An introductory overview [J]. IEEE Communications Magazine, 2001

[15] Sun Microsystems. Java JAR File Specification[R/OL]. http://java.sun.com /j2se/1.4.2/docs/guide/jar/jar.html, 2009

[16] Waldo J. The Jini architecture for network-centric computing [J]. Communications of the ACM, 1999, 42(7): 76-82

[17] Beisiegel M, Blohm H. SCA Service Component Architecture-Assembly Model Specification v. 1.0[R]. Open Service Oriented Architecture collaboration, Mar. 2007

[18] Veizades J, Guttman E, Perkins C. RFC 2165: Service Location Protocol[R]. IETF, 1997

[19] OSGi Alliance. RFC 126 Specification[R/OL]. http://www.osgi.org/ Specifications/HomePage, 2009

[20] Horstmann C S, Cornell G. Core Java 2, 7th Ed[M]. 2007(1): 132-145

[21] Philippsen M, Zenger M. Javaparty-transparent remote objects in Java[C]// ACM 1997 Workshop on Java for Science and Engineering Computation. June 1997