基于时间自动机的Web服务模型检测

骆翔宇[1,2] 轩爱成[1] 沙宗鲁[1]

(桂林电子科技大学计算机与控制学院 桂林 541004)[1] (清华大学软件学院 北京 100084)[2]

摘 要 传统的基于有限状态机的组合Web服务模型检测方法不能保证带有时间约束的组合Web服务的正确性。把组合Web服务看成多智能体系统,将带有时间约束的Web服务智能体建模为时间自动机,通过并发组合构成时间自动机网络,从而用时间自动机验证工具UPPAAL对组合Web服务的运行过程进行模拟,并验证其活性、安全性和死锁等性质。采用该方法对雇员出差安排组合Web服务进行建模和验证,结果表明,该组合Web服务存在死锁问题。最后通过分析死锁产生的路径,完善该组合Web服务的通信协议,从而消除了死锁。

关键词 模型检测,Web服务,时间自动机,UPPAAL

中图法分类号 TP393 **文献标识码** A

Model Checking Web Services Based on Timed Automata

LUO Xiang-yu[1,2] XUAN Ai-cheng[1] SHA Zong-lu[1]

(School of Computer and Control, Guilin University of Electronic Technology, Guilin 541004, China)[1]

(School of Software, Tsinghua University, Beijing 100084, China)[2]

Abstract The traditional model checking techniques based on finite state automaton cannot guarantee the correctness of Web services composition with timed constraints. We regarded Web services composition as multi-agent system. Each atomic Web service was modeled as timed automaton and by parallel composition of them, a network of timed automata was generated and inputted into the model checker UPPAAL. By using the proposed method and UPPAAL we were able to simulate the execution process and verify the liveness, safety and deadlock properties of a Web services composition. We took the atomic services of employee evection arrangements service as a case study of the proposed method and verified some related liveness and safety properties. A deadlock problem of the case study was found by simulation. By analyzing the execution path leading to the deadlock state, we found the reason and finally eliminated the deadlock by revising the communication protocol of the Web services composition.

Keywords Model checking, Web Services, Timed Automata, UPPAAL

1 引言

Web服务技术的广泛使用使得Web服务正逐步成为Internet网络环境中资源封装的标准形式。尽管Web服务技术给企业之间和企业内应用程序的集成带来了方便,但是单一的Web服务功能毕竟简单、有限,难以满足某些实际应用的需要,因此有必要对现有的单个Web服务进行组合,以生成功能更复杂的Web服务来支持各种应用需求。Web服务的这种组合方式使得其各原子服务间产生了大量复杂的信息交互,用来支持组合的Web服务正常运行。为了保障组合Web服务的安全性和活性,有必要对Web服务的运行进行检测。

时间自动机是一个带有有限时钟集合的有穷状态机,它提供了一种简单而有效的方法以描述带有时间因素的系统状态转换图,因此为实时系统的行为建模和性能分析提供了形式化方法,在实时系统的规范说明和模型验证中占据重要地位。本文用时间自动机对组合Web服务进行建模,并用时间自动机验证工具UPPAAL评估该模型的性质,验证其能否满足设计者的规范要求,检测其是否存在隐性的冲突问题。

本文第2节简单介绍了Web服务和Web服务组合;第3节介绍了时间自动机的模型、基于时间自动机的验证及时间自动机验证工具UPPAAL;第4节以雇员出差安排合成Web服务实例为例阐述了如何用时间自动机工具UPPAAL为合成Web服务建模,并验证了该组合Web服务的规范特性;最后在结束语中展望了未来的工作。

2 Web服务

Web服务是一种自包含的、模块化的应用程序,可在网络中执行描述、发布、查找以及调用等操作。由于Web服务是以XML为主的、开放的Web规范技术,因此其具有比任何

到稿日期:2009-09-16 返修日期:2009-12-21 本文受国家自然科学基金(60763004),中国博士后科学基金(20090450389),广西青年科学基金(桂科青0728090),广西研究生教育创新计划项目(2008105950812M424)资助。

骆翔宇(1974-),男,博士,副教授,硕士生导师,主要研究方向为模型检测、网络安全,E-mail:shiangyuluo@gmail.com;轩爱成(1982-),女,硕士生,主要研究方向为模型检测、Web Services;沙宗鲁(1981-),男,硕士生,主要研究方向为信息安全技术。

现有的对象技术更好的开放性，是建立可互操作的分布式应用程序的新平台。虽然 Web 服务技术为分布式应用提供了一个集成和交互机制，但是单一的 Web 服务功能简单有限，难以满足一些实际应用的需求，因此有必要对单个 Web 服务进行组合，以生成功能更复杂的 Web 服务来支持各种应用需求。

Web 服务组合[1]就是通过服务查找以及服务之间的接口集成，将多个自治的 Web 服务根据应用需求进行组合，从而提供新的、功能更强的 Web 服务，或者说提供一些增值的 Web 服务。一个组合的 Web 服务是将一些独立的、相互交互的 Web 服务进行聚集，并把所聚集的 Web 服务作为本身的组件来看待，从而获得比先前的 Web 服务更强大的功能。这种服务生成方式使得组合的 Web 服务中各原子服务间产生了大量复杂的消息交互，以支持组合过程的正常实现。由于各原子服务可能由不同的第三方开发商独立完成，并且它们之间的消息交互处于开放的网络环境，可能导致 Web 服务在组合过程中引入不可预见、不可察觉的错误，因此有必要对该组合的 Web 服务进行模型检测，从而用严谨的形式化验证方法来验证其安全性、活性和死锁等特性。

3 时间自动机

3.1 时间自动机模型

时间自动机(Timed Automata，TA)在传统的有限状态机的基础上添加时间的约束机制。自从有限状态机引入时间概念后，使得时间自动机这种模型具备了时间表达能力，从而成为描述时间系统的标准模型。时间自动机拥有有限多个控制位置和实数值时钟。两个控制位置之间可能存在迁移关系。两个控制位置之间发生迁移当且仅当与之相关联的时钟约束满足迁移关系。时间自动机的形式定义如下。

定义 1 一个时间自动机 TA 是一个带有时间约束、能够识别时间字的位置迁移系统，可以用一个六元组$(\Sigma, S, S_0, X, I, E)$表示[2,3]，其中

- Σ 是一个有限字符集合；
- S 是一个有限的位置集合；
- S_0 是一个起始位置集合；
- X 是一个有限时钟集合；
- I 是一个映射，它为 S 中的每一个位置 s 指定 $\Phi(X)$ 中的某一个时钟约束；
- $E \subseteq S \times \Sigma \times \Phi(X) \times 2^X \times S$ 是一个迁移关系集合。

假设$(s, a, \delta, \lambda, s')$是一个迁移关系，表示输入字符 a 时，从位置 s 到 s'的一个转移。δ 是定义在时钟集 X 上的一个时钟约束，在位置迁移发生时其必须被满足。λ 表示发生位置转移时被重置的所有时钟变量组成的集合，且满足 $\lambda \subseteq X$。

如果给定一个时间字$(\sigma, \tau) = (\sigma_1, \sigma_2, \cdots, \tau_1, \tau_2, \cdots)$，时间自动机 TA 在零时刻的所有迁移将位于起始位置，而且所有的时钟值均被初始化为 0，即所有的时钟均以同一速度开始计时，记录从起始位置开始所流逝的时间。在 τ_i 时刻，如果一个位置输入字符 σ_i，并且目前的时钟值满足从该位置出发的某一个位置迁移上的时钟约束 δ，就会发生从位置 s 到 s'的形如$(s, \sigma_i, \delta, \lambda, s')$的转移。在时间自动机中发生一次迁移，不仅会改变模型当前位置，也可能对迁移过程涉及到的某些时钟变量进行重置操作。时间自动机的这种行为能用它的一个运行来描述。

定义 2 时间自动机$(\Sigma, S, S_0, X, I, E)$所识别的时间字$(\sigma, \tau)$上的一个运行 r 是一个如下形式的无限序列[4]：

$$r: (s_0, v_0) \xrightarrow[\tau_1]{\sigma_1} (s_1, v_1) \xrightarrow[\tau_2]{\sigma_2} (s_2, v_2) \xrightarrow[\tau_3]{\sigma_3} \cdots$$

式中，$s_i \in S$ 并且对 $\forall i \geqslant 0$，v_i 表示从时钟集合 X 到 R^+ 的一个赋值，可以简记为 $v_i \in [x \rightarrow R^+]$，而且满足以下要求：

- 初始化：$s_0 \in S_0$，且对 $\forall x \in X, v_0(x) = 0$；
- 连续性：对 $\forall i \geqslant 1$，在迁移关系 E 中存在一个形如$(s_{i-1}, \sigma_i, \delta_i, \lambda_i, s_i)$的迁移，使得$(v_{i-1} + \tau_i - \tau_{i-1})$满足 δ_i，并且如果存在 $x(x \in \lambda_i)$，则 $v_i(x) = 0$；否则 $v_i(x) = v_{i-1}(x) + \tau_i - \tau_{i-1}$。

时间自动机可以识别时间字，即一个无穷字符序列，并且其中的每一个字符都伴随一个非负实数时钟值，以表示事件发生的时间。时间自动机提供了一种简单而有效的方法，描述带有时间因素的系统状态迁移图，因此为实时系统的行为建模和性能分析提供了形式化方法，在实时系统的规范说明和模型验证中占据重要地位。对于 Web 服务系统，由于大多数 Web 服务本身以及其中的消息传递具有时效性，因此用时间自动机对 Web 服务组合进行精确建模也具有重要意义。

3.2 基于时间自动机的验证

时间自动机在实时系统的规范验证中得到了越来越广泛的应用。利用时间自动机来验证一个具有有限状态位置的实时系统的正确性，可以通过判定相应的两个时间自动机所识别的时间正则语言是否相互包含来实现，也可以通过判定这两个时间正则语言的交集是否为空来实现。

判定一个时间正则语言是否为空的方法是将识别该语言的时间自动机转化为相应的 ω-自动机，比较常用的方法是利用文献[2]提出的构造区域自动机的算法将时间自动机模型转化为区域自动机。但在构建区域自动机的过程中，状态个数可能会随着时钟个数的增加呈指数级递增，且状态个数的增加与时钟约束中的常量大小成正比，因此为了消除状态空间组合爆炸问题，研究人员提出了对状态空间进行最小化的一些方法，这些方法可以比较好地减少状态个数。

近年来，随着对时间自动机研究的不断深入，利用时间自动机对实时系统进行自动规范验证有了一定的发展，特别是在通信协议自动化验证方面显示了极大的优越性。其中以时间自动机作为模型的自动验证工具 UPPAAL 可以有效地对实时系统进行建模、模拟和自动验证，以确保系统在实际运行中的高度正确性。

3.3 时间自动机验证工具 UPPAAL

自动验证工具 UPPAAL[5-7]主要采用一组带有整型变量的时间自动机对实时系统的行为进行模拟，对其性质进行验证。UPPAAL 使用的行为模型是由 R. Alur 和 Dill 提出的时间自动机模型[2]，每个时间自动机模型都有可能拥有一组整型变量和实数时钟变量，时间自动机之间可以通过共享的整型变量和通道进行通信。对传统的有限状态系统来说，系统的规范验证是通过检测系统的所有可达状态实现的，但时间自动机中的时钟变量的取值范围是 R^+，因此系统的状态个数是不可数的。所以在 UPPAAL 中适用限制技术来表示系统中称之为赋值集的时间区域，并且将系统的规范说明表示成一组系统全局状态，采用快速验证技术对其可达性进行分

析。

UPPAAL 的用户界面包括 3 个主要部分：系统编辑器、模拟器和验证器。系统编辑器用于创建和编辑要分析的系统。模拟器是一个确认工具，用于模拟运行检查所建系统模型可能的执行是否存在死锁，并保存死锁产生的路径，有助于设计者分析死锁产生的原因。验证器通过快速搜索系统的状态空间来检查时间约束和活性。在 UPPAAL 中用户可以定义一般值变量、全局时钟和用于同步的管道。UPPAAL 中使用规范的验证语言[8]，形式如表 1 所列（p 表示一个状态性质）。

表 1　UPPAAL 的规范语言

类型	性质	等价性质
Possibly	E⟨⟩p	
Invariantly	A[]p	not E⟨⟩not p
Potentially always	E[]p	
Eventually	A⟨⟩p	not E[] not p
Lead to	p→q	A[](p imply A⟨⟩q)

• $E\langle\rangle p$ 表示 Possibly。在时间自动机中，该公式为真当且仅当存在一个状态序列 $s_0 \rightarrow s_1 \rightarrow \cdots \rightarrow s_n$，使得 p 在状态 s_n 得以满足，其中 s_0 是起始状态。

• $A[]p$ 表示 Invariantly，等价于 not $E\langle\rangle$not p。

• $E[]p$ 表示 Potentially always。在时间自动机中，该公式为真当且仅当存在一个序列 $s_0 \rightarrow s_1 \rightarrow \cdots s_i \rightarrow \cdots$ 使得 p 在所有状态 s_i 中得以满足，并且这个序列是无穷的或者在状态 (l_n, v_n) 终止。

• $A\langle\rangle p$ 表示 Eventually，等价于 not $E[]$ not p。

• $p \rightarrow q$ 表示 Lead to，等价于 $A[](p$ imply $A\langle\rangle q)$。

4　实例研究

本节以一个雇员出差安排 Web 服务为实例，阐明如何用时间自动机对 Web 服务进行建模、模拟和验证。

雇员出差安排 Web 服务涉及到 4 个参与者：用户、出差服务、A 航空公司和 B 航空公司。首先，用户向出差服务发送一个出差请求（如雇员姓名、目的地、出发日期以及返回日期），出差服务开始检测符合用户要求的出差人员名单。如果没有员工符合用户要求，出差服务则返回请求失败的消息，否则就检测是否有航空公司可以提供服务。出差服务分别向 A 航空公司和 B 航空公司发送查询航班的请求，如果两家航空公司中至少有一家能提供服务，出差服务就向用户发送“请求成功”的消息，然后询问用户是否订机票。如果两家航空公司都能提供服务，出差服务将选择价格较低的一家。在订机票前，用户可以改变或取消请求。

4.1　系统建模

把用户、出差服务、A 航空公司和 B 航空公司 4 个 Web 服务视为 4 个智能体进行建模。这样，组合的 Web 服务就被看成了一个多智能体系统。一个多智能体系统由一个环境和若干个智能体组成。智能体可以通过自身的行为改变自身和环境的状态，也可以与其他智能体交互通信。这些智能体利用通道发送和接受消息，使得自己的状态得以改变，从而可以完成系统的运行过程。该实例的多智能体系统通信的过程可以用图 1 来表示。

根据图 1 的通信过程，用 UPPAAL 工具分别将 4 个智能体建模为 4 个时间自动机。图 2—图 5 中的“!”表示该智能体利用通道发送消息；“?”表示该智能体接收消息。

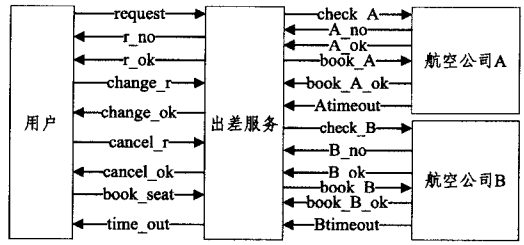

图 1　智能体间的交互通信框架

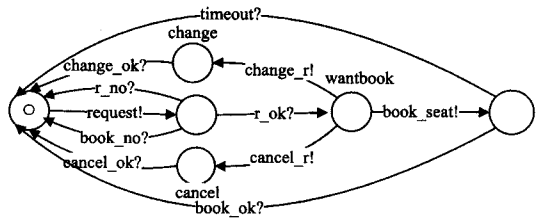

图 2　用户时间自动机

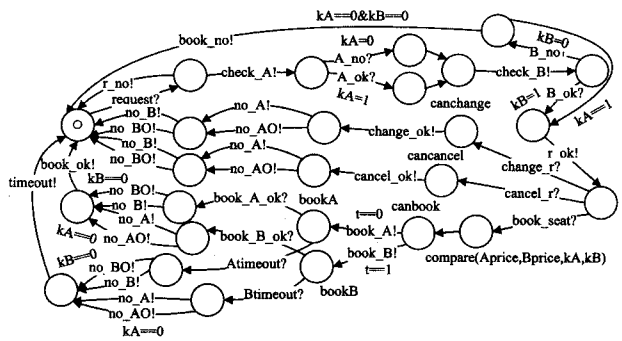

图 3　出差服务时间自动机

出差服务智能体接收到用户发送的“request”旅行请求后，首先检查自身能否提供服务。若不能提供服务，就向用户发送请求无效的消息“r_no”，否则分别向 A 航空公司和 B 航空公司发送消息，检查航班情况，然后再向用户返回其旅行请求是否可行。在这期间，用户可更改或取消旅行请求。

图 3 中的变量 kA，kB 表示航空公司是否能够提供服务。等于 0 时表示不能提供服务，等于 1 时表示可以提供服务。当 kA 和 kB 同时等于 0 时表示两家航空公司都不能提供服务，出差服务向用户返回请求失败的信息。函数 $compare()$ 表示当两家航空公司都能提供服务时从中选出花费较少的一家。

图 4 中 x 为 A 航空公司时钟。当 A 航空公司接到预订票的请求后，x 开始计时。如果在 12h 内出差服务没有完成相应的手续，该服务就会发送超时的消息，并取消订票。

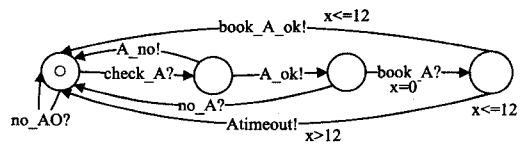

图 4　A 航空公司时间自动机

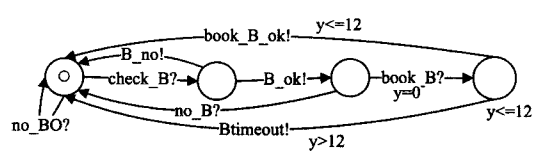

图 5　B 航空公司时间自动机

图5中 y 为B航空公司时钟。时钟 y 的功能和时钟 x 的功能相似。

4.2 模拟

通过模拟可以检测该系统运行过程是否符合设计者所期待的动作行为。模拟是沿着系统的演化，选择不同的迁移和延迟，并且在模拟中可以观察变量的值和所选择的使能的迁移过程。在模拟时，不但可以选择所要验证的迁移，即按照执行者的需求向下迁移，而且可以选择随机执行，即该系统随机地向下一步迁移。模拟的路径可以被保存下来，当该系统出现死锁时，用来分析死锁产生的原因。在雇员出差安排 Web 服务中通过模拟发现了死锁存在的路径，如图6所示。

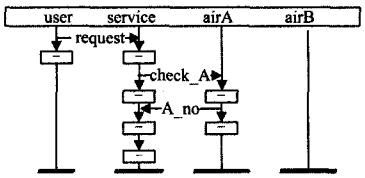

图6 死锁产生的路径

分析死锁产生的路径可知，当A航空公司和B航空公司同时可以提供服务时，它们相互竞争。如果出差服务代理选择了B航空公司提供服务，那么A航空公司一直等待。当用户再一次请求时，用户代理无法检测出A航空公司，所以出现了死锁。

为了避免死锁的产生，我们修改雇员出差安排 Web 服务的通信协议。当出差服务代理选择了A(B)航空公司后，出差代理再向B(A)航空公司发送取消预订的消息，使B(A)航空公司返回初始状态，如图3中发送的“no_B”消息等。4.1节中4个时间自动机为已经消除死锁的4个智能体对应的建模图。

4.3 验证

在验证前，必须明确该模型所满足的特性。本文将这些特性分为3类：安全性、活性和死锁。这些特性可用时态逻辑表述[8]，用模型检查工具 UPPAAL 验证[9]。

安全性：模型要满足设计者的某些安全限制。例如两个进程都需要进入临界区 cs，但安全性要求它们不能同时存在于临界区 cs 中。可用式(1)表示：

$$A[]\ \text{not}(P(1).cs\ \&\&\ P(2).cs) \tag{1}$$

该实例中安全性要求主要有如下几方面：

(1)当出差服务确定采用A航空公司提供服务时，B航空公司不能再提供服务。用公式表示如下：

$$A[]\ service.bookA\ \text{imply not}\ service.bookB \tag{2}$$

式(2)的验证结果为 true。

(2)在购票前出差服务智能体必须允许用户修改其请求内容。用公式表示如下：

$$A[]\ user.\text{change imply}\ service.\text{canchange} \tag{3}$$

式(3)验证结果为 true。

(3)在购票前出差服务智能体必须允许用户撤销其请求。用公式表示如下：

$$A[]\ user.\text{cancel imply}\ service.\text{cancancel} \tag{4}$$

式(4)的验证结果为 true。

(4)假设当A航空公司提供服务时，如果出差服务要订票，则它必须在A航空公司规定的时间内发出请求。用公式表示如下：

$$A[]\ service.bookA\ \text{imply}\ airA.x<=12 \tag{5}$$

式(5)的验证结果为 true。同理可验证B航空公司的类似性质。

活性：检测模型是否按照正确的顺序迁移。如在进程互斥进入临界区的实例中，当有进程处于等待状态，其就有机会进入临界区。可用式(6)表示：

$$A()(P(1).\text{wait imply}\ P(1).cs) \tag{6}$$

在该实例中对活性的要求主要有：

(1) 假若A航空公司提供服务，并且在其规定的时间内用户预订机票，出差服务智能体必须可以订票。用公式表示如下：

$$A()((user.\text{wantbook and}\ airA.x<10)\ \text{imply}\ service.\text{canbook}) \tag{7}$$

式(7)验证结果为 true。

(2) 如果A航空公司和B航空公司都可以提供服务，但是A航空公司提供服务的花费少于B航空公司，那么出差服务最终应该选择A航空公司。公式表示如下：

$$A()((kA==1\ \&\&\ kB==1\ \&\&\ Aprice<Bprice)\ \text{imply}\ service.bookA) \tag{8}$$

式(8)的验证结果为 true。

死锁：用式(9)可以检测该模型中是否存在死锁现象。

$$A[]\ \text{not deadlock} \tag{9}$$

式(9)的结果为 true，说明该模型中不存在死锁。

通过验证发现，修改后的组合 Web 服务不存在冲突，且符合设计者的要求。

结束语 本文介绍了以时间自动机形式模型为基础的组合 Web 服务正确性验证的模型检测方法。以一个合成 Web 服务为例，介绍了利用时间自动机模型检测工具 UPPAAL 对其进行建模，模拟并验证了该模型的安全性、活性和死锁等问题，发现该合成服务中存在死锁问题。通过分析死锁产生的路径，完善服务通信的协议，消除了死锁。该实例验证表明了该方法的有效性。今后的研究工作将致力于设计并实现从 Web 服务的描述语言到时间自动机网络的自动转换算法，提供一整套关于带有时间约束的组合 Web 服务的建模与验证方法学，从而促进形式化验证方法应用于实际 Web 服务系统的开发过程中，保证所开发系统符合设计需求。

参 考 文 献

[1] Lallali M, Zaidi F, Cavalli A, et al. Automatic Timed Test Case Generation for Web Services Composition[C]// Proc. of the 6th IEEE European Conference on Web Services. United States: IEEE Press, 2008: 53-62

[2] Alur R, Dill D L. A Theory of Timed Automata[J]. Theoretical Computer Science, 1994, 126: 183-235

[3] Alur R. Timed Automata[C]// Proc. of 11th International Conference on Computer-Aided Verification (CAV'99). Berlin: Springer-Verlag, 1999, 1633: 8-22

[4] Alur R, Courcoubetis C, Dill D. Verifying Automata Specification of Probabilistic Real-time Systems[C]// Proc. of REX Workshop "Real-time: Theory in Practice". Berlin: Springer-Verlag, 1991, 600: 28-44

(下转第197页)

在数据空间中的位置，分配到相应的聚类结果簇中，计算该数据集中数据点分配正确的比例，此比例则为在指定时间窗口 h 上 RDFCluStream 算法的聚类精度。

同样，可以定义 CluStream 算法的精度如下：在指定时间窗口 h 上，把带有预先类标识的数据集按数据点在第一阶段聚类获得的微簇 id 号，分配到相应的聚类结果簇中，计算该数据集中数据点分配正确的比例。此比例则为在指定时间窗口 h 上 CluStream 算法的聚类精度。

图 1 所示为 KDD CUP99 数据集 corrected 在流速固定为 1000/s 条记录、时间窗口 h=60s、微簇数目为 1024 块时在数据流不同查询时点上的聚类质量的比较情况。由于 KDD CUP99 数据集 corrected 中的网络连接类型分布严重不均匀，从图 1 可以看出，能很好地支持任意形状聚类的 RDFCluStream 算法在精度上明显优于 CluStream 算法。

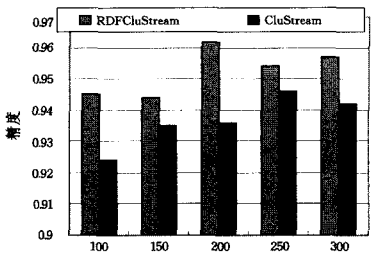

图 1　数据流不同查询点上的聚类精度比较(1000 条/s，h=60s)

4.3　算法处理时间

对于数据流处理算法来说，在线处理速度的高低是至关重要的。质量再高的处理算法，若处理速度无法达到数据流的峰值速度，也是无法使用的。为此，这里重点比较分析算法 *RDFCluStream* 和 *CluStream* 的在线处理速度。

RDFCluStream 算法由于对数据空间采用了网格化处理，每当新数据点到达，则可直接定位其所属网格单元，时间复杂度为常数 1。而 CluStream 算法则要通过计算新数据对象到各子聚类中心的距离，并比较之后才能决定其所属的子聚类，其时间复杂度为 $O(k)$，其中 k 为子聚类的数量。

在处理微簇聚合时，RDFCluStream 算法判断两个微簇是否可以合并，主要看它们是否相邻，即两者之间是否存在一个公共的超平面，只须进行一次减法运算，时间复杂度为常数 1。而 CluStream 算法判断两个微簇是否可以合并，则要计算各微簇中心之间的距离，并进行比较之后才能决定，其时间复杂度为 $O(k)$。

从上面的分析可知，RDFCluStream 算法的处理时间不随数据流的大小、微簇数目多少而明显改变，能很好地完成连续数据流环境下的在线聚类任务。相比可知，CluStream 算法的速度虽然也可以不随数据流的大小而改变，但随微簇的数量增多而明显下降。由于在线处理阶段的微簇数目多少直接影响到离线阶段的最终聚类质量，因此算法 RDFCluStream 相比 CluStream 而言，能更好地支持数据流的高质量聚类。

图 2 所示是算法 RDFCluStream 与 CluStream 在不同微簇数目设置下的运行时间比较。图 2 的实验结果表明，RDFCluStream 算法的速度总体上明显高于 CluStream 算法。只是在微簇数目与最终结果类数目相近时，RDFCluStream 算法因微簇之间的合并与分裂操作十分频繁，使得速度有所下降，而 CluStream 算法的速度却达到最高，不过这时因微簇数目过少，使得最终聚类质量很差，难以使用。

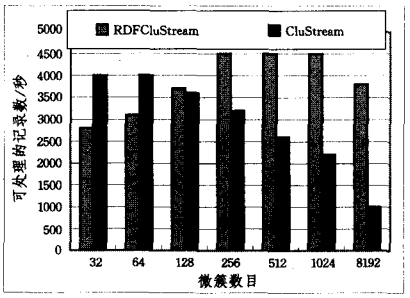

图 2　不同微簇数目时的在线处理速度比较

结束语　基于相对密度的数据流模糊聚类算法利用微簇的空间位置重叠关系，定义了微簇集合间的差运算，以支持用户指定时间窗口内的数据流聚类要求。同时，此算法在借鉴 CluStream 算法的两阶段聚类思想和 pyramid time frame 的快照储存结构的基础上，结合相对密度聚类[8]和模糊聚类[7]的优点，克服了 CluStream 算法的固有缺陷，能形成任意形状、多密度分辨率的层次聚类结果。

参 考 文 献

[1] Aggarwal C, Han J, Wang J, et al. A framework for clustering evolving data streams[C]// Proc. of VLDB. 2003:81-92

[2] Aggarwal C, Han J, Wang J, et al. A framework for projected clustering of high dimensional data streams[C]// Proc. 2004 Int. Conf. Very Large Data Bases (VLDB'04). Toronto, Canada, 2004, 8:852-863

[3] Cao F, Martin E, Qian W, et al. Density-based Clustering over an Evolving Data Stream with Noise[C]// Proc. of the 2006 SIAM Conference on Data Mining (SDM'2006). 2006

[4] Liu Qing-Bao, Deng Su, Lu Chang-Hui, et al. Relative density based K-nearest neighbors clustering algorithm[C]// Proc. 2003 Int. Conf. on Machine Learning and Cybernetics. 2003:133-137

[5] 朱蔚恒，印鉴，谢益煌. 基于数据流的任意形状聚类算法[J]. 软件学报，2006，3：379-387

[6] 刘青宝，戴超凡，邓苏，等. 基于网格的数据流聚类算法[J]. 计算机科学，2007(3)：159-162

[7] 孙即祥，等. 现代模式识别[M]. 长沙：国防科技大学出版社，2002

[8] 刘青宝，何勇，邓苏，等. 基于相对密度的多分辨率聚类算法[J]. 小型微型计算机系统，2007(6)：1287-1292

[9] 刘青宝，金燕，邓苏，等. 基于模糊聚类的属性匹配算法[J]. 模糊系统与数学，2006，20(6)：96-102

(上接第 142 页)

[5] Diaz G, Cuartero F, Valero V, et al. Automatic Verification of the TLS Handshake Protocol[C]// Proc. of the 2004 ACM Symposium on Applied Computing. 2004, 1:789-794

[6] Diaz G, Larsen K G, Pardo J, et al. An Approach to Handle Real Time and Probabilistic Behaviors in E-commerce: Validating the SET Protocol[C]// Proc. of the 20th Annual ACM Symposium on Applied Computing. 2005, 1:815-820

[7] Larsen K, Pettersson P, Wang Yi. UPPAAL in a Nutshell[J]. International Journal on Software Tools for Technology Transfer, 1997:134-152

[8] Larsen K G, Petterson P, Wang Yi. Diagnostic Model-checking for Real-time Systems[C]// Proc. of Workshop on Verification and Control of Hybrid Systems III. 1995, 1066:575-586

[9] Diaz G, Cambronero M E, Pardo J J, et al. Model Checking Techniques Applied to the Design of Web Services[J]. CLEI Electronic Journal, 2007, 10