# 证书系统缓存替换算法的研究*)

佘 堃 牛新征 周明天

(电子科技大学计算机科学与工程学院 卫士通信息安全实验室 成都610054)

**摘 要** Cache技术是一种能够减少时延,节省带宽和降低服务器负载的有效方法。文章分析了现有证书服务器中Cache算法和证书系统的结构特性;指出证书缓存的简单替换算法:FIFO(first in first out)和LRU(Least Recently Used),无法提供较高的Cache命中率。文中提出了一种智能预留控制缓存替换算法,理论分析和仿真数据表明该算法能大大减少访问证书的时间,改善证书管理效率,并有效提高缓存的命中率。

**关键词** 证书,证书数据库,预留,Cache,优先率,替换

## Research on Cache Replacement Policy for Certificate System

SHE Kun NIU Xin-Zheng ZHOU Ming-Tian

(School of Computer Sci. and Engineering, University of Electronic Sci & Tech of China,
Information Secure United Lab of UESTC-Westone, Chengdu 610054)

**Abstract** Cache technology is one of the effective ways to reduce time delay, save bandwidth and lower server's load. In this paper, we analyze the characteristics of certificate system, and the Cache algorithms in current certificate servers. Current simple Cache replacement algorithms, such as FIFO (first in first out) and LRU (Least Recently Used) are not suitable for parallel scientific application because of their low hit ratio. We propose a new intellectualized Cache replacement algorithm. The theory analysis and simulation results show that the new algorithm for cache replacement not only reduces the time of processing certificates, but also improves efficiency of certificate management obviously and higher the hit ratio.

**Keywords** Cache, Certificate server, Certificate, Priority rate, Replacement, Prefetch

## 1 引言

证书访问处理是实现公钥基础设施中安全认证和授权的重要一环[1]。证书的大规模使用带来了一个问题:如何迅速、有效地实时响应用户对证书的访问,快捷高效地取出需要的证书。针对目前已有的简单证书缓存Cache替换算法FIFO和LRU[15]不能适应大规模证书应用系统的需求,本文提出了一种新的证书库替换算法,并通过仿真验证其有效性。

证书存取工作的模型可用图1表示。

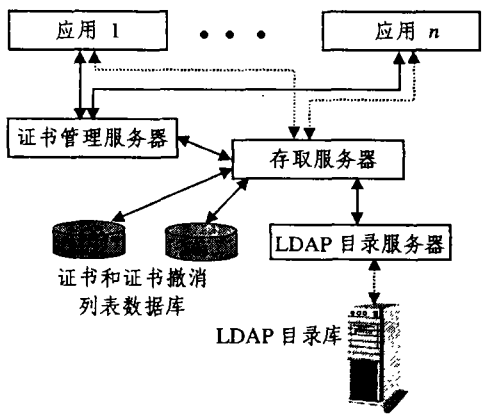


图1 证书存取系统框架图

由图1可见无论是本地的证书和证书撤消列表数据库,还是远方的LDAP目录库,对证书查询访问,都需要通过Cache来加快查询访问速度,所以必须使用较好的替换策略。

目前证书系统中使用较多的Cache替换算法是:FIFO和LRU,由于证书库要考虑诸如历史信息及证书的使用特性等,上述Cache替换策略难以适应。

## 2 智能预留控制替换算法

根据证书库的访问特性和实际情况,实践证明当证书库相对稳定后,zipf法则[9~11]可以很好地逼近在某个时刻证书库中证书的实际访问概率分布。

**定义1** 假设证书库中共有X个证书,将证书的访问概率按降序排列,排序后符合zipf法则,第$k$个证书的访问概率表示为$p_k$,则证书库中证书的实际访问概率可以表示为

$$s=\{p_k \mid p_k \text{ 的访问概率为 } f_k=\frac{h}{k^b},\ h=\frac{1}{\sum_{i=1}^{x}\frac{1}{i^b}},\ k=1,2\cdots\cdots x\}$$

其中b为待定常量,与具体的证书系统有关。

### 2.1 定义智能预留控制替换算法

假设为存取服务器的Cache预留出了n个空闲的缓冲块,N为同时到达的Cache替换请求个数,Cache共有m个块。

步骤1.在Cache中查找用户请求的证书,如找到需要的证书,则响应请求,如未找到,进入步骤2。

步骤2.在后台数据库进行查找,如果找到证书则响应请

*)基金项目:国家863宽带VPN项目863-104-03-01课题资助。佘 堃 在职博士生,主要研究方向为网络计算,信息安全,软件工程。牛新征 硕士研究生,主要研究方向为网络计算,分布式计算,并行计算,移动计算,信息安全。周明天 博士生导师,主要研究方向为网络计算,信息安全,分布式计算,并行计算,移动计算。

求，并把找到的证书放入 Cache，进入步骤3进行替换处理；如果没有找到证书，给出未找到证书的报告。

步骤3. 假设需要进行替换的证书数为 N，如果预留出来的 n 个空闲块数目大于等于 N，进入步骤4处理，否则进入步骤5处理。

步骤4. 从 Cache 中分配 N 个空闲块给用户，及时响应用户的替换要求；并同时采用2.2节所述的替换模型的数学方法，对 $m-n$ 个 Cache 块计算优先率，求出 N 个优先率最小的 Cache 块清除内容，并把这 N 个空出的块划为预留块，恢复 n 个预留块的初始状态，算法结束。

步骤5. 从 Cache 中分配 n 个空闲块给用户，及时响应用户的替换要求，拒绝后继的 N－n 个的替换请求，同时采用2.2节所述的替换模型的数学方法，求出 n 个优先率最小的 Cache 块清除内容，恢复到 n 个预留块的初始状态，跳回步骤3进行判断处理。

从上面的描述可知，n 既不能太大也不能太小，太大就浪费了 Cache 资源，太小就不能完全满足对用户的实时访问操作的要求了。

下面使用“代价”的概念来衡量系统的性能，由此求出 n 的大小。

**定义2** 代价为对系统资源使用的开销和用户等待延迟花费的开销。

因为在系统稳定后某个时刻，Cache 中存放的证书是最常用的，所以当采用预留控制后剩余的 m－n 块的概率，就是证书库中按证书的访问概率降序排列后的前 m－n 个块的概率。最小访问概率的块的概率可记为 $p_{m-n}=h/(m-n)^b$。

通过文[5]的代价算法思路，进行重要改进，可以求得当剩余的 m－n 个 Cache 块的访问概率和都大于概率阈值 U 时，命中率最大，以此可求出 n 最合适的大小。

**定义3** 设 λ＝用户请求的到达率；$\lambda_1$＝没有命中 Cache 块时，对用户请求的响应率；$\lambda_2$＝命中 Cache 块时，对用户请求的响应率。

**定义4** 设 p 为 Cache 块的利用率，即剩余 m－n 个 Cache 块的访问概率和，可表示为：

$$p=\sum_{i=1}^{m-n} p_i \tag{1}$$

先记为 p(方便后续计算)

因为通过 Cache 块满足的用户请求率为 $\lambda-\lambda_1$，而 Cache 块的利用率，即被使用的概率为 p，则前式也可写作 $p\lambda_2$，即 $\lambda_1+\lambda_2=\lambda+(1-p)\lambda_2$ 或

$$\lambda_1+p\lambda_2=\lambda \tag{2}$$

设在等时间片分时共享处理机系统中，平均处理时间为 $i$ 单位的请求响应时间 $t_i$ 为

$$t_i=\frac{i}{1-\alpha}=\frac{s}{\omega(1-\alpha)} \tag{3}$$

$$\alpha=\frac{s(\lambda_1+\lambda_2)}{\omega} \tag{4}$$

其中 $\alpha$ 为系统负载，s 为每个被请求访问的证书的平均大小，$\omega$ 为系统处理速度。

命中 Cache 块的请求处理开销为：

$c_2=w\times s$

没有命中 Cache 块的请求处理开销为：

$c_1=w\times s+z\times t_i$

其中 $w$ 表示证书每字节的处理开销。

另：$w(i)=c_i/s_i$[6]，其中 $c_i$：证书 $i$ 的处理开销，$s_i$：证书 $i$ 的大小，这里把所有证书每字节的处理开销看作相同大小。

$z$：每秒等待时间的开销。

最终的平均处理代价为：

$$c=\frac{\lambda_1\times c_1+\lambda_2\times c_2}{\lambda}=\frac{s}{\lambda}\left\{[\lambda+(1-p)\lambda_2]w+\frac{(\lambda-p\lambda_2)z}{\omega-[\lambda+(1-p)\lambda_2]s}\right\} \tag{5}$$

将上面的 $p$(1)式代入(5)式可求得成本代价函数为：

$$c=\frac{s}{\lambda}\left\{[\lambda+(1-\sum_{i=1}^{m-n}p_i)\lambda_2]w+\frac{(\lambda-\sum_{i=1}^{m-n}p_i\lambda_2)z}{\omega-[\lambda+(1-\sum_{i=1}^{m-n}p_i)\lambda_2]s}\right\} \tag{6}$$

假定 p 和 λ 的值已知，为使系统的平均请求代价最小，确定 $\lambda_2$ 进而以此确定 p 的大小，对式(6)求 C 对 $\lambda_2$ 的二阶导数得

$$\frac{d_2c}{d\lambda_2^2}=\frac{2s^2}{\lambda}[z(\lambda s-\omega p)(1-p)/(\omega-s(\lambda+\lambda_2(1-p)))^3] \tag{7}$$

其中：未把 $p$ 的具体数值代入。

因为系统稳定的前提是 $(\lambda_1+\lambda_2)s<\omega$，即 $s(\lambda+(1-p)\lambda_2)<\omega$

因此，当 $\lambda s<\omega p(0<p<1)$ 时，式(7)小于0 可知道曲线是上凸的，可以求代价 $c$ 的最大值。

即式(7)在 $\frac{dc}{d\lambda_2}$ 为0时能求得最大值。此时 $\lambda_2$ 取 $\lambda_2'$ 为

$$\lambda_2'=\frac{1}{s(1-p)}\left(\omega-\lambda s-\sqrt{\frac{z(\omega p-\lambda s)}{w(1-p)}}\right) \tag{8}$$

即当 $\lambda_2>\lambda_2'$ 时，代价函数随 $\lambda_2$ 的增大而减小，特别地，当 $\lambda_2'\leqslant 0$ 时，对于给定的 p，λ，当 $\lambda_2$ 在 $[0,\lambda/p]$ 范围内增大时，代价函数减小，也就是说此时若 Cache 块利用率大于等于 p，代价就最低，命中率最大。

由此，可求出 p 的阈值，从式(8)中得到，$\lambda_2'\leqslant 0$，当且仅当

$$p=\sum_{i=1}^{m-n}p_i\geqslant 1-\frac{(1-\alpha)z/w}{(1-\alpha)^2\omega+z/w}=1-\frac{(1-\alpha)z}{(1-\alpha)^2w\omega+z} \tag{9}$$

则可以求得 p 的阈值 U 为：

$$U=1-\frac{(1-\alpha)z}{(1-\alpha)^2w\omega+z}$$

式(9)表明，若利用率 p 大于等于 U，即在 $\lambda_2'\leqslant 0$ 时，$\lambda_2$ 增大，代价函数减小。依前面分析，此时命中剩余的 m－n 个 Cache 块的可能性最高，代价最小。

把访问概率公式 $p_i$ 代入式(9)，可得：

$$\sum_{i=1}^{m-n}p_i\geqslant U\Leftrightarrow\sum_{i=1}^{m-n}\frac{h}{i^b}\geqslant 1-\frac{(1-\alpha)z}{(1-\alpha)^2w\omega+z}$$

其中 $h=\frac{1}{\sum_{j=1}^{x}\frac{1}{j^b}}$

即 n 的大小可以根据实际的系统，用试探法通过上式可求出。

### 2.2 替换模型

下面是证书库的替换模型数学方法：

设 Cache 中共有 m 个 Cache 块，讨论其中的某个 Cache 块，该 Cache 最后一次被访问到的时刻为 $t_1$，用户对它进行再次访问的时刻为 $t_2$，设 ε(随机变量)为从空闲到用户使用所经过的时间间隔，并记任一个空闲时间片大于 t 的概率为 w(t)＝p{ε>t}，w(t)为一个可靠度分布函数，记 $\tau$ 为单位时间该块被访问的次数，则 $\tau$ 越大，w(t)越小。

1. Cache 块从空闲到被用户使用之间时间的分布为可靠度分布函数 w(t)[3]

$$w(t)=\begin{cases}\dfrac{1}{e^{\tau}(t+1)}, & t\geqslant 0\\ 1, & t<0\end{cases}$$

2. 累积分布函数 Y(t)[4]为随机变量 ε 的概率分布函数

$$Y(t)=1-w(t)=\begin{cases}1-\dfrac{1}{e^{\pi(t+1)}}, & t\geqslant 0\\ 0, & t<0\end{cases}$$

3. 优先率 q(t)为在 Cache 块空闲 t 时间之后，它在$(t,t+\Delta t]$时间内被使用的概率

$$q(t)=\frac{y(t)}{1-Y(t)}$$

其中 y(t)是 Y(t)的概率密度。

显然，优先率越高，被优先使用的可能性越大，且

$$q(t)=\frac{\dfrac{dY(t)}{dt}}{w(t)}=\frac{-w'(t)}{w(t)}=-\frac{(e^{-\pi(t+1)^{-1}})'}{e^{-\pi(t+1)^{-1}}}=\tau+(t+1)^{-1}$$

$$=\tau+\frac{1}{t+1} \tag{10}$$

从该式看出，q(t)考虑了以前的历史因素，使用了条件概率的概念，从下式(12)可看出，它与几个时间因素和历史使用次数有关。

在 $t_2$时刻，从式(10)，计算优先率有

$$q(t_2-t_1)=\tau+\frac{1}{t_2-t_1+1} \tag{11}$$

在实际的情况中，记 $c_a$ 为该块被访问的总次数，$t_0$为该块被装入 Cache 的时刻，则 $\tau\approx c_a/t_2-t_0$。把 $\tau$ 代入式(11)有

$$q(t_2-t_1)\approx\frac{c_a}{t_2-t_0}+\frac{1}{t_2-t_1+1} \tag{12}$$

式(12)为最终的优先率公式。其中：Cache 中共有 m 个 Cache 块，而 m 等于系统和用户的应用建立会话时，分配给用户应用的 Cache 块的大小，证书库中有 X 个证书(某个时刻)，$\alpha$ 为系统负载，$\omega$ 为系统处理速度，z 为每秒等待时间的开销，w 为每字节证书的处理开销，b 为根据不同证书系统的待定常量。

## 3 仿真

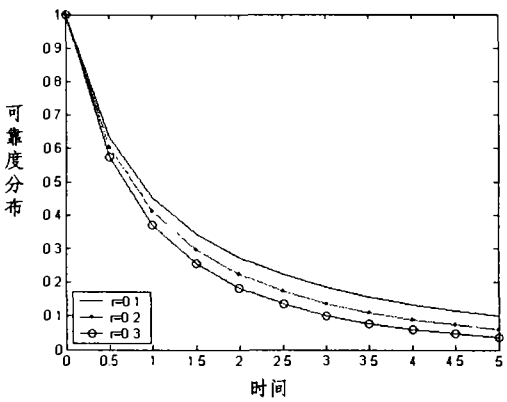


图2 时间与可靠度分布的关系(r 即为 τ)

图2～图6给出了在 zipf 法则分布中 b=1.2(根据实际证书库概率分布可得出)，m 大小为1000，我们讨论的为百万级的证书库，证书服务器同时建立的会话端口有1000个(系统信道容量即上面的系统处理速度)的情况下，优先率和 n 的个数的曲线图。

从图3我们可以看出在同一时间间隔上，当 r 越大，优先率越高，r 为 $c_a/(t_2-t_0)$；当 r 相同时，间隔越长，优先率越低；从图5可以看出当负载很小时，随着负载增大，概率阈值减小，即可知为了满足式(9)和使 p 尽量小(既能保证命中率，又可以充分发挥预留控制的优势)，要使 m－n 数目减小，即 n 要增大。同时当 n 增大时，负载增大，所以我们可以在某个允许的负载范围内，求出 n 最合适的大小，既可以让访问 Cache 时，命中率最高，又可以发挥预留的最大优势，及时响应用户替换请求。图6即描述了在不同的负载下，n 的大小。所以这些表现与理论分析是一致的，可以为我们的实际系统应用提供参考。

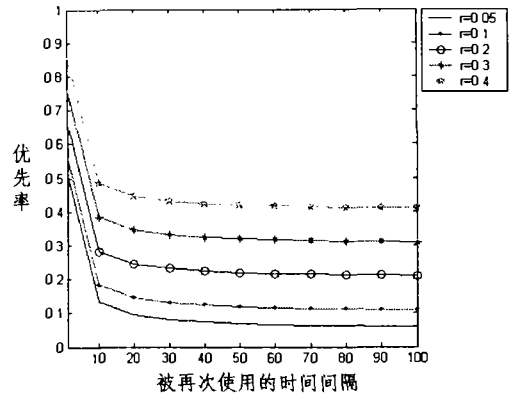


图3 被再次使用时间间隔与优先率的关系

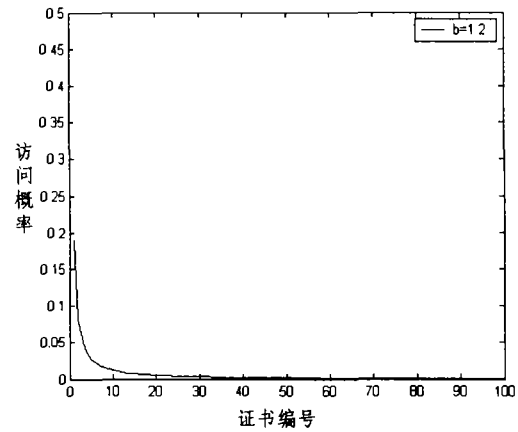


图4 证书编号与访问概率的关系

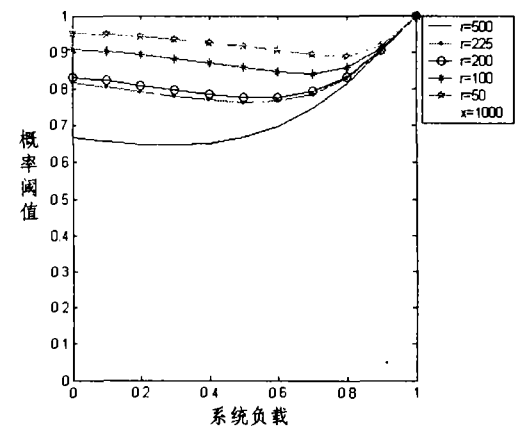


图5 系统负载和概率阈值的关系(r=z/w)

为了验证智能预留控制替换算法的有效性，根据文[14]的方法，开发了一个事件驱动的模拟模型，访问序列由一个随机数发生器产生，随机数的概率分布符合 zipf 法则，可以从实验中证明该替换算法的命中率最高。使用其余两种任意且武断的方法，对于较大的证书库，证书系统很不稳定，替换较多，并且造成了证书存取系统的瘫痪。

并且由于当 n 值确定后，优先率公式(12)时间复杂度为

O(1),FIFO,LRU 的时间复杂度也为 O(1),且空间复杂度也相同,可见我们提出的算法复杂性没有增加。

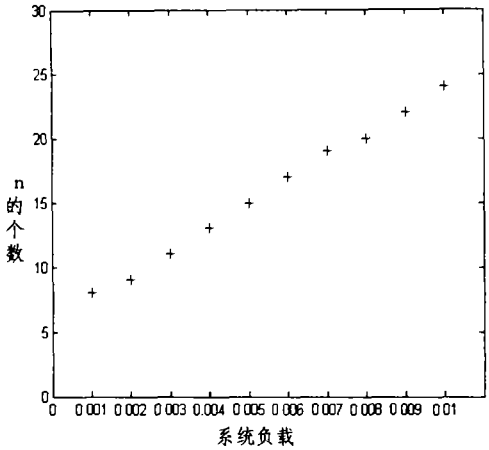


图6 系统负载与 n 的个数的关系

**结论** 证书访问处理是实现 Internet 公钥基础设施中安全认证和授权的重要一环。而证书服务器中的现有方法并没有解决好 Cache 的置换问题,本文提出了一种新的方法,该方法能有效地缩短网络对用户的实时性响应的时间,最大程度也节约了系统资源,改变了以前被动的替换策略,使替换更加合理迅速。

## 参 考 文 献


1 Housley R, Ford W, Polk W, et al. Internet X.509 Public Key Infrastructure Certificate and CRL Profile. RFC2459, 1999-01
2 Aggarwal C, et al. Caching on the world wide web [J]. IEEE Transaction on knowledge and data engineering, 1999.11
3 Nelson N. Accelerated life testing-step-stress models and data analysis[J]. IEEE Transactions of Reliability, 1980, R-29(2)
4 Nishida S. Failure Analysis in Engineering Applications [M]. Oxford: Butterworth-Heinemann Ltd. 1992. 22～2
5 Zhimei J. An adaptive network prefetch scheme [A]. In: IEEE Intl. Conf. on Communications [C] Part 1, 1997. 8～12
6 Hyokyung B, Kern K, Sam H N, Sang L. Efficient Replacement of Nonuniform Objects In Web Caches. IEEE Journal of Computer Science, 2002, 6: 65～73
7 Tanenbaum A S. Computer Nerwork [M]. Third Edition Prentice Hall Internation, Inc
8 Frankel Y, Gemmell P, MacKenzie P D, et al. Optimal-resilience proactive public-key cryptosystems. In: IEEE symposium on Foundations of Computer Science, 1997. 384～393
9 Nussbaumer J P, et al. Networking requirement for interactive video on demand. IEEE Journal of Selected Areas in Communication, 1995, 13(5): 779～787
10 Zipf G K. Human Behavior and the Principle of Least Effort. Addison-Wesley, 1949
11 Wolf J, et al. Disk load balancing for Video-on-Demand systems. Multimedia system, 1997, 5(6): 358～370
12 茆诗松, 王玲玲. 可靠性统计[M]. 上海: 华东师范大学出版社, 1984
13 唐三平, 等. 一种处理服务器连接的控制模型. 计算机研究与发展, 2002, 39(6): 668～671
14 李勇, 等. 大规模 VOD 及其 Cache 机制. 计算机工程与科学, 1999, 21(5): 52～55
15 Access Control Library (ACL) Library File, Getronics Government Solutions. http://www.getronicsgov.com/hot/acl_home.htm


(上接第60页)

CL 语言在工作流模型中有以下两种应用:

(1)将 CL 语言的逻辑表达式计算功能应用于起始条件、中止条件、转移条件的逻辑值计算中。类 CXT_Wf_Condition 中的条件可以是用 CL 语言写成的复杂逻辑表达式,当需要计算此表达式的值时只需调用 CL 语言的解释执行程序。

(2)CL 语言能对变量进行定义和赋值,并且能对对象的方法进行调用,这样用 CL 语言写成的数据处理程序就能方便地被解释执行,使得工作流模型中的数据流处理变得更为容易。

由于 CL 语言的应用,工作流模型可以处理非常复杂的各种逻辑条件以及数据映射,变得更为灵活。

## 7 数据模型管理

数据模型实现了对元数据的管理与动态的数据总线,应用实例在运行状态下可以依据事务的处理状态动态拼装数据。我们采用了 Composite 模式[5]来设计数据模型,类图如图3所示。

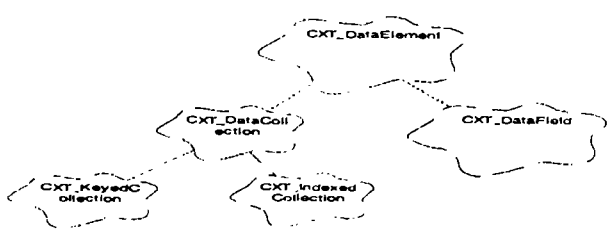


图3

1)CXT_DataElement 是最基本的基类,简单数据类 CXT_DataField 和复杂数据类 CXT_DataCollection 都由它派出来。它提供了数据操作的统一接口。

2)类 CXT_DataField 代表最基本类型的数据,如整型、浮点型、字符串型等。

3)类 CXT_DataCollection 代表复合型的数据,可衍生出 CXT_KeyedCollecion 和 CXT_IndexedCollection 两类复合型数据。

4)类 CXT_KeyedCollection 由有序的 CXT_DataElement集合组成,各个 CXT_DataElement 可以是不同类型的数据,类似于 C 语言中的 struct 结构。

5)类 CXT_IndexedCollection 是由相同的 CXT_DataElement 复合而成,相当于 C 语言中的数组,各个 CXT_DataElement 子项以下标标识。

在已定义的数据字典的基础上,工作流模型中的活动可以用这些表示数据的类动态地拼装出所需要的数据结构并赋予相应的数据,极大地增加了数据管理的灵活性和可用性。

**结论** 本文着重讨论了面对企业应用集成的工作流引擎的设计和实现,以及如何通过对 Script 语言的解释执行和灵活的数据管理来提高工作流模型的复杂程度和表述能力。对工作流模型的描述标准的支持和工作流引擎执行性能的提高是需要进一步研究的问题。

## 参 考 文 献


1 Workflow Management Coalition. The Workflow Reference Model. WFMC TC00-1003, 1994
2 Leymann F, Roller D. Production Workflow Concepts and Techniques. ISBN 0-13-021753-0, 2000
3 Workflow Management Coaliti-on. Workflow Management Coalition Specification: Terminology & Glossary. WFMC TC1011, 1944
4 吕映芝, 张素琴, 蒋维杜. 编译原理. ISBN 7-302-02732-3, 1998
5 Booch G. Design Patterns