第35卷 第19期 计 算 机 工 程 2009年10月
Vol.35 No.19 Computer Engineering October 2009
·安全技术· 文章编号：1000—3428(2009)19—0151—03 文献标识码：A 中图分类号：TP301.6

VLCC中的DAG并行算法

周 深，杨路明，段桂华
(中南大学信息科学与工程学院，长沙 410083)

摘 要：基于组件的密码学虚拟实验室(VLCC)采用有向无环图(DAG)的拓扑排序机制管理组件。在分析VLCC各组件之间的数据依赖和运行次序关系的基础上，提出一种新的基于Java多线程机制和“唤醒”机制的DAG并行算法。与拓扑排序算法相比，具有低算法时间复杂度的特点。实验结果表明，系统在新算法下较大地缩短了系统运行时间，提高资源使用效率和用户满意度，能更好地完善VLCC。
关键词：基于组件的密码学虚拟实验室；有向无环图；拓扑排序；并行

DAG Parallel Arithmetic in Virtual Laboratory of Cryptology Based on Components

ZHOU Shen, YANG Lu-ming, DUAN Gui-hua
(College of Information Science and Engineering, Central South University, Changsha 410083)

【Abstract】 Virtual Laboratory of Cryptology based on Components(VLCC) uses topological sorting in Directed Acyclic Graph(DAG) to manage the data-dependant and execute-order between components. Based on the Java thread and “wake” mechanism, this paper proposes a new DAG parallel arithmetic. Compared with topological sorting, new arithmetic’s time-complexity is highly reduced. Experimental result shows that VLCC works much faster and the usage rate of resource is heightened, and makes user more satisfied. New arithmetic is a better solution for VLCC.
【Key words】 Virtual Laboratory of Cryptology based on Components(VLCC); Directed Acyclic Graph(DAG); topological sorting; parallel

1 概述

随着计算机虚拟技术和网络教学的不断发展，计算机虚拟实验室逐渐成为虚拟技术的研究热点之一。近年来，国内外各高校和研究机构都致力于研制和开发自己的虚拟实验室系统[1-2]，并已取得一系列研究成果。

基于组件的虚拟实验室[3]将各种实验流程分解为一个个算法组件，并将组件以可视化的形式展现出来。与同类系统相比，基于组件虚拟实验室的优点在于：

(1)组件是对实验流程和各种算法的细分和精化，学生可更深入地了解各个实验和算法的细节，能更深刻地理解实验。

(2)实验和算法的组件化，使得系统的结构更加清晰和稳定。

(3)各个组件拥有更好的重用性、可扩充性和可拔插性，使得系统灵活性大为提升。

密码技术[4]是计算机信息安全的核心技术，是保护数据安全的重要手段之一。密码技术不仅被应用于军事、国防等高精领域，同时也被广泛应用于人们的日常生活，如个人账号、隐私等。因此，密码学成为各高校计算机专业必修的重要学科。

基于组件的密码学虚拟实验室(Virtual Laboratory of Cryptology based on Components, VLCC)是一种用Java语言设计的可视化密码学实验的教学平台。通过这个平台，学生可以自主地进行实验流程定制、保存仿真结果、对密码学算法进行分析和验证，以及自行编写算法加入到平台，从而可以让学生更深入地理解各种密码学算法的本质。

VLCC是面向多用户的实验系统，使用有向无环图(Directed Acyclic Graph, DAG)拓扑排序机制管理算法组件。系统需要在短时间内能够快速地处理大量用户请求，并及时将处理结果反馈给用户。因此，如何提高系统的处理和响应速度是整个系统设计需要重点考虑的问题。密码学算法计算量通常较大，而且对于经典算法一般来说很难再从算法级别上对速度进行改进。所以，本文着重从算法组件的组织和运行机制上进行分析，从而提高系统处理速度。

DAG并行算法[5]常用于分布式系统及网格计算和集群。国内对DAG算法在虚拟实验中的研究尚处于起步阶段。VLCC虽然是单处理器系统，但系统由多个组件组成，每个组件可视为一个单独的计算节点。通过资源调度和分配，构成一个基于计算节点的网格计算模型。由此可见，将DAG并行算法经过优化调整后应用于VLCC，可以提高系统计算效率。

2 VLCC关键技术和运行机制

VLCC使用Java Bean技术开发密码学算法组件、Java反射机制实现系统组件的自省功能、Java Swing技术实现系统的客户端。图1是密码学中EIGamal密码的加密实验设计图。

由图1可见，各个组件由连线相接，初始数据由用户通过输入组件输入，数据经过各个组件处理后再通过连线流向下一个组件，最终将实验流程的结果输出给用户。在整个实验流程中，如果将各个组件视为节点，连线视为路径，则整

作者简介：周 深(1983－)，男，硕士研究生，主研方向：虚拟技术，信息安全；杨路明，教授、博士生导师；段桂华，副教授、博士研究生
收稿日期：2009-06-02 **E-mail：**dsdzs@163.com

个实验可以被抽象成一个 DAG。由图 1 抽象的 DAG 如图 2 所示。

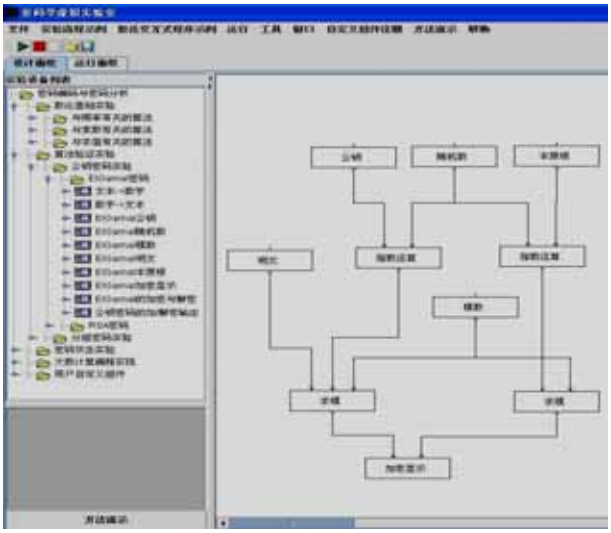

图 1　虚拟实验室系统

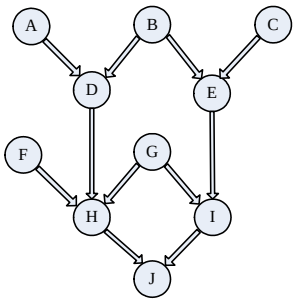

图 2　EIGamal 密码实验拓扑图

为了让数据正确地从起始节点流向终结节点，系统采用 DAG 的拓扑排序机制决定数据的流向和组件执行的次序。根据 DAG 拓扑排序算法，图 2 中各个节点的一种可能的排序顺序是：A-B-C-D-E-F-G-H-I-J，系统按照这个顺序依次运行各个组件。

3　DAG 并行算法

3.1　算法基本思想

由图 2 所示的 DAG 例子可以看出，D 节点的执行依赖于 A, B 节点的执行结果；H 节点的执行依赖于 F, D, G 节点；J 节点依赖于 H, I 节点，其他节点类似。进一步分析可知，对于 A, B 节点来说是可以同时执行的，因为 A, B 节点并不具有相互依赖性。DAG 拓扑排序可以解决节点间的依赖问题，即可以保证子节点一定在父节点执行完之后才执行。拓扑排序将所有节点按照一定顺序排成队列，然后按照队列顺序依次执行，本质上是串行的，所以不能让没有依赖关系的节点(如 A, B 节点)同时运行。

通过以上分析，可以得出改进算法的基本思想：让 DAG 中没有依赖关系的节点并行运行。

基于这个思想，本文提出一种基于 Java 多线程和“ 唤醒 ”机制的 DAG 并行算法模型。算法核心在于：(1)对于每个节点增加 2 个数据结构——父节点集合和子节点集合，用以记录各个节点之间的关系。(2)经典算法是基于任务调度的，由系统建立队列，然后根据队列次序运行。本算法只建立起始队列，由父节点根据节点关系主动“ 唤醒子节点 ”。算法包含 2 个部分，详细描述如下：

(1)初始化

Step1　从节点集合中取一个节点。

Step2　如果该节点有来自其他节点的连线，则将连线另一端的节点设置为该节点的父节点。如果该节点没有父节点，则将该节点加入队列。

Step3　如果该节点有通向其他节点的连线，则将连线另一端的节点设置为该节点的子节点。

Step4　将该节点置为“ 未执行 ”状态。

Step5　如果节点集合中下一个节点不为空，则跳往 Step1，否则结束。

(2)执行

Step1　取出队列中所有节点，为每个节点开启一个线程，并行执行节点。

Step2　队列中任何一个节点执行完毕后，将该节点置为“ 已执行 ”状态。

Step3 获取该节点的所有子节点。如果子节点不为空，则为每个子节点开启一个线程“ 唤醒 ”子节点，要求子节点执行。否则，该节点线程结束，跳往 Step 6。

Step4　子节点首先检查所有父节点，如果所有父节点全部为“ 已执行 ”状态，则子节点执行。否则，子节点继续“ 睡眠 ”(不执行)。

Step5　子节点执行完毕，则置为“ 已执行 ”状态。跳往 Step3。

Step6　如果所有线程都结束，则算法结束。否则，等待所有线程结束。

3.2　算法分析与比较

由算法描述可以看出，初始化过程需要遍历每一个节点和每一条边，因此，初始化过程是 $O(n+e)$的。其中，n 是节点个数；e 是边的条数。执行过程的理想状态是并行计算过程。假设每个没有父亲的节点都有一个无时间损耗的起始节点，每个没有孩子的节点都有一个无时间损耗的终结节点，同时忽略数据在边上的传播延迟，则整个执行过程可视为从起始节点到终结节点中关键路径的时间复杂度，记为 $O(n_{\sum i})$，其中，$\sum i$ 是指关键路径中节点执行时间之和。

拓扑排序时间复杂度是 $O(n+e)$，节点执行时间复杂度是 $O(n)$，由分析可知 $n_{\sum i} \leq n$。在最坏情况下，DAG 并行算法和拓扑排序的效率是相等的，其他情况下 DAG 并行算法都比拓扑排序效率更高。

以图 2 中 DAG 为例，假设每一个节点的执行时间都为单位时间 1，并假设算法的初始化时间忽略(初始化时间和节点执行时间相比是一个小计算量过程)，则拓扑排序后执行需要时间 10，而 DAG 并行算法的执行顺序为：(ABCFG)-(DE)-(HI)-(J)，括号内表示可以并行的部分。可知 DAG 并行算法需要的时间为 4，只有拓扑排序计算时间的 40%。

4　DAG 并行算法实验与分析

4.1　模型简化实验

根据算法描述，用 Java 语言实现 DAG 并行算法基本模型。在实际过程中为了实现 Java 多线程，还需要添加一张记录节点运行状态的表来供子节点查询父节点的运行状态，以及一个信号量来保证数据之间的同步和互斥。

图 2 和图 3 是几种典型的 DAG 图，用以测试 DAG 并行算法的实际执行效率。图 3 中箭头表示数据流向是从上往下。

拓扑排序和并行算法的初始化过程在量级上都是

$O(n+e)$，此处近似认为初始化过程两者时间消耗相等。在设计实验时，让每个节点做大计算量操作，使得两者在初始化过程中对比的误差尽可能减小。同时让每个节点都采取相同的操作，则可以认为每个节点执行操作的时间是一样的，并取这个执行时间为 1 单位时间。

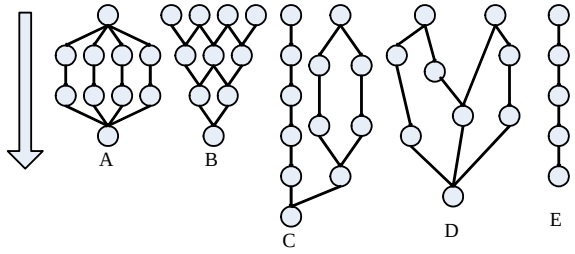

图 3　典型 DAG 图

表 1 是拓扑排序算法和 DAG 并行算法的理论运行时间和实际运行时间的对比。实际运行时间是在单服务器(主频 1.8 GHz，内存 1 GB，Windows XP 操作系统)上测试取得的。

表 1　算法模型实验结果

图像	理论值			实际值		
	拓扑排序/s	并行算法/s	并行/拓扑	拓扑排序/ms	并行算法/ms	并行/拓扑
图 2	10	4	0.40	6 377	3 611	0.57
图 3(A)	10	4	0.40	6 403	3 915	0.61
图 3(B)	10	4	0.40	6 469	3 609	0.56
图 3(C)	12	6	0.50	7 750	4 683	0.60
图 3(D)	9	4	0.44	5 844	3 297	0.56
图 3(E)	5	5	1.00	3 212	3 350	1.04

由表 1 的结果可以看出，实际比值比理论比值稍高。这是由于在算法模型的实际运行中并不是真正地并行执行，而是采用的 Java 多线程机制。在单处理器的情况下，Java 多线程程序依赖于 Java 编译器对于程序的优化、依赖于操作系统本身对于多线程的调度机制、依赖于底层硬件资源的占用情况。因此，在系统资源如 CPU 运算器、累加器、内存等资源有限的情况下，多线程程序最好情况下应该是达到流水线处理方式。同时程序为了对线程进行管理，要考虑数据之间的同步和互斥，也增加了额外的数据量和计算量。

实验结果表明：在 DAG 并行算法模型下，系统运算速度有很大的提升。

假设当各个节点的计算量与初始化、线程数据同步开销相比足够大，硬件资源足够多，则实际比值无限接近于理论比值。

4.2　实验

在 VLCC 中，用 DeviceCarrier 类表示组件，用 Deviece Connector 类表示连线。将 DAG 并行算法模型运用于 VLCC，需要对原系统进行重构。

(1)在 DeviceCarrier 类中增加数据结构的代码如下：

```
private List<String> fatherCarrierList = new ArrayList<String>(); //父节点的集合，集合内是父节点的名字
private List<String> childrenCarrierList = new ArrayList <String>(); //子节点的集合，集合内是子节点的名字
private boolean isRun = false; //判断这个 carrier 是否已经在运
//行，如果是，则不开启另外一个线程
```

(2)在 DeviceCarrier 类中添加新的方法的代码如下：

```
private boolean checkCanDo(cs_DeviceCarrier nextCarrier) throws Exception; //检测一个节点是否能够运行，如果子节点要运行，
//需要它的所有父节点都执行完
private void setRelations(DeviceCarrier carrier); //设置父子关系
public void run(); //多线程方法，用于执行节点计算
```

系统经过重构后，依然保持良好的运行稳定性和计算结果的准确性。表 2 是 VLCC 中已实现的几种典型密码学算法在拓扑排序和 DAG 并行算法情况下运行时间的对比结果。由于各个算法组件实际运行时间不同，因此只列出实际运行时间(取 10 次计算时间之和)。测试环境和 4.1 节的实验环境相同。

表 2　VLCC 实验结果

实验名称	拓扑排序运行时间/ms	并行算法运行时间/ms	并行/拓扑
求素数本原根	1 154	638	0.55
EIGamal 密码的加密	1 721	780	0.45
IDEA 密码第 1 轮加密	5 041	2 038	0.40
BBS 伪随机序列生成器	1 175	1 252	1.07
DES 加密含 festil 网络	1 889	1 007	0.53
DES 加密无 festil 网络	3 592	1 255	0.35

表 2 结果表明，几种密码学算法在 DAG 并行算法下的计算时间和模型、理论值是吻合的。

5　结束语

本文通过对 VLBCB 运行机制：DAG 拓扑排序的讨论和分析，指出拓扑排序在运算速度、执行效率上的不足。根据“让没有依赖关系的节点并行执行”的思想，设计并实现了一种适用于组件的 DAG 并行算法模型。通过分析算法复杂度，从理论上说明了该算法在计算速度上比原算法更快。通过简单建模，证明了该算法的实际可行性。将并行算法模型运用于 VLCC 中，实验数据证明：新算法不仅保证了系统正确性，而且计算速度和模型及理论值是一致的。

下一步工作着重点在于：如何有效地拆分和组件化各种密码学算法，优化已有的组件，使得密码学算法能够尽可能地在 DAG 并行算法下提高运算速度。

参考文献

[1] Fabrega L, Massaguer J, Love T, et al. A Virtual Network Laboratory for Learning IP Network[C]//Proc. of the 7th Annual Conference on Innovation and Technology in Compute Science Education. Aarhus, Denmark: [s. n.], 2002: 89-95.

[2] 王建新, 陆炜妮, 王伟平. 基于组件的数字图像处理仿真系统的设计与实现[J]. 系统仿真学报, 2004, 16(6): 1213-1216.

[3] 段桂华, 潘卫敏, 王伟平. 基于组件的密码学虚拟实验平台的设计与实现[J]. 吉首大学学报: 自然科学版, 2007, 28(3): 40-44.

[4] Schneier B. Applied Cryptography: Protocols, Algorithms, and Source Code in C[M]. 北京: 机械工业出版社, 2004.

[5] 石　威. 相关任务图的均衡动态关键路径调度算法[J]. 计算机学报, 2001, 24(9): 991-997.

编辑　顾逸斐