第33卷 第10期 计 算 机 工 程 2007年5月
Vol.33 No.10 Computer Engineering May 2007
·软件技术与数据库· 文章编号：1000—3428(2007)10—0087—03 文献标识码：A 中图分类号：TP311

图形化编程中模块间并行性的自动挖掘

尹华祥[1,2]，丁玉磊[2]，徐 铸[2]，洪学海[2]，刘金刚[1]，冯圣中[2]，邱振戈[2]
(1. 首都师范大学计算机应用联合研究院，北京 100037；2. 中国科学院计算技术研究所国家智能计算机中心，北京 100080)

摘 要：在基于模块组合的图形化编程中，存在大量互不依赖的模块，这些模块具有并行执行的性质。翻译程序以拓扑排序算法遍历该有向无环图，为每个模块产生一个线程，为每条输入线产生一个信号量，以同步有依赖关系的模块的执行顺序，最终产生一个可并行执行的多线程程序，从而达到自动挖掘并行性、提高所生成程序的运行效率的目的。

关键词：图形化编程；自动挖掘；多线程；并行性

Automatic Mining of Parallelism Between Modules in Graphic Programming

YIN Huaxiang[1,2], TING Yulei[2], XU Zhu[2], HONG Xuehai[2], LIU Jingang[1], HONG Shengzhong[2], QIU Zhenge[2]
(1. Joint Academy of Computer Application, Capital Normal University, Beijing 100037;
2. National Center of Intelligent Computer, Institute of Computer Technology, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 In graphic programming based on module composition, there are many modules which are independent of each other and can be executed in parallel. The translating program traverses the DAG, generates a thread for every module, and generates a semaphore for every arc in order to synchronize the executing sequence of the modules which are dependent on each other, then generates a multithread program, which runs in parallel more efficiently at last.

【Key words】 Graphic programming; Automatic mining; Multithread; Parallelism

本文在国家自然科学基金重大项目“网络计算环境综合试验平台”的子课题“以网络为基础的科学活动环境研究”的基础上，研究目标定位在网络、网格所带来的共享、协作和高性能等方向。为了体现网格的共享特征，把已有的代码按照事先定义好的规范包装成模块，并提供一个代码共享平台，便于用户根据功能、开发平台等组合条件进行搜索。为了体现网格的协作特征，我们的目标是实现即时协作编程。为了用户方便使用，本文初步实现了基于模块组合的图形化编程基础上的协作编程环境。

在实现基于模块的图形化编程环境时，发现图形化程序(即图形化编程得到的图形)中，存在大量互不依赖的模块，而这些模块可以并行执行。因此通过用程序将图形化程序翻译成多线程程序，达到自动挖掘并行性的目的。

1 基于模块组合的图形化编程

软件发展到今天，积累了很多完成特定功能的函数。在用户实现一个特定功能时，很可能可以通过组合使用已有的函数来实现，或者自己写少量函数再与已有函数组合实现。为了支持用户方便地以图形方式组合函数，我们给函数加描述信息，把函数包装成模块，并以一个方框表示一个模块。这样，用户用连线将方框连接起来，得到用图形表现的程序，然后由程序将该图形翻译成代码。需要说明的是，这里的连线代表数据。

在翻译成代码时，由于整幅图是一个有向无环图，因此可以用拓扑排序算法遍历该图，依次翻译对每个模块的调用。从数据结构角度来说，每个顶点类的成员主要有：向量(Vector)类型的对象，表示流入该顶点的边的引用和流出该顶点的边的引用；一个表示该顶点对应模块的引用。边类的成员主要有：该边流出的顶点的引用，该边流入的顶点的引用，该边的变量名。算法具体步骤如下：

```
for(每个顶点){
  求入度；
  如果入度为0，将该顶点放入队列；
}while(队列非空){
    从队列中取出一个顶点；
    将该顶点每个输入的边的变量名加入调用参数表中；
    如果该顶点有输出，且用户没有为流出的边设置变量名，则为流出的边分配一个变量名；
    生成调用该模块的代码；
    将该顶点每个流出边的流入顶点的入度减 1，如果入度为0，也要将该顶点加入队列；
}
```

例 1 主要目标是将一幅图像(tar.bmp)的纹理和一幅图像(ref.bmp)的颜色综合成一幅图像；但完成综合的模块SynthesisImage是别人写的一个Matlab模块，需要2幅图像的格式为lab格式，于是又找到将bmp图像转化为lab图形的模块rgb2lab；考虑到tar.bmp图像有很多噪声，找了一个过滤图像噪声的模块FilterWave。完成该功能的模块组合如图1。

基金项目：国家自然科学基金资助项目(90412010)；国家“863”计划基金资助项目(2004AA616010)
作者简介：尹华祥(1982－)，男，硕士生，主研方向：网格和高性能计算；丁玉磊，硕士生；徐 铸，研究实习员；洪学海，博士、副研究员；刘金刚，博士、研究员、博导；冯圣中，博士、高工；邱振戈，博士、副研究员
收稿日期：2006-06-10 **E-mail：**yhx@ercist.iscas.ac.cn

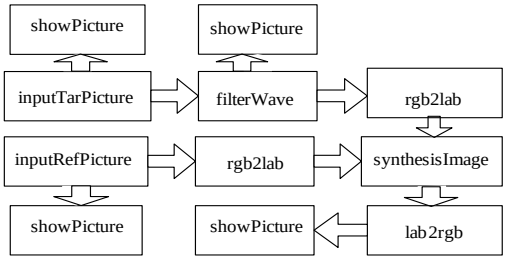

图 1 例 1 的模块组合

自动翻译得到的代码如下(无需作任何修改即可在 Matlab 中运行)：

```
function genFile()
var0 = inputTarPicture(    )
var1 = inputRefPicture(    )
var2 = filterWave( var0 )
showPicture( var0 )
var3 = rgb2lab( var1 )
showPicture( var1 )
var4 = rgb2lab( var2 )
showPicture( var2 )
var5 = synthesisImage( var4, var3 )
var6 = lab2rgb( var5 )
showPicture( var6 )
end
```

2 自动挖掘并行性

在前面的例子中，将 tar.bmp 转化为 lab 格式的模块与将 ref.bmp 转化为 lab 格式的模块是互不依赖的，可以按任意顺序执行。为了挖掘这种并行性，就需要存在多个执行流；要实现多个执行流，可以做成多个进程，也可以做成多线程，考虑到多线程既可以共享资源，又容易在多个线程中共享数据，就采用多线程。为了保持简单的美，也考虑到一般模块不是很多，线程的开销不是主要矛盾，故没有优化，而是在主线程中，为每个模块生成一个线程，然后启动这些线程，最后等待这些线程结束。

有些模块之间有数据连线，比如输入 tar.bmp 的模块与将 tar.bmp 转化为 lab 格式的模块之间有连线，表示后者需要前者执行完产生的数据。为每条连线生成一个初始值为 0 的信号量，在该连线流出的模块所对应的线程代码中，调用执行完该模块后，就对该信号量执行 V 操作，通知数据准备好；在该连线流入的模块所对应的线程代码中，调用执行该模块前，对该信号量执行 P 操作，表示只有连线对应的数据准备好后，才可以执行该模块。为了便于实现，边类添加一个信号量名的成员。

综上所述，翻译算法如下：

```
indegree = 0；
for(每个顶点){
求入度，放入一个映射(HashMap)对象中，以顶点对象的引用为关键字；
如果入度为 0，将该顶点放入队列；
将该顶点入度加到 indegree 中；
}
生成 indegree 个信号量的定义代码；
n = 1；
while(队列非空){
    从队列中取出一个顶点;
    将该顶点每个输入的边的变量名加入调用参数表中；
    生成第 n 个线程的头；
    生成对该顶点每个输入的边的信号量的 P 操作代码；
    如果该顶点有输出，且用户没有为流出的边设置变量名，则为流出的边分配一个变量名；
    生成调用该模块的代码；
    如果该顶点有输入，为每条输出边分配一个信号量，并生成对该信号量的 V 操作代码；
    将该顶点每个流出边的流入定点的入度减 1，如果入度为 0，也要将该顶点加入队列；
    生成第 n 个线程的尾；
    n++；
}
生成这 n 个线程的启动(start)代码；
生成等待(join)这 n 个线程的代码；
(算法结束)
```

为了验证这种方法的可行性，本文做了演示的例子。考虑到 Matlab 脚本语言不支持多线程编程，生成的代码不是 Matlab 代码，而是 Java 代码。需要说明的是，Java 语言中，没有直接提供信号量的 P、V 操作，于是编写一个信号量的类，定义如下：

```
public class MySemaphore {
  private int semaphore;
  public MySemaphore() {
        semaphore = 1;
  }
  public MySemaphore(int i) {
        semaphore = i;
  }
  public synchronized void P()
  {
        semaphore--;
        if (semaphore < 0) {
              try {
                    wait();
              } catch (InterruptedException ie) {
  ie.printStackTrace(System.err);
              } }
  }
  public synchronized void V() {
        semaphore++;
        if (semaphore <= 0)
              notify();
  } }
```

例 2　完成一个简单矩阵运算 A*B+C*D，仅仅是为了验证可行性。该例子的模块组合如图 2。

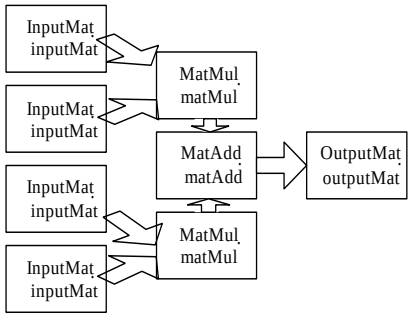

图 2 矩阵运算 A*B+C*D 的模块组合

自动翻译生成的代码，无需作任何修改即可编译运行。考虑到篇幅，只节选生成 A*B 矩阵运算的代码如下：

```java
import java.util.Hashtable;
public class GenClass {
  public static void main(String[] args) {
        final Hashtable hashtable = new Hashtable();
        final MySemaphore sema1 = new MySemaphore(0);
        final MySemaphore sema2 = new MySemaphore(0);
        final MySemaphore sema3 = new MySemaphore(0);
        Thread thread1 = new Thread(){
              public void run(){
                    double[][] var1 = InputMat.inputMat();
                    hashtable.put("var1", var1);
                    sema1.V();   } };
        Thread thread2 = new Thread(){
              public void run(){
                    double[][] var2 = InputMat.inputMat();
                    hashtable.put("var2", var2);
                    sema2.V();   }    };
        Thread thread3 = new Thread(){
              public void run(){
                    sema1.P();
                    sema2.P();
                    double[][] var3 =
MatMul.matMul((double[][])hashtable.get("var1"),
(double[][])hashtable.get("var2"));
                    hashtable.put("var3", var3);
                    sema3.V();   } };
        Thread thread4 = new Thread(){
              public void run(){
                    sema3.P();
OutputMat.outputMat((double[][])hashtable.get("var3"));
                    } };
        thread1.start();
        thread2.start();
        thread3.start();
        thread4.start();
        try{
              thread1.join();
              thread2.join();
              thread3.join();
              thread4.join();
        }catch(InterruptedException ie){
JOptionPane.showMessageDialog(null, "InterruptedException");
        } }   }
```

3 结论

在基于模块组合的图形化编程中，存在大量互不依赖的模块，这些模块具有并行执行的性质。通过程序将图形化程序自动翻译成的多线程程序，代码长度有所增长，但通过在多处理器的机器上并行执行可以显著提高运行效率。本文提出的方法，也可以用于提高工作流的效率。

参考文献

1 Oaks S. Java 线程[M]. 2 版. 北京: 中国电力出版社, 2003.

2 刘君华, 贾惠芹, 丁 晖. 虚拟仪器图形化编程语言 LabVIEW 教程[M]. 西安: 西安电子科技大学出版社, 2001.

3 Van der Aalst W, Van Hee K. 工作流管理——模型方法和系统[M]. 北京: 清华大学出版社, 2004.

(上接第 80 页)

图 1 中的横坐标代表遗传算法的迭代次数，纵坐标代表每一代种群得到的最优结果(即最低的边界点比例)。在实验中为设定的参数 fn(选择特征数)为 3。从图 1 中可以看出，在遗传算法的迭代过程中，Prostate Cancer 数据集的边界点比例在持续下降，最后收敛到一个较低的比例 26.47%。图 2 给出了整个数据集样本在所选出的 3 维特征空间投影的散点图，其中“ * ”代表前列腺癌样本，“ ▽ ”代表正常样本。从图 2 可看出，在所选出的 3 维特征空间中，该数据集具有较好的可分性。为了进一步验证所选特征的可分性，分别采用 Fisher 线性分类算法及 k 近邻分类算法对降维后的该数据集进行训练及测试，分类正确率分别为 82.5%及 91.2%。从 12 599 维的高维特征空间降至 3 维特征子空间，且获得较高的分类正确率，说明 GABPFL 算法能较为有效地进行特征选择。

5 结论

本文主要针对高维数据的特征选择问题，提出了一种基于可分性度量及遗传算法的特征选择算法，在编码方式、评价方法及适应度定义等方面对传统的基于遗传算法的特征选择方法进行了改进。针对高维基因表达数据的实验证明，该算法能较为有效地找出具有较好的可分离性的特征子集，并可得到较高的分类精度。

参考文献

1 John G H, Kohavi R, Pfleger K. Irrelevant Features and the Subset Selection Problem[C]//Proceedings of the 11^{th} International Conference on Machine Learning. 1994: 121-129.

2 Kohavi R, John G H. Wrappers for Feature Subset Selection[J]. Artificial Intelligence, 1997, 97(1/2): 273-324.

3 Jiang Daxin, Tang Chun, Zhang Aidong. Cluster Analysis for Gene Expression Data: A Survey[J]. IEEE Transactions on Knowledge and Data Engineering, 2004, 16 (11): 1370-1386.

4 Liu Huan, Yu Lei. Toward Integrating Feature Selection Algorithms for Classification and Clustering[J]. IEEE Transactions on Knowledge and Data Engineering, 2005, 17(5): 491-502.

5 Sancho J, Pierson W E, Ulug B, et al. Class Separability Estimation and Incremental Learning Using Boundary Methods[J]. Neurocomputing, 2000, 35(1-4): 3-26.

6 Ho T K, Basu M. Complexity Measures of Supervised Classification Problems[J]. IEEE Transaction on Pattern Analysis and Machine Intelligence, 2002, 24(3): 289-300.

7 Yang J, Honavar V. Feature Subset Selection Using a Genetic Algorithm[J]. IEEE Intelligent Systems, 1998, 13(2): 44-49.

8 Bir B, Lin Yingqiang. Genetic Algorithm Based Feature Selection for Target Detection in SAR Images[J]. Image and Vision Computing, 2003, 21(7): 591-608.

9 II-Seok Oh, Lee Jin-Seon, Moon Byung-Ro. Hybrid Genetic Algorithms for Feature Selection[J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004, 26 (11): 1424-1437.

10 Han Jiawei, Kamber M. 数据挖掘概念与技术[M]. 范 明, 孟小峰, 译. 北京: 机械工业出版社, 2001.