

基于 LLVM 架构的图像处理程序的并行分类

王时雨,张盛兵,安建峰,黄小平,白弘猷
(西北工业大学 计算机学院,陕西 西安 710072)

摘 要:设计了一种针对图像处理类应用程序的并行性分类的方法,并使用工具初步实现了对程序的分类,还研究了图像处理算法的并行性.通过对图像处理算法的主要特征进行分析,得出图像处理算法包含多种并行特征的结论,分别从图像处理程序的混合结构的模型,以及程序中的依赖关系和并行特征两方面进行了分析.结合对程序的分类方式,使用 LLVM 编译架构初步实现了自动对程序中的程序段进行识别其中的分类模式.

关键词:图像处理;混合架构;依赖分析;并行性分类;编译器架构

中图分类号: TP31 **文献标识码:** A **文章编号:** 1000—7180(2018)01—0066—06

Parallel Classification of Image Processing Program Based on LLVM Architecture

WANG Shi-yu, ZHANG Sheng-bing, AN Jian-feng, HUANG Xiao-ping, BAI Hong-you

(School of Computer Science and Engineering, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: According to the main characteristics of the image processing algorithm, the conclusion that the image processing algorithm contains a variety of parallel features is analyzed from the model of the mixed structure of the image processing program and the dependencies and parallel features in the program. Then, according to the analysis of the dependencies in the loop, the parallelization of the different levels of the program is found, and the loops are classified. The different categories correspond to the different execution units, and the typical image processing procedures are further analyzed.

Key words: image processing; hybrid architecture; dependency analysis; parallelism classification; compiler architecture

1 引言

近些年来,图像处理类应用程序越来越多,图像处理类的算法本身也很复杂.人们对图像分辨率和实时性的要求越来越高,需要处理的数据量也越来越大.同时图像处理算法的复杂度也在不断地增加,因此提高处理图像程序的效率显得至关重要.单独用传统处理器远远满足不了这类程序的需求^[1]. GPU 其本身的结构更加适合处理数据级并行的操作. FPGA 的高性能来自于其灵活性,以及大量的片上存储器支持高并行性.对于图像处理程序,三者结合的性能远超过了单独使用任意一种架构时的性能^[2].

使用混合架构对图像处理类应用程序进行加速,对图像处理类应用程序进行并行性分析和分类有利于编程人员实现并优化相关程序,因此对程序进行进一步分析是很有必要的.对图像处理类应用程序的并行性进行分析主要是分析程序中的依赖关系和循环部分,依赖关系决定程序的执行顺序,而循环则是程序多次执行的部分,包含大量的并行特征.而根据并行性的分析进行分类则是为了提供一种使得程序在不同架构上执行的方法.

本文通过对图像处理算法的主要特征进行分析,得出图像处理算法包含多种并行特征的结论,分别从图像处理程序的混合结构的模型,以及程序中的依赖关系和并行特征两方面进行了分析.并对典

型的图像处理程序进行了进一步的分析. 结合对程序的分类方式,使用 LLVM 编译架构初步实现了自动对程序中的程序段进行识别其中的分类模式.

2 图像处理算法分类

数字图像处理能够将图像信号转换成数字信号,同时利用计算机对图像程序进行各种方式的处理方法和技术^[3-6]. 图像处理类应用程序中存在着许多图像处理的算法,而且这些算法主要分为图像几何变换、图像增强、图像复原和重建、图像分割、图像压缩编码^[7]等.

从上述图像处理算法中可以发现这些算法的差异很大,而这些算法大多数都存在着不同的并行特征. 图像处理算法通常处理的数据规模很大,但其中存在很多可并行的部分,不仅存在着数据级并行,也存在指令级并行和线程级并行等多种并行部分. 图像算法中包含的操作也可以看出,在图像变换类程序大量存在着数据级并行,不同数据的计算不存在数据相关性,适合并行处理.

3 图形处理类程序并行性分析

3.1 程序的依赖关系

在对图像处理类应用程序的并行性分析的时候,首要的是分析程序中数据之间的关系. 串行程序存在着许多依赖关系,而图像处理类程序也存在着这些特征,甚至更为突出,由于程序中存在着依赖关系,而程序的执行不能违背这种顺序,程序的并行执行也要遵循这种关系. 因此研究程序的依赖关系是必要的.

定义 1 假设语句 S、T 分别为不同语句, $W()$ 、 $R()$ 为语句的读写集合. 对不同语句的读写集合分别进行交集操作:

$$W(S) \cap W(T) \neq \emptyset$$

$$W(S) \cap R(T) \neq \emptyset$$

$$R(S) \cap W(T) \neq \emptyset$$

若满足以上三种情况之一,就代表 S 与 T 存在数据依赖关系. S 与 T 之间若存在依赖关系,必然是访问同一个内存单元,而且至少有一次写操作. 根据读写顺序的不同,将依赖可以划分为三种,假设 S 优先于 T 执行,某一公用的变量设为 A.

若 S 先对 A 进行了一次赋值,然后 T 又对 A 进行了一次赋值,该依赖关系称为输出依赖.

若 S 先对 A 进行了一次赋值,然后 T 又对 A 进行了一次读取,该依赖关系称为流依赖.

若 S 先对 A 进行了一次读取,然后 T 又对 A 进行了一次赋值,该依赖关系称为反依赖.

定义 2 假设语句 S、T 分别为不同语句, S 必须执行在 T 之前,这样的依赖关系称为控制依赖关系. 程序中的 if 等条件转移语句,也属于控制依赖关系.

程序控制流图 CFG(Control flow Graph)能够表示程序结构. 程序控制流图以基本块为单位,能够清楚的显示出程序基本块之间的执行关系,一个简单的单层循环程序控制流程如图 1 所示.

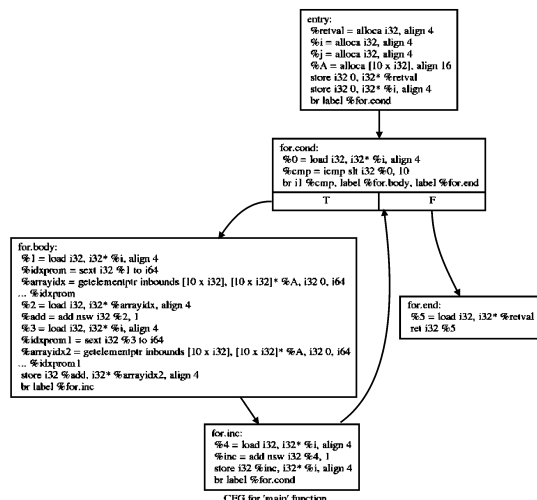


图 1 程序控制流图

3.2 多粒度并行分析

图像处理算法中常常将图像划分为多个部分进行处理,每个部分有可以划分为更小的处理过程,因此可以从不同的粒度对图像算法进行并行处理,下面以高斯滤波为例来说明图像处理算法中的并行粒度.

在高斯滤波中,需要使用特定的模板对图像进行二维卷积操作. 如图 2~5 所示,二维卷积可以看作是加权求和的过程,计算公式如式(1)所示.

$$z = \frac{1}{9} (w_1 z_1 + w_2 z_2 + \dots + w_9 z_9)$$

$$= \frac{1}{9} \sum_{i=1}^9 w_i z_i \quad (1)$$

如图 2 所示,一幅图像在进行卷积操作时,可以将原始图像分为适当大小的图像块,每个图像块分配到不同的处理单元中并行处理.

与粗粒度并行相比,如图 3 所示,细粒度并行划分的任务量更小. 例如单个原始图像块与卷积模板进行卷积操作,卷积过程需要 9 次乘法操作,8 次加法操作. 但是,各个像素点间相互独立,因此可以将

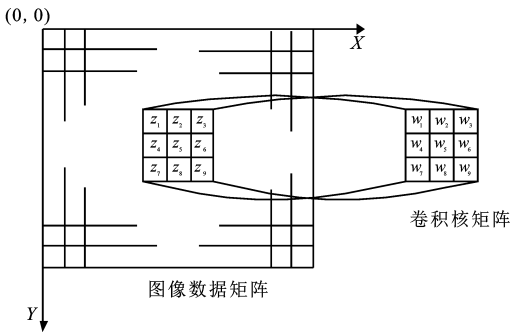


图 2 卷积核为 3×3 的图像卷积

乘法操作并行执行,如图 4 所示.从图中可知,累加操作并不像乘法操作一样可以完全并行执行,因为下一步的加法操作需要等待上一步的计算结果,利用树形处理的方式可以计算,树的每一个层次可以并行处理.

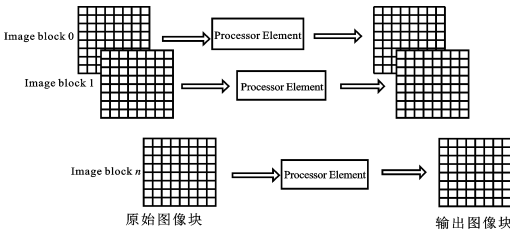


图 3 粗粒度并行

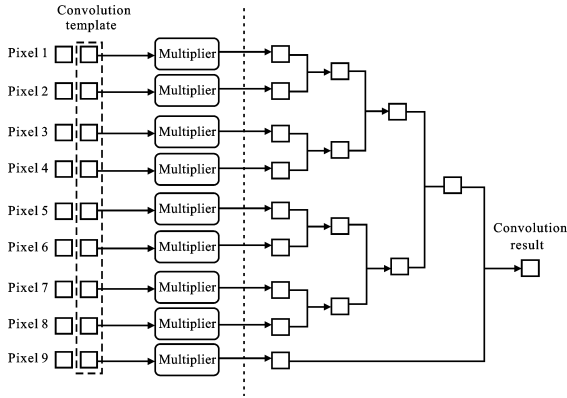


图 4 细粒度并行

图像处理类应用的数据级并行特征很突出,但在不同体系架构的硬件平台下,算法可以被组织为不同的并行粒度来执行.因此,一个并行算法在某一个并行体系结构上可能表现得非常高效,但在另一个体系结构上却非常糟糕.在权衡存储和硬件资源的基础上,充分细化算法并行粒度是挖掘其并行性的重要手段.

4 并行性分类实现

在对图像处理类应用程序进行了并行性分析

后,为了帮助编程人员找到合适的加速方式,设计一种能够更直观体现对程序的并行性分类的结构就显得尤为重要.

4.1 系统算法设计

为了初步实现对程序的分析,首先要找出程序中是否存在循环,其次判断循环是单层循环还是嵌套循环,最后再判断出循环里的控制依赖和数据依赖.算法流程如图 5 所示.

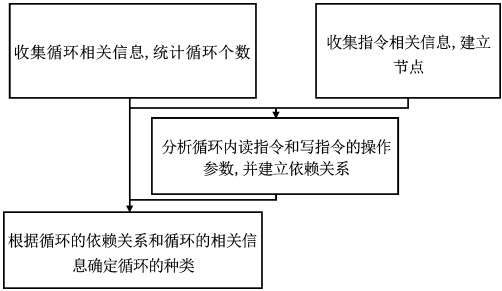


图 5 整体算法流程

算法步骤实现如下所示.

(1)统计程序中所有出现的循环.

统计程序中所有出现的循环. LLVM 中提供了对循环的操作的接口,通过利用接口可以提取到程序中的循环信息. 首先建立一个 LoopcountPass,让这个 pass 继承 LoopPass, LoopPass 中提供了寻找循环和对循环进行判断以及获取循环中的想关信息的函数. 通过 runOnLoop(Loop * L, LPPassManager &) override{}对这个函数的重写,在判断循环的过程中,通过是指一个循环统计变量 LoopCount 用来统计循环出现的个数. 其中用 L->getParentLoop()判断循环是否是外层循环来统计嵌套循环的个数. L->getParentLoop()为假时,通过累加得到循环出现的个数. 另外, for (Loop::block_iterator b = L->block_begin(), be = L->block_end(); b != be; ++b) {}该循环用来统计循环中出现的基本块的地址,对循环中出现的基本块的地址用来储存.

下面是一个简单的循环程序:

```
int main()
{
    int i,j;
    int A[10],B[10],C[10],D[10];
    for(i=0;i<10;i++)
        A[i]=A[i]+1;
    for(i=0;i<10;i++)
        for(j=0;j<10;j++)
```

```
B[i]=C[i]+D[i];
```

}

通过测试程序,能够统计出程序中的所有循环,如图 6 所示。另外简单的给循环标记出单层循环和嵌套循环的个数,以便后面的分析。

```
erces@erces-ESC2000-G2:~/baihy/build/lib/Transforms/dependences$ clang++ `llvm-config --libs --ldflags --cxxflags` LoopCountPass.cpp -std=c++0x -fPIC -shared -o LoopCountPass.so
erces@erces-ESC2000-G2:~/baihy/build/lib/Transforms/dependences$ opt -load ./LoopCountPass.so -LoopCount <tl.bc> /dev/null
LoopCount 2
simpleLoop 1
nestLoop 1
erces@erces-ESC2000-G2:~/baihy/build/lib/Transforms/dependences$
```

图 6 循环结果统计

(2)根据 IR 将每条指令做成一个节点

根据 IR 将每条指令做成做一个节点,为每条指令建立调用关系。通过调用 runOnModule 对所有函数进行统计方便之后维护,通过 runOnFunction 记录函数中的每个块,并统计块中的相关信息,在对块中的每条指令进行统计,通过 inst->getOpcode()来统计程序产生的 IR 中的指令类型的个数。并统计每条指令的操作码和对应的操作码的参数个数,另外设置节点存放操作码参数的相关信息。常用的操作码参数信息如表 1 所示。

表 1 常用操作码

Operator	Name	Parameter
2	br	1
8	add	2
10	sub	2
26	alloca	1
27	load	1
28	store	2
29	getelementptr	3
35	sext	1
46	icmp	2

对所有参数进行统计以后,根据每条指令的调用情况,先将所有指令的参数建立有向边。再进行一次参数合并,当后面的指令的参数与前面的指令中的参数相同时,进行一次合并,并去除相同的节点,剩下的节点就构成了一个有向无环图。

(3)建立读写集合

建立读写集合是对程序依赖分析中的最重要的一步,通过建立读集合和写多重集,就能够找出程序中的调用关系和依赖关系。

通过分析 IR 中的读指令和写指令后,建立读

集合和写多重集,然后将读集合和写多重集的元素进行比较,若读集合的某元素与写多重集的某元素拥有完全一样的参数节点,程序中存在流依赖和反依赖,再对写多重集之间的元素进行比较,若出现多个元素拥有一样的参数节点,则说明程序中出现了输出依赖,这些依赖都是循环无关的依赖关系。其次,判断两个数据是否有数组相关关系,出现这种相关关系,往往说明程序中出现了循环携带的依赖关系。

(4)确定循环种类

在对循环进行分析时,主要是针对循环体的程序进行分析,给定一个循环如下:

```
int main()
{
    int i,j;
    int A[10];
    for(i=0;i<10;i++)
        A[i]=A[i]+1;
}
```

在对循环分析发现,在中间格式时将循环分成了多个部分,如图 7 所示。

```
for.cond:                                ; preds = %for.inc, %entry
%0 = load i32, i32* %i, align 4
%cmp = icmp slt i32 %0, 10
br i1 %cmp, label %for.body, label %for.end

for.body:                                ; preds = %for.cond
%1 = load i32, i32* %i, align 4
%idxprom = sext i32 %1 to i64
%arrayidx = getelementptr inbounds [10 x i32], [10 x i32]* %A, i32 0, i64 %idxprom
%2 = load i32, i32* %arrayidx, align 4
%add = add nsw i32 %2, 1
%3 = load i32, i32* %i, align 4
%idxprom1 = sext i32 %3 to i64
%arrayidx2 = getelementptr inbounds [10 x i32], [10 x i32]* %A, i32 0, i64 %idxprom1
store i32 %add, i32* %arrayidx2, align 4
br label %for.inc

for.inc:                                  ; preds = %for.body
%4 = load i32, i32* %i, align 4
%inc = add nsw i32 %4, 1
store i32 %inc, i32* %i, align 4
br label %for.cond

for.end:                                  ; preds = %for.cond
%5 = load i32, i32* %retval
ret i32 %5
```

图 7 循环结构

在分析时,主要针对循环的循环体进行数据和依赖关系的分析,而循环其他部分的内容主要来判断循环层是否属于规范循环层。在对循环进行分析时,由于结合循环和依赖关系进行分析时,是对 BasicBlock 中的内容进行分析,如果循环中调用了函数,甚至调用的函数中仍含有循环,对于这种情况在目前设计中无法判断,将这一类程序也归于保守分析。对于非规范循环和循环中出现了调用函数的情况,由于这些程序特有的属性在分析中容易造成分析的偏差,所以在对于这类的循环做并行性分析时,认为程序是无法进行并行的。

通过将 LoopCountPass 中的循环内信息与 FindDependence Pass 中的读写集合和依赖关系结

合起来产生一个 ParallelAnalyse Pass. Parallel Pass 能够找出程序中的所有循环,对 for 循环能够进行初步的并行性分类,并能统计出程序中不同循环的个数.

4.2 测试与评估

4.2.1 测试程序

测试程序从图像处理类应用程序中选取,在图像处理程序中卷积运算多次出现,由于卷积运算的运算量大,有进行并行性计算的需求,选取卷积运算作为循环分析的程序是非常合适的,下面是最基本的 4 层循环实现卷积核心的循环,其核心代码如下:

```
for(row=0;row<R;row++){
for(col=0;col<C;col++){
for(i=row;i<K;i++){
for(j=col;j<K;j++){
output[row][col] += weights[i][j] * input[row+i]
[col+j];
}
```

在 Linux 系统下安装 LLVM 编译工具链,首先根据编写好的对程序循环进行查找和对程序进行分析的 pass 生成 .so 文件.其次将输入的卷积程序. C 文件通过编译变成中间格式文件. bc 或者. ll,最后使用 opt 工具就能够实现对原始程序的分析. 操作步骤和结果如图 8、图 9 所示.

4.2.2 测试结果

在测试程序的选取上,挑选了图像处理应用程序中的典型程序,其中包括图像变换类程序、图像增强类程序、图像复原和重建类程序、图像形态学处理程序,以及图像分割和图像匹配等多个程序.对这些程序分别进行了手动的并行性分析和分类,并使用工具对这些程序进行了自动识别,如表 2 所示.

表 2 程序测试结果

	循环数	可并行循环	识别出的循环
图像平移	1	1	1
图像镜像	2	2	2
傅里叶变换	7	5	4
线性灰度增强	1	1	1
直方图增强	2	2	2
Krisch 算子分割	7	5	5
领域平均	1	1	1
图像线性复原	5	3	2
二值形态学运算	2	2	2
图像压缩编码	6	5	4

在对实际程序进行循环的分析时,跟之前的分类相比仍出现一些差异,首先在对循环并行分类时,程序中出现的 while 循环由于其条件的特殊性,无法对这类程序进行划分,导致在对程序分析时,未能分析出其中的加速模式.其次在对 for 循环分析时,有的循环中循环本身的边界的不确定或者由于程序中的依赖关系较为复杂,所以这类程序虽然具有并行性,但是对其进行加速代价较大,因此对这类程序进行加速显然是不明智的.此时,自动分析时遇到此类函数则选择跳过.

4.2.3 结果分析

针对提出的图像处理程序的并行性分类方法进行了初步的实现,先是发现 LLVM 完全契合对程序的分析,其次通过 LLVM 编译工具实现了对循环中程序的分类,最后采用了典型的图像处理的核心算法对所设计的分类工具进行了验证,达到了预期的效果.

5 结束语

本文从研究图像处理类应用程序角度出发,发现程序中蕴含的并行性,研究了主要的图像算法和其中的并行性特征,对于混合架构进行了研究分析,

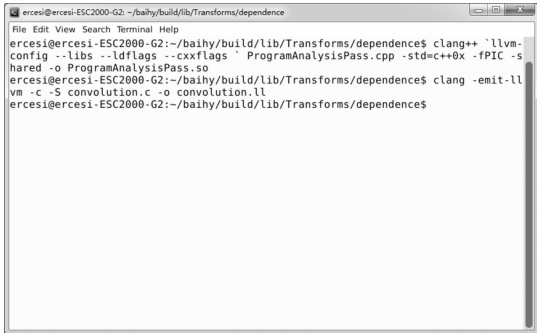


图 8 程序中间格式文件

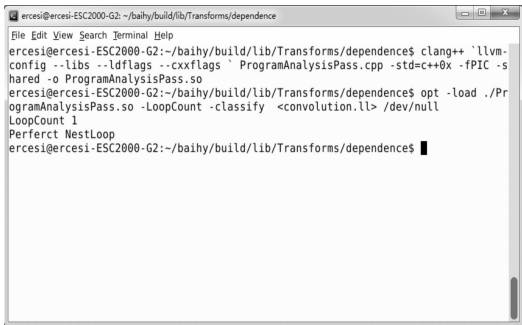


图 9 根据中间格式文件进行循环统计和检测分类

通过对程序的依赖关系进行分析,进而提取了程序中的循环并对循环进行了分析,提出了一种图像处理类并行性分类的方案,能够帮助编程人员解决如何使用混合架构来实现对图像处理类应用程序的加速,并采用编译器架构对分类进行了初步的实现。

参考文献:

- [1] 代媛. 基于 GPU 并行计算的图像压缩技术研究[D]. 杨凌:西北农林科技大学,2014.
- [2] Silva B D, An B, DHollander E H, et al. Comparing and combining GPU and FPGA accelerators in an image processing context[C]// International Conference on Field Programmable Logic and Applications. IEEE, 2013:1-4.
- [3] 张玮雄, 刘建霞. 数字图像处理技术的发展现状及趋势[J]. 科学之友, 2012,33(6):153.
- [4] 龚志远. 基于图像分析技术的轴承故障诊断[J]. 煤矿机械,2005 ,26(11) :171-173.
- [5] 郭启军,潘炼,钱新恩. 基于图象识别的高抛光透射面

缺陷检测[J]. 计算技术与自动化, 2008, 27 (4) : 95-97.

- [6] 许琳娟,曹文洪,刘春晶,等. 图像处理技术在推移质运动颗粒参数提取中的应用[J]. 长江科学院学报, 2017, 34 (1) :1-5.
- [7] Gonzalez R C, Woods R E. 冈萨雷斯数字图像处理[M]. 北京,电子工业出版社,2003.

作者简介:

王时雨 男,(1984-),博士研究生. 研究方向为微处理器设计、雷达成像、视频处理.

E-mail: onion0709@mail. nwpu. edu. cn.

张盛兵 男,(1968-),博士,教授. 研究方向为嵌入式微处理器的体系结构与微处理器芯片.

安建峰 男,(1977-),博士,副教授. 研究方向为高性能计算机体系结构、嵌入式系统软硬件协同设计.

黄小平 男,(1979-),博士,讲师. 研究方向为计算机体系结构.

白弘骥 男,(1992-),硕士. 研究方向为计算机体系结构.

(上接第 65 页)

参考文献:

- [1] ARMBRUST M, FOX A, GRIFFITH R, et al. A view of cloud computing [J]. Communications of the ACM, 2010, 53(4): 50-58.
- [2] Wolpert D H, Macready W G. No free lunch theorems for optimization[J]. IEEE Transactions on Evolutionary Communication, 1997,1(1): 67-82.
- [3] 刘漫丹. 文化基因算法(Memetic Algorithm)研究进展[J]. 自动化技术与应用, 2007, 26(11):1-4.
- [4] 蔡斌. 基于文化基因算法的车间作业调度理论研究及实践[D]. 重庆:重庆大学, 2012.
- [5] 潘铁文. 基于文化基因算法的图像检索研究[D]. 哈尔滨: 哈尔滨工程大学,2010.
- [6] 龙志翔. 多目标 Memetic 算法在网格作业调度中的应用研究[D]. 武汉:武汉理工大学, 2011.
- [7] 黄婷婷, 梁意文. 云工作流任务调度的模拟退火遗传改进算法[J]. 微电子学与计算机, 2016, 33(1): 41-46.

[8] 黄伟建,郭芳. 基于烟花算法的云计算多目标任务调度[J]. 计算机应用研究, 2017 , 34 (6) :1718-1720.

[9] 黄超,胡德敏,余星. 多目标遗传算法在云计算任务调度中的应用[J]. 信息技术, 2014,38(5):130-134.

[10] 刘之家,张体荣,谢雄程. 基于云计算的"用户期待"任务调度算法的研究[J]. 大众科技, 2011,13(4): 75-77.

[11] Chuan-Kang Ting, Cheng-Feng Ko, Chih-Hui Huang. Selecting survivors in genetic algorithm using tabu search strategies [J]. Memetic Comp, 2009 (1): 191-203.

作者简介:

郝晓弘 男,(1960-),硕士,教授,博士生导师. 研究方向为迭代学习控制和智能信息处理.

韩于芳(通讯作者) 女,(1991-),硕士. 研究方向为云任务调度理论及应用. E-mail: cicyyufang@163. com.

晏燕 女,(1980-),博士研究生,副教授. 研究方向为多媒体信息安全、位置隐私保护.