基于区域图数据流分析的通信优化算法*

钟洪涛, 舒继武+, 温冬婵, 郑纬民

(清华大学 计算机科学与技术系,北京 100084)

A Communication Optimization Algorithm Based on Data-Flow Analysis of Region Graph

ZHONG Hong-Tao, SHU Ji-Wu+, WEN Dong-Chan, ZHENG Wei-Min

(Department of Computer Science and Techonlogy, Tsinghua University, Beijing 100084, China)

+ Corresponding author: Phn: 86-10-62785592, Fax: 86-10-62771138, E-mail: shujw@tsinghua.edu.cn

http://www.tsinghua.edu.cn

Received 2001-12-05; Accepted 2002-04-10

Zhong HT, Shu JW, Wen DC, Zheng WM. A communication optimization algorithm based on data-flow analysis of region graph. *Journal of Software*, 2003,14(2):175~182.

Abstract: Reducing communication overhead is extremely important for parallelizing compiler to generate efficient codes for distributed-memory systems. In this paper, a redundant parallel execution model (RPEM) is proposed as an execution model for target programs optimized by the new algorithm. The region graph is introduced, and an effective algorithm is proposed to maximize the regions in the region graph. A region-based data-flow analysis algorithm is proposed to perform communication optimization. The overhead of data-flow analysis can be reduced by performing analysis on the maximized region graph. The coarse grain analysis also helps to communication lift up and aggregation. This communication optimization algorithm is able to perform inter-loop and inter-procedure analysis. Experimental results show that this algorithm is effective in reducing both communication volume and number of messages in programs with a large communication amount.

Key words: communication optimization; data-flow analysis; region graph; distributed memory system

摘 要: 减少通信开销对于并行化编译器生成高效的分布代码是非常重要的.首先提出了一个冗余并行执行模型(RPEM)作为通信优化算法生成的目标程序的执行模型,之后给出了区域图的概念和区域最大化算法,在最大化区域图的基础上进行数据流分析可以增大数据流分析粒度,提高分析的效率,同时也有助于通信的提前与合并.最后提出了一种基于区域图数据流分析的通信优化算法.该算法能够进行跨循环、跨过程的数据流分析,提高分析的精度,改善通信优化效果.实验结果表明,该算法对于通信量较大的程序能够有效地减少通信的次数和通信量,具有良好的可扩展性.

关键词: 通信优化;数据流分析;区域图;分布存储系统

中图法分类号: TP301 **文献标识码**: A

* Supported by the National Natural Science Foundation of China under Grant No.60103019 (国家自然科学基金)

第一作者简介: 钟洪涛(1978－),男,山东济南人,硕士生,主要研究领域为并行编译.

在分布存储的计算机系统中,数据的远程访问一般通过消息通信来完成,通信的开销使数据的远程访问时间远远高于本地访问的时间.因此,针对分布系统的并行编译器的研究必须充分考虑到分布存储的特点,在生成消息通信语句时对通信进行优化,控制通信量和通信的次数,以减少程序的运行时间.

目前在并行编译中使用的通信优化技术有:消息向量化(message vectorization)[1,2]、消息联合(message coalescing)[3]、消息聚合(message aggregation)[3]等.其中使用得最为普遍的通信优化技术是消息向量化,即将循环内的针对单个数组元素的通信提到循环外以形成一个消息,从而减少消息发送和接收的系统开销.以如图 1 所示的程序段为例,假定数组 A,B 和 C 都是(0:127,0:127)的二维数组,共有 4 个处理器,处理器号从 0 到 3.数组的分布方式为(*,BLOCK).为了简化起见,在并行代码中仅给出消息接收语句,略去与之对应的消息发送语句.如果不进行任何优化,则如图 1(b)所示(p 为当前处理器号),所有非本地的数组元素在访问前从其他处理器获取.每一条通信语句仅进行一个数组元素的通信,导致数量巨大的短消息.消息向量化能够通过编译器分析确定数组的访问之间不存在跨循环的相关,因此可以将它们提前到循环外,组成一个消息进行发送和接收.经过消息向量化之后的代码如图 2 所示,127 次长度为 1 的消息传递通过向量化后形成一个长度为 127 的消息.

```
DO J = 0,127
      DO I = 0,126
            A(I,J) = A(I,J)+B(I,J−1)
      ENDDO
ENDDO

DO J = 0,127
      DO I = 1,127
        C(I,J)=B(I,J−2)+B(I,J−1)+C(I,J−1)
      ENDDO
ENDDO
```

(a) Original serial code

```
DO J = p*32, p*32+31
 DO I = 0,126
       IF (J.EQ.p*32) THEN
            recv(p−1,B,1)
       ENDIF
       A(I,J) = A(I,J)+B(I, J−1)
 ENDDO
ENDDO
DO J = p*32, 32*p+31
 DO I = 1,127
       IF (J.EQ.p*32) THEN
       recv(p−1,B,1)
       recv(p−1,C,1)
       ENDIF
       C(I,J) = B(I,J−1)+C(I,J−1)
 ENDDO
ENDDO
```

(b) Parallelized code

Fig.1 Program before optimization

图 1 优化前的程序

```
recv(p−1,B,127)
DO J = p*32 +1,p*32 +32
 DO I = 0,126
       A(I,J) = A(I,J)+B(I,J−1)
 ENDDO
ENDDO
recv(p−1,B,127)
recv(p−1,C,127)
DO J = p*32 +1,32*p +32
 DO I = 1,127
       C(I,J)=B(I,J−1)+C(I,J−1)
 ENDDO
ENDDO
```

(a) Codes after message vectorization

```
recv(p−1,B+C,255)
DO J= p*32 +1,p*32+32
 DO I = 0,126
       A(I,J)=A(I,J)+B(I,J−1)
 ENDDO
ENDDO
DO J = p*32 +1,32*p+32
 DO I =1,127
       C(I,J)= B(I,J−1)+C(I,J−1)
 ENDDO
ENDDO
```

(b) Global message coalescing and aggregation

Fig.2 Program after optimization

图 2 优化后的程序

消息联合和消息聚合等进一步优化的技术常常是在循环间和过程间进行的.通过进一步的循环间的分析可以看出,第 2 个循环所使用的数组 B 和 C 在第 1 个循环内没有进行修改,因此它们可以提前到第 1 个循环前进行.对于第 0 个处理器,第 1 个循环需要接收的数据为 B(0:126,127),第 2 个循环需要接收的数据为 B(1:127,127)和 C(1:127,127).通过消息联合可以对 B 进行合并,形成 B(0:127,127),这样,两个长度为 127 的消息变成 1 个长度为 128 的消息,既减少了消息量又删除了重叠部分的冗余通信.更进一步地,可以通过消息聚合将 B 消息和 C 消息合并,从而形成 1 个长度为 255 的消息.这样,通过循环间的消息联合和聚合,将 3 个长度为 127 的消息进一步合并为 1 个长度为 255 的消息,最终的代码如图 2(b)所示.这是个循环间的例子,该分析过程可以跨过程进行,从而进行整个程序范围的全局优化.需要注意的是,大范围的消息联合和聚合都需要在通信向前提的基础上

进行,因此需要进行全局的数据流分析,以确定可以向前提的消息.

现有的算法一般都在语句一级进行,使得通信优化的过程非常耗时.对于一个交互式的并行编译系统来说,通信优化的开销是一个非常重要的问题,因为用户需要经常与编译器交互,如果用户每次交互以后都要进行长时间的通信优化则难以令人接受.此外,现有的数据流分析一般在过程内进行,没有进行过程间的分析.

本文提出了一种高效的通信优化算法.我们通过将控制流图上的节点按照一定的规则合并,形成区域图,并将区域图中的节点按照一定的规则合并得到最大化区域图,使得数据流分析的粒度尽量增大,以提高分析的效率.同时我们提出的算法还可以进行过程间的分析.本文第 1 节介绍控制流图与区间分析这两个数据流分析中常用的基本概念.第 2 节给出我们的通信优化算法.首先提出了冗余并行执行模型(RPEM),优化生成的目标程序都将遵循这个模型执行;接着详细介绍了区域图,给出最大化区域图的概念以及一个高效的区域最大化算法,在最大化区域图上进行数据流分析可以有效地减少分析的开销,并且有利于消息的提前与合并;然后给出了数据流分析的数据流方程和过程间数据流分析的框架.第 3 节是实验结果,可以看出,我们的算法对于一些程序是非常有效的,具有良好的扩展性.

1 基本概念

1.1 控制流图(CFG)

大部分基于数据流分析的通信优化都是以控制流图为基础的.一个程序的控制流图可以表示为一个有向图 $G=\langle V,E\rangle$.V 是控制流图的节点集合,节点 $v\in V$ 可以是一条语句或基本块;E 是有向边的集合,E 中的元素 $e\in E$ 是程序的控制边.两个节点 s 与 t 分别代表了程序的开始与结束节点,可以认为,任意 $v\in V$ 都位于一条从 s 开始到 t 结束的路径上.

控制流图的一个重要性质是它的可规约性(reducibility).如果程序中不存在从循环外跳到循环内的 goto 语句,那么这个程序对应的控制流图是可规约的(reducible),反之这个控制流图就是不可规约的(irreducible).Knuth 的研究表明,绝大部分的 Fortran 程序的控制流图都是可规约的[4].此外,不可规约流图可以经过适当的变换转化为可规约流图[5].本文假定所有的分析都是针对可规约流图进行的.

1.2 区间分析(IntervalAnalysis)

区间分析是一种在控制流图上进行的数据流分析方法,它最早由 Allen 与 Cocke[6]提出.区间分析可以有效地处理可规约流图上的数据流问题,是程序优化中非常重要的分析方法.对结构化的程序来说,一个区间一般对应于一个循环,文献[6]给出了将控制流图划分为互不相交的区间的算法.

区间分析包括收缩(contraction)与展开(expansion)两个阶段.收缩阶段主要收集每个区间内生成和注销的数据集合.然后这个区间收缩为一个节点,这个节点包括了在区间内收集到的信息.这是一个递归的过程,当控制流图收缩到不再包含循环时停止.收缩阶段的主要目的是防止每个区间内的信息影响全局的数据流分析,使迭代的收敛速度变慢.

在展开阶段,节点根据与收缩相反的顺序依次展开,同时计算每个节点包含的区间内的数据流信息,通信集合也在这一个阶段计算.图 3 给出了区间分析的一般过程.

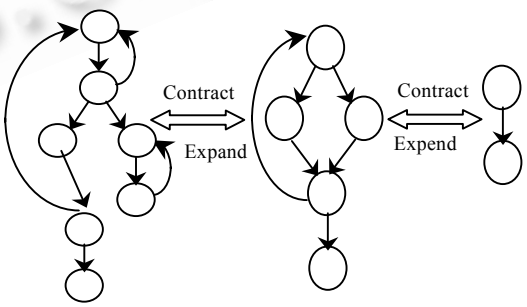

Fig.3 Interval analysis

图 3 区间分析

2 基于区域图数据流分析通信优化算法

我们假定在进行通信优化之前编译器已经完成了数据与计算的划分工作,也就是说,已经将共享数据划分到各个处理器,同时将循环的迭代空间划分到了各个处理器上.

2.1 冗余并行执行模型RPEM

由本文提出的算法优化生成的目标程序遵循我们提出的冗余并行执行模型 RPEM(redundant parallel execution model).在这个模型中,程序的执行可以分为多个阶段.每个阶段可能是 3 种类型之一:串行阶段、并行阶段或流水执行阶段.在串行阶段,程序将在所有的处理器上冗余执行;在并行阶段,程序在多个处理器上并行执行;在流水执行阶段,任何一个时刻都只有一个处理器在执行,多个处理器将以流水的方式执行一个循环的迭代空间.之所以存在一个流水阶段,是因为程序中存在一些循环由于跨迭代的数据相关而无法并行执行,但是循环中需要访问的数据考虑到其他循环的并行化而被分布到多个节点上.如果这些无法并行的循环采用在某个节点执行或者在所有节点冗余执行的方式,都会因为需要访问别的节点上的数据而带来大量通信.如果将这些循环的迭代空间根据数据划分的结果划分到各个节点上,则可以有效地减少通信.图 4 形象地给出了程序在这个模型下执行的过程.在这个模型中,通信只在不同的阶段之间进行,同一个阶段内不进行通信.将每个阶段开始前的通信称为前通信,它的作用是从数据的拥有者处获得数据,将每个阶段结束后的通信称为后通信,它的作用是将在本处理机上修改过的数据发送给数据的拥有者,保证数据的一致性.

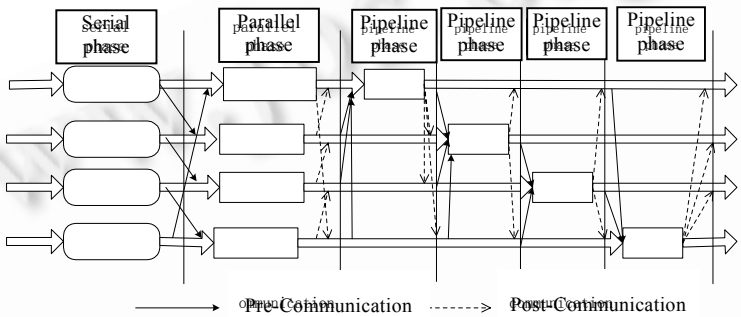

Fig.4 Program execution following RPEM

图 4 遵循 RPEM 模型的程序执行

在这个模型中,由于串行阶段的冗余执行,串行阶段被修改的数据在每个处理器上都是有效的,因此串行阶段无须进行后通信.

2.2 区域图与最大化区域图

传统的数据流分析一般以语句或者基本块为单位进行,由于数据流分析需要反复迭代,这种细粒度的数据流分析是非常耗时的.此外,细粒度的数据流分析不利于进行消息前提、消息合并、消息向量化等通信优化操作,因为这些通信优化的进行需要知道大范围的数据流信息.因此,我们需要增大通信优化的分析粒度,为此提出了区域图以及最大化区域图的概念.

我们首先对程序中的循环、过程以及过程调用根据计算划分的结果分为以下几类:

Ld: 划分循环集.在进行计算划分时,如果循环的最外层迭代空间被划分到各节点上并行执行,称该循环为划分循环.由划分循环组成的集合称为划分循环集.

Lp: 流水循环集.在进行计算划分时,如果循环的最外层迭代空间被划分到各节点上以流水方式执行,称该循环为流水循环.由流水循环组成的集合称为流水循环集.

CS: 复合调用集.如果一个过程或者该过程直接和间接调用的过程中存在划分循环,则该过程称为复合过程.对于一个过程调用语句,如果被调用的过程是复合过程,则这个调用就是复合调用.复合的意义是,在数据流分析中,这个节点需要展开作进一步的分析.由复合调用组成的集合称为复合调用集.

Lc: 复合循环集.如果循环内存在划分循环、流水循环、复合调用,则这个循环被称为复合循环.由复合循环组成的集合称为复合循环集.

将控制流图中所有的最外层循环收缩为一个节点,得到的有向无环图就是一个区域图.下面给出区域图的形式化定义.

定义 1. 区域图 $RG(S)$.对于有单一入口和出口的程序段 S,$RG(S)$为具有惟一根结点的有向无环图,可以用七元组$\langle V_d,V_p,V_c,V_{cs},V_s,E,root\rangle$来表示.其中 V_d 由 L^d 中的循环构成;V_p 由 L^p 中的循环构成;结点集 V_c 由 L^c 中的循环构成;结点集 V_{cs} 由 CS 中的过程调用语句构成;结点集 V_s 为串行程序段组成的结点集,每一个结点为仅有一个入口和一个出口的程序段,该程序段中不包含 L^d,L^c,L^p 以及 CS 中的循环和过程调用语句.E 为结点之间的控制边形成的边集,*root* 为入口语句所在的结点.

在数据流分析中,对于串行的程序段,我们只需要知道入口与出口处的数据流信息,并不需要知道节点内的数据流信息,因此在区域图中,如果存在一个单入口单出口的子图 $SG(S)$,其中的每个节点都是 V_s,那么可以将这个子图合并为一个 V_s 节点.这样可以增大数据流分析的粒度,而且在展开阶段也无须将这些合并的节点展开,降低了分析的开销.为此,我们提出了最大化区域图的概念:

定义 2. 最大化区域图 $MRG(S)$.$MRG(S)$是区域图 $RG(S)$的一个特例.其中所有的 V_s 结点不能通过合并生成新的区域图.此时,图中的每个区域都是最大化的.

需要指出的是,在由区域图构造最大化区域图的时候,只有 V_s 节点可以被合并,V_d,V_p,V_c,V_{cs} 都不能合并,因为我们需要知道每个节点详细的数据流信息.图 5 给出了一个由控制流图得到区域图,再将区域图的区域最大化,从而得到最大化区域图的例子.

算法 1. 区域的区域最大化算法.

(1) 给定一个区域图 $RG(S)$,根据深度优先搜索的出栈逆序得到节点拓扑序列 A.

(2) 将拓扑序列 A 从每个 V_d,V_p,V_c,V_{cs} 处断开,这时候,序列 A 被分割为若干个互不相交的子序列.

(3) 对于序列 A 中的每个 V_s 节点,如果该节点存在一条来自另外一个子序列的入边,则把这个节点标记为这个子序列的一个入口节点;如果该节点存在一条指向另外一个子序列的出边,则把这个节点标记为这个子序列的一个出口节点.

(4) 对每个子序列中除了第 1 个节点之外的入口节点,在这个节点之前将子序列断开.

(5) 对每个子序列中除了最后一个节点之外的出口节点,在这个节点之后将子序列断开.

(6) 反复执行(3)~(5),直到每个子序列只有一个入口节点与一个出口节点.

(7) 将每个子序列中的 V_s 节点合并为一个新的 V_s 节点,得到的图就是最大化的区域图 $MRG(S)$.

下面以图 5 中的区域图为例,解释最大化区域图的构造方法:

(1) 得到区域图的深度优先拓扑序 A:{12,3,7,8,9,*a*,456,*bc*,*d*,*e*,*f*,*g*};

(2) 从 V_d,V_c,V_{cs} 处断开,得到子序列{12,3,7,8,9,*a*},{456},{*bc*},{*d*},{*e*},{*f*,*g*};

(3) 节点 3 是一个出口节点,第 1 个子序列从 3 之后断开;*g* 是一个入口节点,最后一个子序列从 *g* 之前断开.得到新的子序列为{12,3},{7,8,9,*a*},{456},{*bc*},{*d*},{*e*},{*f*},{*g*}.

(4) 此时每个子序列都只有一个入口和一个出口,将每个子序列里的节点合并,得到最大化区域图.

这个算法充分利用了深度优先拓扑序的特性,降低了算法复杂性.从上面的例子可以看出,虽然该算法需要迭代,但是对于大多数实际程序形成的区域图来说,边数 $E<<V(V-1)$,计算在很少的几次迭代后就可以结束.

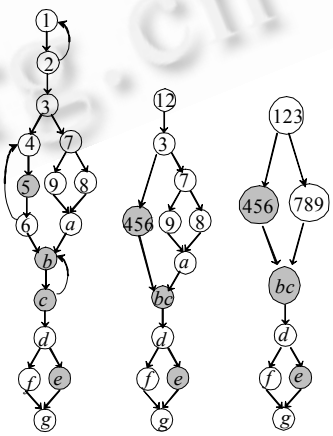

Fig.5 Construction and maximization of region graph

图 5 区域图的构造以及最大化

算法正确性证明:

引理 1. 设 $ND(v)$是树 T 中每个节点 v 的子节点总数.如果树 T 共有 V 个节点,并且节点按照深度优先拓扑序从 1~N 编号,那么节点 v 存在到节点 w 的路径的充要条件是 $v\leq w<v+ND(v)$.

证明:见文献[7]. □

引理 2. 在一个有向无环图 G 中,如果 S 是 G 的一个单入口、单出口的子图,那么 S 中的节点的深度优先拓

扑序是连续的.

证明:将 G 中的节点按深度优先拓扑序标号.设 T 是 G 的深度优先生成树,根据深度优先拓扑序的定义,T 与 G 中节点的拓扑序是一样的.设 a,b 分别是 S 的入口和出口.对于 S 中的任意一个节点 v,根据拓扑序的定义,显然有 $a<v<b$.下面用反证法证明 S 中的节点的深度优先拓扑序是连续的.假设存在 $a<w<b$,且 w 不属于 S.因为 a 是从 root 到 b 的必经节点,所以在 T 中存在从 a 到 b 的路径,由引理 1,$a\le b<a+ND(a)$,所以 $a<w<b<a+ND(a)$,亦即存在从 a 到 w 的路径.因为 b 是 a 的后必经节点,从 a 到 S 的任意一条路径都必须经过 b,所以 a 到 w 的路径经过 b.由拓扑序的定义 $b<w$,矛盾.由上可见,S 中的节点的拓扑序都在 a,b 之间.且 a,b 之间没有其他节点,所以 S 的深度优先拓扑序是连续的. □

定理 1. 算法 1 构造的是最大化区域图.

证明:显然,在算法 1 中合并的都是单入口单出口的节点集,所以合并的结构仍然是区域图.如果算法 1 得到的结果中仍然存在可以合并的单入口单出口的子图,由引理 2,该图中的节点的深度优先拓扑序应该是连续的.因此在算法中这些节点不可能被分割.所以算法 1 构造的是最大化区域图. □

2.3 数据流分析

对于复杂的程序来说,数据流分析的开销是很大的,最近出现了一些对高效数据流分析算法的研究[8,9].我们进行的数据流分析采用了区间分析的思想,比传统的区间分析更高效.在收缩阶段,传统的区间分析算法通过递归将程序中每个循环收缩为一个节点,当所有外层循环都收缩为一个节点后得到的就是一个区域图.我们的分析算法在这个基础上进一步满足一定条件的节点合并得到最大化区域图.在展开阶段,我们只对复合循环节点与复合调用进行展开分析,而对串行节点、并行节点、流水节点都不再展开,提高了分析的效率.下面我们来定义在数据流分析中需要用到的集合,并给出计算的数据流方程.

2.3.1 收缩阶段

$USE(S,p)=\{v|\ v$ 为在处理器 p 上的 S 中可能向上暴露引用的变量,而且 $owner(v)\neq p\}$,

$MOD(S,p)=\{v|\ v$ 为在处理器 p 上的 S 中可能定值的变量$\}$,

$WRI(S,p)=\{v|\ v$ 为在处理器 p 上的 S 中必定定值的变量$\}$.

以上 3 个集合将在分析的收缩阶段计算.在传统的数据流分析中对这些集合的计算已经有了很多研究,本文不再详述.

2.3.2 展开阶段

根据收缩阶段得到的 USE,MOD,WRI 的信息,可以在展开阶段计算出能够提前的消息通信.定义:

$FWD_IN(S,p)=\{v|$在处理器 p 上 v 可以提前到 S 之前接收$\}$,

$FWD_OUT(S,p)=\{v|$在处理器 p 上 v 可以提前到 S 之后接收$\}$.

下面给出计算这两个集合的数据流方程:

$$FWD\_OUT(S,p)=\cap_{s'\in succ(s)}FWD\_IN(S',p),$$

$$FWD\_OUT(S,p)=\varnothing \text{ if } S \text{ 为出口结点},$$

$$FWD\_IN(S,p)=FWD\_OUT(S,p)-(\cup_{p'\in P}MOD(S,p'))\cup USE(S,p) \text{ if } S\in V_c,V_{cs},V_d,$$

$$FWD\_IN(S,p)=FWD\_OUT(S,p)-MOD(S,p)\cup USE(S,p) \text{ if } S\in V_s,V_p.$$

为了减少通信量,我们需要进行数据的有效性分析,如果数据在计算前是有效的,那么就不需要通过远程访问来获取数据.下面定义有效数据的集合:

$VALID_IN(S,p)=\{v|$在处理器 p 上变量 v 在 S 之前上有效$\}$,

$VALID_OUT(S,p)=\{v|$在处理器 p 上变量 v 在 S 之后有效$\}$.

此外,我们定义:$RECV(S,p)=\{v\ |$处理器 p 在 S 之前从拥有者处理器接收变量 $v\}$.

为了计算这些集合,需要解以下的数据流方程:

$$VALID\_IN(S,p)=\cap_{s'\in prev(s)}VALID\_OUT(S',p)-(\cup_{p'\in P,p>p,S''\in prev(s),S''\in Vp}MOD(S,p')) \text{if } s\notin V_p,$$

$$VALID\_IN(S,p)=\cap_{s'\in prev(s)}VALID\_OUT(S',p)-(\cup_{p'\in P,p>p,S''\in prev(s),S''\in Vp}MOD(S,p'))-\int_{p'\in P,p'>p}MOD(S,p') \text{ if } s\in V_p,$$

$VALID\_OUT(S,p)=VALID\_IN(S,p)\cup RECV(S,p)\cup WRI(S,p)$,if $s\in V_s,V_p$,

$VALID\_OUT(S,p)=VALID\_IN(S,p)\cup RECV(S,p)\cup WRI(S,p)-(\cup_{p'\in P,p'\neq p}MOD(S,p'))$ if $S\notin V_s,V_p$,

$RECV(S,p)=USE(S,p)-VALID\_IN(s,p)$,if S 不支配其所有后继结点,

$RECV(S,p)=FWD\_IN(S,p)-VALID\_IN(s,p)$, if S 支配其所有后继结点.

在计算出 RECV 集合之后,我们就可以根据它,在每个代码中插入前通信的语句.对于后通信,它的目的是在每个阶段完成后将在非拥有者上赋值的数据发送给数据的拥有者,保证数据的一致性.但是,如果在后面程序的执行中其他程序不再使用本次计算的定值,那么可以将这次后通信去掉.为此,我们定义以下的数据集合:

$LIVE\_IN(S,p)=\{v|v$ 在程序段 S 前可能在处理器 p 上是活跃的$\}$,

$LIVE\_OUT(S,p)=\{v|v$ 在程序段 S 后可能在处理器 p 上是活跃的$\}$,

$SEND(S,p)=\{v|$在处理器 p 上需要在划分循环 S 后发送 v 到拥有者处理器$\}$,

$LIVE\_OUT(S,p)=\cup_{s'\in succ(s)}LIVE\_IN(S',p)-(\cup_{p'\in P,p'<p,S''\in succ(s),s''\in Vp}LIVE\_IN(S'',p'))$, if $S\notin V_p$,

$LIVE\_OUT(S,p)=\cup_{s'\in succ(s)}LIVE\_IN(S',p)-(\cup_{p'\in P,p'<p,S''\in succ(s),s''\in Vp}WRI(S'',p'))-(\cup_{p'\in P,p>p}WRI(S,p'))$, if $S\in V_p$,

$LIVE\_IN(S,p)=LIVE\_OUT(S,p)-(\cup_{p'\in P}WRI(S,p'))\cup USE(S,p)$ if $S\notin V_s,V_p$,

$LIVE\_IN(S,p)=LIVE\_OUT(S,p)-WRI(S,p')\cup USE(S,p)$ if $S\in V_s,V_p$,

对于 $S\in V_d,V_p$,$SEND(S,p)=MOD(S,p)\cap(\cup_{p'\in P,p'\neq p}LIVE\_OUT(S,p'))$.

2.4 通信优化算法

综合以上的数据流方程,我们可以得到一个通信优化算法.与区间分析类似,算法分为收缩与展开两个阶段,同时采用与文献[10]类似的数据流分析框架.但是我们的算法是基于最大化区域图的,因此更加高效.在展开阶段完成之后,我们可以根据计算得出的 Send 与 Recv 集合在程序中插入消息通信语句.

对于过程间的数据流分析,我们采用了过程克隆的技术[11],根据调用点的 VALID_IN 集合的不同,克隆出多个被调过程,并把数据流信息传播到过程之中.

算法 2. 通信优化算法.

(1) 初始阶段:对于程序中的每一个过程:

将不可归约流图转化为可归约流图 CFG;

将 CFG 中每个最外层循环收缩为一个节点形成区域图 RG;

对 RG 应用算法 1 生成最大化区域图 MRG.

(2) 收缩阶段:对于程序中的每一个过程,按照逆拓扑序:

从内向外计算 WRI,MOD 以及 USE 集合,在计算中递归地将循环收缩,得到收缩后节点的 WRI,MOD 以及 USE 集合.

(3) 展开阶段:对于程序中的每一个过程,按照拓扑序:

进行克隆分析,计算出过程入口的 VALID_IN 集合;

初始化程序出口的 VALID_OUT 以及 LIVE_OUT 集合;

对该过程最大化区域图中的节点 t,按逆拓扑序,根据前面给出的数据流方程计算后向数据流集合:FWD_OUT,FWD_IN,LIVE_OUT,LIVE_IN,SEND;

对该过程 MGR 中的节点 t,按拓扑序,根据前面给出的数据流方程计算前向数据流集合: VALID_IN,RECV;

对于过程中的每一个调用点,将 VALID_IN 信息经过过滤后映射到被调用过程.

(4) 根据 RECV 集合和 SEND 集合插入相应的通信语句.

3 实验结果

我们应用该通信优化算法对程序进行了优化,这几个程序分别选自不同的 Benchmark,其基本信息见表 1.进行通信优化前后的消息个数与消息量见表 2.

Table 1 Basic information of benchmarks
表 1 测试程序的基本信息

Program	SWIN	TOMCATV	EP	TRFD
Source	SPEC92	SPEC92	NAS	PEFECT

Table 2 Messages before and after optimization
表 2 优化先后的消息数与消息量

Program	Tasks	Number of messages			Volume of messages (KB)		
		Original	Optimized	Reduced (%)	Original	Optimized	Reduced (%)
TOMCATV	2	40 928	42	99.9	1 015	336	66.9
	4	122 796	132	99.9	3 045	1 009	66.8
	8	286 580	336	99.9	7 104	2 353	66.9
SWIM	2	6 106	3 477	43.1	6 144	6 134	0.16
	4	18 318	10 434	43.4	18 432	18 401	0.17
	8	42 742	25 179	41.1	43 008	42 938	0.16
EP	2	2	2	0.0	0.088	0.088	0.0
	4	6	6	0.0	0.264	0.264	0.0
	8	14	14	0.0	0.616	0.616	0.0
TRFD	2	113	67	41.7	3 067	2 122	30.8
	4	342	198	42.1	9 507	6 588	30.7
	8	1 067	620	41.9	30 422	20 930	31.2

从表 2 可以看出,由于进行了跨循环的消息提前、消息聚合与消息合并,同时进行了过程间的数据有效性分析,对 TOMCATV, SWIM, TRFD 这 3 个程序,经过该算法优化后,消息的数量有了较大的减少.因为通信开销的很大一部分来自于消息的初始化,所以减少通信数量可以有效地减少通信时间.此外,对于 TOMCATV 与 TRFD,消息量也有很大的减少,原因是数据有效性分析减少了有效数据的重复通信.注意到,对于 SWIM 消息量减少不多,一方面是因为 SWIM 本身可以优化的空间有限,另一方面是因为我们在数据流分析中还作了一些保守的假定,限制了可以优化的范围.进一步的工作可以考虑进行更详细的分析来减少通信.我们的优化算法对 EP 没有什么优化效果,这是由于 EP 本身的并行性很高,通信量也不大,所有没有太大的优化空间.同时,从表 2 可以看出,随着任务数的增加,通过优化减少的通信比例基本不变.可见该算法具有良好的可扩展性.

References:

[1] Balasundaram V, Fox G, Kennedy K, Kremer U. An interactive environment for data partitioning and distribution. In: Charleston SC, ed. Proceedings of the 5th Distributed Memory Computing Conference. New York: ACM Press, 1990.

[2] Banerjee P, Chandy JA, Gupta M, Hodges IVEW, Holm JG, Lain A, Palermo DJ, Ramaswamy S, Su E. The PARADIGM compiler for distributed-memory multicomputers. IEEE Computer, 1995,28(10):37~47.

[3] Adve V, Mellor-Crummey J, Sethi A. An integer set framework for HPF analysis and code generation. Technical Report, TR97-275, Computer Science Department, Rice University. 1997.

[4] Knuth DE. An empirical study of FORTRAN programs. Software-Practice and Experience, 1971,1(2):105~134.

[5] Johnson SP, Ierotheou CS, Cross M. Automatic parallel code generation for message passing on distributed memory systems. Parallel Computing, 1996,22(2):227~258.

[6] Allen FE, Cocke J. A program data flow analysis procedure. Communications of the ACM, 1978,19(3):137~174.

[7] Tarjan RE. Finding dominators in directed graphs. SIAM Journal of Computing, 1974,3(1):62~89.

[8] Atkinson DC, Griswold WG. Implementation techniques for efficient data-flow analysis of large programs. In: Anneliese AA, Aniello C, eds. Proceedings of the IEEE International Conference on Software Maintenance. Italy: University of Sannio Press, 2001. 52~61.

[9] Kandemir M, Banerjee P, Choudhary A, *et al*. A global communication optimization technique based on data-flow analysis and linear algebra. ACM Transactions on Programming Languages and Systems (TOPLAS), 1999,21(6):1251~1297.

[10] Hall MW, Murphy BR, Amarasinghe SP, Liao S, Lam MS. Interprocedural analysis for parallelization. In: Huang CH, Sadayappan P, Banerjee U, eds. Proceedings of the 8th International Workshop on LCPC. Columbus, Ohio: Springer-Verlag, 1995. 61~80.

[11] Cooper KD, Hall MW, Kennedy K. A methodology for procedural cloning. Computer Languages, 1993,19(2):105~118.