二维非线性对流扩散方程求解程序的测试与优化

赵永刚[1],付立东[2],邓福岐[1]
(1.西安卫星测控中心喀什测控站,新疆 喀什 844000;
2. 西安科技大学 计算机系,陕西 西安 710054)

摘 要:在IA-64架构Itanium2处理器上,应用gprof和pfmon对二维非线性对流扩散方程求解程序源代码进行了性能测试。在分析给定程序的数据结构,子过程调用关系,重点子程序中循环体的迭代空间、数据空间、访问轨迹,输入输出数据量大小和程序结构等的基础上,应用子过程合并、循环变换、分支消除、循环顺序逆转、数组一维结构化为二维结构、输入参数给定等方法,改善了数据访问的时空局部性,程序性能有15%的提高。

关键词:程序性能优化;数据局部性;循环变换;数据变换
中图分类号:TP301 **文献标识码**:A **文章编号**:1673-629X(2009)07-0137-04

Performance Testing and Optimization for Solution of 2D Nonlinear Convection Dominated Diffusion Problem

ZHAO Yong-gang[1], FU Li-dong[2], DENG Fu-qi[1]
(1. Kashi TT & C Station, Xi'an Satellite Control Center, Kashi 844000, China;
2. Dept. of Computer, Xi'an University of Science and Technology, Xi'an 710054, China)

Abstract: Applying the tool named gprof and pfmon to get the program's measurement and analysis on IA-64. For improving the performance, the data structures, iteration-space, array-space, data accessing tracks in loop nests and the call tree of the given program is described. By applying some methods, such as merging subroutines, loop interchange, branch removing, loop skewing, inline function, for guessing the logical multi-dimensional array structures from the flat one-dimensional arrays and parameter constant to enhance program code's memory locality optimizations, performances achieved is 15 percent higher than the original program.

Key words: program performance optimization; data locality; loop transformations; data transformations

0 引 言

求解由实际问题数学物理建模得到的复杂偏微分方程是大规模、长时间的数值计算问题。为了提高问题求解效率,满足应用需求,除了研制高性能计算机,针对目标问题构造好的算法外,还要对高性能计算软件进行性能优化,以充分发挥机器的计算潜力。

1 二维非线性对流扩散方程求解程序模型

BNLAG2D程序用于行求解二维非线性对流扩散问题,所求解的非线性不稳定扩散方程为[1]:

$$Q\frac{\partial u}{\partial t} = B_1\frac{\partial^2 u}{\partial x^2} + B_2\frac{\partial^2 u}{\partial y^2} + B_3\frac{\partial u}{\partial x} + B_4\frac{\partial u}{\partial y} + eu + g$$

其中,

$$Q = f + a_1U^3/R,$$
$$B_1 = a_1U^3/R,$$
$$B_2 = a_2U^3/R,$$
$$B_3 = c_1\sin(2\pi x) + c_2,$$
$$B_4 = d_1\sin(2\pi y) + d_2,$$

f、a_0、a_1、a_2、c_1、c_2、d_1、d_2、e、g 为矩形域上由DIRICHLET边界条件定义的常数。离散格式为五点全隐式,采用非线性代数方程组解法器,包括牛顿局部线性化和稀疏线性方程组解法器(对角预条件QMRCGSTAB方法)进行求解。程序的主要计算量集中在用QMRCGSTAB法求解稀疏线性方程组。为便于测试,在程序中设定计算6个时间步,每个时间步中线性方程组求解的迭代次数为2500,即线性方程组求解的总迭代次数为15000次。

测试环境为IA-64系统,主频为1.4MHz的Itanium2。内存大小是8107MB[2]。各级cach情况如表1所示。

收稿日期:2008-10-11;**修回日期**:2008-12-23
作者简介:赵永刚(1976-),男,陕西岐山人,工程师,硕士,研究方向为高性能计算。

2 源代码分析

表2给出了应用工具gprof对程序进行测试的结果,可以看出子程序pqmrcgstab、mvmi1p2d、pr2d1p2d和prod1p2d是重点子程序,现主要对这些核心子程序中的数据存储访问进行分析[3]。在科学计算中,嵌套循环中的数组访问时间占整个程序执行时间的比重相当大,所以当前的局部性优化方法大都是针对嵌套循环中的数组访问进行的。

表1 Cache指标参数

Cache / 指标	L1I	L1D	L2	L3
取数周期	1	1	5	14
存数周期	0	3	7	7
CACH容量	16kB	16kB	262kB	1.5MB
CACH行大小	64B	64B	128B	128B
CACH行数	256	256	2048	12288

表2 程序主要过程/函数调用关系、调用次数与时间分布

index	%time	Self(s)	Child	Called	name
[5]	66.4	512.77	258.73	6	pqmrcgstab[5]
		173.86	0.00	29988/29988	mvmi1p2d[6]
		36.95	0.00	14994/14994	pr2d1p2d[7]
		33.52	0.00	15000/15006	prod1p2d[8]
		14.41	0.00	29988/29988	mvmb1p2d[9]
[6]	22.5	173.86	0.00	29988	mvmi1p2d[6]
[7]	4.8	36.95	0.00	14994	pr2d1p2d[7]
[8]	4.3	33.53	0.00	15006	prod1p2d[8]
[9]	1.9	14.41	0.00	29988	mvmb1p2d[9]
合计	99.9	771.52			
		772.29			

对BNLAG2D的核心子程序中嵌套循环的数组访问情况进行分析后发现,子程序中循环体都是2重循环,循环体迭代空间都是 $N*N$ 的正方形。以某个循环体中数组 C 为例,在图1中,每个小圆圈代表一个数组元素,数据空间是802*802正方形,C 数组的访问轨迹由灰色圆圈标出。从图1中可以看出,除过边界元素外,数组被访问的轨迹是连续的,说明数组访问具有良好的空间局部性。再对单个循环体中数组之间的运算作分析,可以看出大多是类似于内积的运算,都是按列访问的,每个数组元素只参加一次运算。

再分析一下核心子程序的输入/输出数据量的大小。程序中数组定义是双精度实数,一个数组元素占8B大小。以子程序mvmi1p2d(NDIM,R,CC,TT,U,NSTENCIL)为例,其中输入参数是数组CC,TT,U。数组CC,U大小都是802*802个双精度实数,TT大小是802*802*8个双精度实数,总的输入数据量大小是802*802*10个双精度实数,即约是51.456MB。输出数组R为802*802个双精度实数,即约是5.146MB。1级Cache行长为64B,一次可以装入8个数组元素。2,3级Cache行长为128B,一次可以装入16个数组元素。3级Cache的容量1.6MB,小于单个数组的大小,所以一个数组不能全部装入Cache,在一次循环体中数组访问时会发生自冲突,但不会引起失效,因为数组的每个元素在一次循环中只被访问一次,但在下一个循环体中要访问它时会出现容量失效。由于Cache采用组相联的映射方式,分析程序在一个循环体中访问的数组个数不会大于相联度,所以多个数组之间不会发生交叉冲突失效。

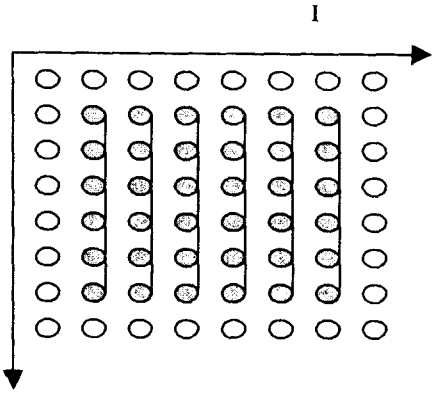

图1 程序BNLAG2D中数组C访问轨迹示意图

通过前面的分析,可以得出存储延迟主要是因为第一次访问的数组元素不在Cache中引起的,即强制失效引起的,但这种失效是不可避免的。但可以减少程序中由于多个循环体间数组时间局部性比较差引起的失效开销。所以优化重点在多个嵌套循环体中同一数组的重复使用方面。

程序BNLAG2D用gprof测试的结果如表2所示,从中可以看出,在执行时间占总时间接近100%的5个子程序中,pqmrcgstab调用了其他4个子程序,被调用子程序与pqmrcgstab自身的执行时间各约占一半,而这4个子程序内部没有调用其他子程序。所以要从2个方面来对代码优化,一是在5个子程序内各自分析多个循环体间数组的重用信息,二是在5个子程序间寻找数组的重用信息。

3 优化技术的应用

3.1 输入值嵌入(Input Value Embedding)

它的实质是:基于程序开发人员的知识,将运行时从文件读入、但其取值只有少数几种可能的性能关键数据转换为编译时静态可知的数据,使得编译器能够

进行针对性的优化。虽然这样做可能损失编译得到的可执行程序的通用性,但因为一次程序编译的时间远比程序执行的时间短,所以针对不同输入进行不同的编译是值得的。该方法给人们的启示是,向编译器提供的信息越具体,编译器越能够识别优化机会,进行优化。

对程序 BNLAG2D 来说,程序执行过程中从参数文件 bench.in 中读入的一些变量,在程序执行过程中值保持不变。于是,将这些参数在程序头文件中定义为常数,这样可以为编译器创造优化机会,如常量传播、预计算(Precomputation)等。这样处理后,程序执行时间缩短了 3 秒。由于一些参数已知,编译器对部分依赖这些参数值的表达式进行了预计算,程序运行时不需要计算,这样有些分支语句的跳转也就确定下来了。

3.2 消除某一类型分支

在子程序 pqmrcgstab 中,图 2 左边所示的代码段出现了 3 次,其中 2 次出现在大的循环体中,通过上面输入值嵌入技术,在编译时 NPREP 值已经知道,所以分支也已经确定下来。想把分支去掉,是为了它和其它循环体可以合并,改进访问数组的时间局部性来减少失效次数。

```
IF(NPREP.EQ.1)                    LQ=0
THEN                              DO 10 J=1,NY
LQ=0                              L=J*NXD+1
  DO 10 J=1,NY                    DO 10 I=1,NX
   L=J*NXD+1                      L=L+1
DO 10 I=1,NX                      LQ=LQ+1
  L=L+1                           R(L)=(2-NPREP)
LQ=LQ+1                           *NPREP*R(L)*
                                  FF(LQ)+(NPREP-1)
R(L)=R(L)*FF(LQ)                  *(NPREP-1)*R(L)
10 CONTINUE                       10 CONTINUE
ENDIF                             ENDIF
```

图 2　BNLAG2D 中消除某一类型分支示例代码

由程序可知,整型变量 NPREP 的取值是 0,1,2 三个离散点。将变量 NPREP 代入 R(L)的计算表达式,完全消除了分支,将控制相关转化成数据相关。对于由简单关系表达式作为条件的分支语句,若表达式中变量取值是有限个离散点,可以用含有该变量的表达式作用于控制执行语句组成新的执行语句,则可能消除分支判断。

3.3 减少子过程的调用次数

核心子程序 prod1p2d 在程序 BNLAG2D 中只被调用 3 次,而且过程实现体简单,语句少,本身只是完成向量内积计算,可以直接将实现语句放到过程被调用的位置,减少了子程序的调用开销,还可以和调用程序的其它循环体可以合并,改进访问数组的时间局部性来减少失效次数。

3.4 循环合并

前面讨论的方法有 2 种是为了优化以后可以采用循环合并来改进数据访问的时间局部性。在核心子程序 pqmrcgstab 中,由于循环体循环重数相同,循环索引变量上下界相同,可以将 8 个 2 重循环合并成 4 个。循环合并后程序执行时间缩短了 106.75 秒。

3.5 子过程合并

在子程序 pqmrcgstab 中,过程 mvmi1p2d 和 mvmb1p2d 总是被一起调用的且次序一致,入口和出口参数也一致,过程的实现体都是 4 分支选择条件下二重嵌套循环。通过改变循环索引变量上下界可以使两个独立的过程融合成一个过程,合并后减少了子程序的调用次数。原来对这 2 个子程序的调用开销也减少了一半。

子过程合并后,不仅减少了子程序调用开销,而且数组的访问也保持了连续性。合并前在子程序 mvmb1p2d 中,程序以 NX 个浮点数字长的跨距访问存储器中的这些数组,合并后程序顺序的访问一个 Cache 行元素,然后再访问下一行的元素,使得在一个 Cache 行被替换之前,能最大限度地利用行中的数据,减少了失效次数,改善了 Cache 的空间局部性[4]。优化后程序执行时间在循环合并后的基础上缩短了 15 秒。

3.6 改变循环变量的迭代顺序

在子程序 pqmrcgstab 中,通过消除某一类型分支、减少子过程的调用次数、循环合并 3 种代码优化调整后,存在两个嵌套循环体的访存模式是相同的,都访问了完整的 QV 数组。因为数组 QV 比 Cache 容量大,第 1 个循环体计算出来的数组 QV 的值在第 2 个循环使用前就替换出 Cache。可以使用迭代空间变换的方法来优化这两个嵌套间的重用。为利用第 1 个循环体中最近加载到 Cache 中的 QV 数组值,将第 2 个循环体中循环索引变量 J 与 I 的迭代顺序逆转。这样,在每个时间步中第 2 个循环体都能重用第 1 个循环体载入 Cache 中的数组 QV 部分元素,如图 3 所示,矩形框内的灰色圆圈表示可以重用的部分 QV 数组元素。改进后程序执行时间在前面优化的基础上时间有所减少[5]。

3.7 改变数组维数

将多重循环中的一维数组化成二维数组,使数组的下标可用循环变量来表示,减少数组是一维时下标

的计算量,同时也可增大 ILP[6],因为数组是二维时可以直接运算,不用像一维时要等到下标值计算出来才可参加表达式运算。

经测试,改进后程序执行时间在前面优化的基础上减少了将近 15 秒。

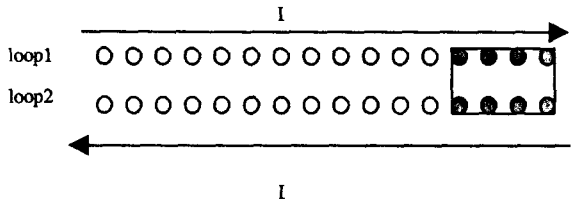

图 3 BNLAG2D 中一次迭代中数组 QV 的访问顺序

3.8 其他方法

使用 do 循环替代 goto 循环,可以使编译器进行循环展开[7]。goto 是无规则的跳转指令,容易造成流水线空转。在迭代中,应尽可能减少循环控制中的测试和分支。经测试,改进后程序执行时间有所减少。

对 BNLAG2D 的 5 个核心子程序应用以上方法进行代码优化后,程序执行时间有所缩短,性能有提高。上面提到的方法一般都是结合使用的,如消除分支,减少子过程的调用,就是为了循环体的合并,而有的合并后的循环体可以使下一个循环体采用循环逆转方法来进行程序优化[8]。

4 结束语

表 3 是优化后程序的测试结果,和表 2 相比较,可以看出,优化后子程序 pqmrcgstab 的执行时间明显缩短,从 772.29 秒减少到 630.63 秒,由于子程序 mvmi1p2d 和 mvmb1p2d 合并,子程序 pr2d1p2d 和 prod1p2d 的实现语句插入到 pqmrcgstab 中,程序的调用关系也发生了变化。

表 3 程序优化后主要过程/函数调用关系、次数与时间分布

index	%time	Self(s)	Children(s)	Called	name
[5]	72.95% 26.93%	460.00 170.83	170.83 0.00	6 29988 /29988	pqmrcgstab[5] mvmi1p2d[6]
合计	99.9%	630.43 /630.63			

表 4 给出了用 pfmon 工具测试的优化前后 16 个计数器的值[9],可以看出程序的时钟周期数、指令数、停顿周期数以及各种子延迟周期数都有减少,2,3 级 Cache 访问数,失效数也有所降低,说明数组访问的时间局部性得到了改善。

通过计算分析,CPI 值减小,浮点速率增大,利用率提高,说明程序总体性能有改善。

表 4 程序 BNLAG2D 优化前后计数器的值(10^10)

计数器名称	旧值	新值	计数器名称	旧值	新值
Cpu_cycles	121.9	111.1	Be_l1d_fpu_bubble_all	80.9	71.0
Ia64_inst_retired_this	136.2	127.9	L3_misses	2.3	2.0
Nops_retired	36.3	36.9	L2dtlb_misses	0.03	0.017
Back_end_bubble_all	91.8	83.7	L2_data_references_l2_all	54.0	41.8
L3_references	3.0	2.6	Be_flush_bubble_all	0.029	0.004
Be_exe_bubble_all	10.5	14.3	Be_rse_bubble_all	0.002	0.002
L2_misses	2.3	2.0	Back_end_bubble_fe	0.36	0.31
L2_references	54.1	41.8	Fp_ops_retired	46.18	52.8

参考文献:

[1] 张小峰,张艳霞,谢作涛. 一阶迎风差分格式求解非线性对流扩散方程的精度[J]. 武汉大学学报, 2003(5):56-60.

[2] 姜晓玲,任国林. 基于 IBM1350 机群的 Linpack 快速测试[J]. 计算机技术与发展,2007,17(3):18-22.

[3] 张 斌. 分布式程序的性能监视及其应用[J]. 微机发展(现更名:计算机技术与发展),2004,14(2):4-8.

[4] 张宇峰. 利用 Itanium2 的 PMU 部件开发程序性能分析工具[J]. 计算机技术与发展,2006,16:69-71.

[5] 曾丽芳,杨学军. 一种利用数据融合来提高局部性和减少伪共享的方法[J]. 计算机学报,2004(1):45-50.

[6] Jarp S. A Methodology for using the Itanium 2 Performance Counters for Bottleneck Analysis, Tech - Report HP Labs [EB/OL]. 2002 - 08 - 27. http://www.gelato.org/pdf/Performance-counters-final.pdf.

[7] Mosberger D, Eranian S. IA-64 linux kernel: Design and Implementation[M]. [s.l.]: Prentice Hall PTR, 2002.

[8] HP Labs Perfmon Project website [EB/OL]. 2005-02. http://www.hpl.hp.com/research/linux/perfmon/.

[9] Lingamneni A. PerfView - A Tool for Source - Level Performance Analysis on the Itanium Processor Family[EB/OL]. 2004-06. http://dynopt.dtc.umn.edu/documents/perfview-ananthplanb-report.pdf.