承 诺 书

我们仔细阅读了第二届“数学中国杯”数学建模网络挑战赛的竞赛规则。

我们完全明白，在竞赛开始后参赛队员不能以任何方式（包括电话、电子邮件、网上咨询等）与队外的任何人（包括指导教师）研究、讨论与赛题有关的问题。

我们知道，抄袭别人的成果是违反竞赛规则的，如果引用别人的成果或其他公开的资料（包括网上查到的资料），必须按照规定的参考文献的表述方式在正文引用处和参考文献中明确列出。

我们郑重承诺，严格遵守竞赛规则，以保证竞赛的公正、公平性。如有违反竞赛规则的行为，我们将受到严肃处理。

我们允许数学中国网站(www.madio.net)公布论文，以供网友之间学习交流，数学中国网站以非商业目的的论文交流不需要提前取得我们的同意。

我们的参赛报名号为： 1176

参赛队员 (签名) ：

队员 1： 谢旭

队员 2： 肖良

队员 3： 马琼敏

参赛队教练员 (签名)：

参赛队伍组别： 大学组

第二届“数学中国杯”数学建模网络挑战赛

编 号 专 用 页

参赛队伍的参赛号码：（请各个参赛队提前填写好）：

1176

竞赛统一编号（由竞赛组委会送至评委团前编号）：

竞赛评阅编号（由竞赛评委团评阅前进行编号）：

2009 年 第二届“数学中国杯”
数学建模网络挑战赛

题　　目：　串并转换算法的效率评价及算法优化

关 键 词：　改进的 Amdahl 加速比　　负载均衡　层次分析法
计算机模拟计算　　算法优化

摘　　要

本文在第一阶段模型的基础上主要做以下工作：

首先，根据评委给我们的提出的“考虑模型的转换能力和效率”的建议，我们参照了并行计算中最常用的效率评价模型——Amdahl 加速比计算模型，并对该模型只适用于固定规模的缺陷进行了改进，提出了适用于我们分解算法的改进 Amdahl 加速比 S，它具有自动适应问题规模的变化、计算简单的优点，把它作为评价分解算法转换效果的评价指标之一。

其次，我们给出了循环结构的计算量估计公式并根据估计出的计算量通过 0—1 规划给出了负载的平衡调度模型。在这里我们没有对顺序结构、选择结构进行深入的探讨，因为我们在挑战赛的第一阶段就通过计算机仿真测试验证了并行化算法对简单的顺序结构程序的运行时间没有明显的缩短，对选择结构的改善效果取决于具体的程序（随机性较大），同时一个程序的计算量并不主要取决于这两种结构。鉴于以上原因，我们主要考虑了循环结构。为了评价负载调度是否平衡，我们引入了负载均衡系数 m，用它来定量地描述负载调度的平衡程度，把它作为评价分解算法转换效果的另一个评价指标。

再次，为了便于计算和比较，我们将上述两个指标加权求和得到总的评价模型：

$$d = a*m + b*S \qquad (a,b \text{ 为权系数})$$

求解权系数必须尽可能客观、具有科学依据，为此，我们采用层次分析法（AHP 方法）求解 a,b。最后得到 $d = 0.17m + 0.83S$。

最后，我们对所建的模型进行了计算机模拟计算并根据计算结果对第一阶段的分解算法进行了相应优化。通过模拟计算我们得出了总体评价指标 d 与计算规模、总体评价指标 d 与程序中不可并行部分的比例 f 之间的关系曲线。通过这些关系曲线，我们有以下结论：当可并行执行的程序段的计算量是 10^7 量级时，双核的效能达到最大；不可并行部分的比例越低，双核的效能发挥的越明显。这两条结论恰恰为我们的分解算法的优化指明了方向。故基于计算机模拟结果，我们得出了分解算法优化方案——在执行分解算法之前加预处理。预处理一：在原先的分解算法的基础上对语句进行重排和无关性处理，降低不可并行部分的比例 f；预处理二：加入循环合并算法，将两个相邻的循环合并为一个循环，再进行并行化处理，从而可以改变并行化后并行执行程序的规模，使它达到最佳的处理规模 10^7。

参赛队号　1176

所选题目　A

参赛密码 _____
（由组委会填写）

Abstract

Based on the model in the first period, we accomplish the following tasks in this paper:

First, according to the suggestion given by the judges that "consider the ability of transforming the model and the effectiveness", we refer to the commonly used model---Amdahl accelerate ratio model. In this way we advance a developed Amdahl accelerate ratio S and make it one of the indices that evaluate the decompose algorithm.

Second, we fix on the evaluate formula to estimate the calculating scale of a cycle structure. Then we carry out a load balanced assigning model. Here we do not deeply explore the issue related to sequence and embranchment structure. In fact in the first period, we have already come to the conclusion by computer simulation that the decompose algorithm wound not have obvious effect in shorten the running of the sequence structure, and the effect to the embranchment structure depends on the certain program (great randomicity). Also, a program would not mainly consist of these two parts. Based on the above reasons we mostly consider the cycle structure. We introduce load balancing ratio m to quantitatively value the balancing level of the load distribution. It is designed as another index at the evaluation system.

Next, in order to make it easy to calculate and compare, we integrate these two indices with different weights. We devise the synthesis evaluating model:

$$d = a * m + b * S \qquad (a, b \text{ are the weights.})$$

As we know that choosing of the weights should be as external as possible, thus we use the method of AHP to obtain a, b. At last we get $d = 0.17m + 0.83S$.

Finally, we optimize the decompose algorithm by making the computer simulation. We get the curves of the relationships between the synthesis evaluate indices d and the calculation scale, also between d and f. Here, f is the proportion of the disparallel part in the program. Through analyzing these curses, we have the conclusion that if the calculation scale of the parallel part in the program is 10^7, two CPUs have the best efficiency. And if the proportion of the disparallel part is small enough, the two CPUs would also have good efficiency. Obviously, we can optimize the parallel algorithm applying to these two conclusions. Thus based on the computer simulation result, we get the optimize scheme to have pretreatment of the program before taking the parallel algorithm. Approach one, rearrange the sequences of sentences to reduce f; approach two, combine the cycle parts to change the calculation scale and finally increase the synthesis evaluate index.

#1176

一、 问题重述

在第一阶段中，我们初步研究了串行算法并行化的一些相关理论，重点讨论了串行算法能否并行化，怎样并行化的问题。我们已经建立了分解模型，对顺序结构，分支结构以及循环结构三种不同程序结构进行预处理，将能够使用双核心并行处理的部分分解开，并分配到两个核心上同时运行。

第二阶段主要解决的是如何判断分解算法的效率并对分解算法进行优化的问题。根据第一阶段建立的分解模型，将可并行部分分配至两个核心上进行运算时，却不一定能较好地达到负载均衡的要求。只有在合理划分任务的基础上，尽量将并行程序较均等地分配至双核进行处理，才能充分发挥双核计算机的速度优势。因此，为了更合理的分配任务给双核，我们需要对已经设计好的分解算法进行评价，并根据评价结果完善原来的分解方法。

问题：请建立合理的模型，对你们设计的分解方法的效果作出评价，并根据你的评价，有针对性地对分解方法作出优化。

二、 问题分析

2.1 如何界定一个好的并行分解算法

对一个串行程序进行并行分解的目的就在于能充分发挥并行处理的计算优势，提高信息处理速度。对于同一个串行程序，如果能用多个分解算法将其分解至双核，从根本上来说，使程序在两个处理器并行运行的时间最短的那个分解算法就是最优的算法。因此我们可以以此来设计分解算法的评价指标。

通过定义并行加速比指标以及负载均衡系数指标，可以较准确地评价并行算法的优劣。根据程序运行的特点，还可以用并行度，算法复杂度等指标，但有些指标所反映的并行算法的情况有所重复，实际上有可能只是一个比例关系，因此我们应该选择最具代表性、反映信息最全面的指标来衡量并行算法的有效性。

2.2 如何将多个指标科学地整合

以上所提到的指标都是从某个侧面来评价分解算法，要得出一个针对并行算法的评价模型，就必须对各个指标进行整合。当有多个指标对一个模型进行评价时，要注意到它们对评价模型的影响程度是不同的，反映信息较多的指标对模型有更大的影响程度。因此我们考虑采用层次分析法算出各指标的权重，以最大程度的减小人为决定权重的影响。最后将各个指标综合为一个评价模型。应用此评价模型对不同分解算法进行比较和评价，为分解算法的优化做准备。

2.3 如何检验模型的合理性，怎样为算法优化找突破口

通过分析可以知道，并行化程序的计算规模与不可并行化部分在串行程序中所占比例都会对并行程序运行时间产生比较大的影响。因此，我们利用计算机对这两个因素进行模拟，以此来分析它们对并行算法的影响，为分解算法的优化提供实验依据。

三、模型假设

1. 不考虑计算机之间的通信时间以及其他额外的程序管理开销，并行算法在求解一个问题时所耗费时间仅为处理器计算的时间；
2. 不考虑两个CPU在运算速度上的差异；
3. 忽略存储器容量的影响。

四、符号说明

a_{ij}	指标 f_i 和指标 f_j 对目标的影响程度之比
Q	循环体的计算量
L	一个程序段的计算量（负载）
m	负载均衡系数
T_i	使用 i 个处理器的程序执行时间
S_p	加速比
f	一个程序的串行部分占的比例
N	一个程序的并行部分占的比例

（注：上述是对本文的全局性符号，对文中建模讨论过程中我们可能引入的新的符号）

五、模型建立与求解

5.1 模型准备

层次分析法简介

在评价一个目标事物时，一个评价体系之中往往有多个指标，每个指标反映了对这个目标某个方面的影响。但是不同指标对于整个评价体系的影响是不同的，所以我们应该在评价体系中给不同指标设定不同权重。层次分析法给我们提供了求解权系数的科学方法。它的大致求解过程如下：按不同指标对评价体系的影响程度，将指标分解为若干层次，各指标之间进行简单的比较和计算，得到不同指标的权重，再将所有指标进行整合得到整个评价体系。

评价系统可以分成两个层次：

1）目标层，即所评价的目标

2）准则层，包括各个不同指标

假设共有 n 个指标 $F=\{f_1, f_2, ..., f_n\}$，要比较这 n 个指标评价同一目标的重要程度，每次取两个指标 f_i 和 f_j，用 a_{ij} 表示 f_i 和 f_j 对目标的影响程度之比，其中 a_{ij} 取值由 *Satty* 的 1~9 值法决定：

#1176

Satty 的 1~9 值法表

标度	含义
1	表示两个因素相比，具有相同重要性
3	表示两个因素相比，前者比后者稍重要
5	表示两个因素相比，前者比后者明显重要
7	表示两个因素相比，前者比后者强烈重要
9	表示两个因素相比，前者比后者极端重要
2，4，6，8	表示上述相邻判断的中间值
倒数	若指标 f_i 与指标 f_j 的重要性之比为 a_{ij}，那么指标 f_j 与指标 f_i 的重要性之比为 $a_{ji}=\frac{1}{a_{ij}}$。

5.2 模型建立

5.2.1 可执行程序段的计算量估计

我们在挑战赛的第一阶段就通过计算机仿真测试验证了并行化算法对简单的顺序结构程序的运行时间没有明显的缩短，对选择结构的改善效果取决于具体的程序（随机性较大），同时一个程序的计算量并不主要取决于这两种结构。鉴于以上原因，我们主要考虑了循环结构。

下面给出循环程序计算量的量化模型：

考虑如下 m 重循环：

```
L1: FOR i1=1 TO N1 DO
L2:     FOR i2=1 TO N2 DO
          ……
Ln:         FOR in=1 TO Nn DO
              { 循环体 }
            END
          ……
        END
    END
```

设循环体的计算量为 Q，则总的计算量大致是 $L=Q\prod_{i=1}^{n}N_i$。

5.2.2 负载平衡调度模型

设 A、B 是一台计算机上的两个处理器。设 $L_1,L_2,...,L_n$ 分别为对应的 n 个并发执行程序段的计算量。分析可知，为了将并发执行程序合理地分配到两个处理器上并使双核的计算时间相当，就是要让两个处理器的计算量接近 $\frac{1}{2}\sum_{i=1}^{n}L_i$。

为此设一组 0—1 变量 x_i $(1 \le i \le n)$

$$x_i = \begin{cases} 0 & L_i \text{ is not assigned to processor A} \\ 1 & L_i \text{ is assigned to processor A} \end{cases}$$

使下式取得极值的一种分配方法就是使两个处理器的计算时间相当的一种任务调度：

$$\min \left| \sum_{i=1}^{n} x_i L_i - \frac{1}{2} \sum_{i=1}^{n} L_i \right|$$

5.2.3 评价指标模型

为了定量地评价我们设计的分解算法的合理性，我们定义了负载均衡系数 m 和改进的 Amdahl 加速比 S 两个指标，并通过加权将它们整合成一个综合评价指标模型。为了最大限度地减少主观因素的影响，我们通过层次分析法求解对应的权系数。

5.2.3.1 负载均衡系数

负载均衡系数主要刻画了两个处理器上负载的“平衡度”。处理机间负载平衡的非最优化会使计算机出现一个处理器忙而另一个处理器闲的情况，从而使两个处理器的整体处理效率达不到预期的设想。

设 C_A 和 C_B 分别表示 A、B 两个处理器的计算量，则

$$C_A = \sum_{i=1}^{n} x_i L_i \,, \quad C_B = \sum_{i=1}^{n} (1 - x_i) L_i$$

我们定义负载均衡系数为： $m = 1 - \dfrac{|C_A - C_B|}{\frac{1}{2}(C_A + C_B)} (0 \le m \le 1)$

当 $C_A = C_B$ 时，$m = 1$，它表示负载被均匀地分配在两个处理器上，这是比较理想的一种情况。

当 $C_A \ne C_B$ 时，$m < 1$，它表示负载分配不均匀，m 越小，两个负载的不均衡情况越严重；m 越大，说明负载越均衡。

5.2.3.2 改进的 Amdahl 加速比模型

串行算法与并行算法执行时间的比称为并行算法加速比。

T_i 表示使用 i 个处理器的算法执行时间。

$$\text{加速比 } S_p = \frac{T_1}{T_p}$$

其中，T_1 指串行程序在单处理机上的运行时间，T_p 指该串行程序被并行化分解后在 p 台处理机上同时执行所需的时间。

由于本模型仅考虑 $p = 2$ 即两个处理器的情况，故

$$\text{加速比 } S = \frac{T_1}{T_2}$$

在并行计算中，常用的加速比计算模型是 Amdahl 加速比模型。该模型适用于固定

规模的问题。它的具体思想如下：设一个程序的串行部分占的比例为 f，并行部分占的比例为 N，则有 $f+N=1$。若并行系统的处理机个数为 p，且忽略通信同步等由于并行化所引起的额外开销，则有：

$$S_p=\frac{f+N}{f+N/p}$$

当处理机个数为 2 时；有：

$$S_2=\frac{f+N}{f+N/2}=\frac{1}{f+N/2}$$

在这个模型中，对于可并行部分，认为可以以字节均分给两个处理器，但是在实际情况下，由于并行粒度等的影响，可并行的部分不一定能完全均分给两个处理器，所以我们对此模型进行一定的改进，得到改进的 Amdahl 加速比模型。

改进的 Amdahl 加速比模型：将并行部分（比例为 N）按照负载平衡调度模型分配到两个处理器之后，两个处理器所分担的负载的比例分别为 N_1,N_2，且满足 $N_1+N_2=N$。定义改进的 Amdahl 加速比模型为：$S=\frac{1}{f+\max(N_1,N_2)}$

对两个处理器而言，若串行部分占的比例 $f=0$，则 S 最大可以达到 2，此时的负载也是最均衡的；若 $f\neq 0$，那么无论怎么增加处理器的个数，S 最大能达到 $\frac{1}{f}$。

我们的评价指标模型就是利用负载均衡系数 m 和改进的 Amdahl 加速比模型 S 的带权组合构成的。

5.2.3.3 综合评价指标模型

我们的综合评价指标模型是利用负载均衡系数 m 和改进的 Amdahl 加速比模型 S 的带权组合构成的。

设综合评价指标模型 $d=a\times m+b\times S$

其中 a，b 是 m 和 S 在评价指标模型 d 中的权重，它们的值用层次分析法确定。d 是 m 的单调递增函数，同时也是 S 的单调递增函数，因而 d 越大，说明并行算法越优越。

下面用层次分析法确定权系数 a,b：

（1） 确定判断矩阵：

判断矩阵为：$A=\begin{bmatrix}a_{11} & a_{12}\\ a_{21} & a_{22}\end{bmatrix}$

a_{11} 表示指标 m 对目标函数的影响程度与指标 m 对目标函数的影响程度之比；

a_{12} 表示指标 m 对目标函数的影响程度与指标 S 对目标函数的影响程度之比；

a_{21} 表示指标 S 对目标函数的影响程度与指标 m 对目标函数的影响程度之比；

a_{22} 表示指标 S 对目标函数的影响程度与指标 S 对目标函数的影响程度之比，

显然 $a_{11}=1$， $a_{22}=1$；

根据上面的分析可知，指标 S 比指标 m 更重要，根据 *Satty* 的 1~9 值法表，可以选取 $a_{12}=1/5$， $a_{21}=5$。

因此， $A=\begin{bmatrix}1 & 1/5\\ 5 & 1\end{bmatrix}$。

（2） 求取判断矩阵 A 的特征值与特征向量

$\det(lI-A)=(l-1)^2-1=l^2-2l$

因此，判断矩阵 A 的最大特征值 $l_{\max}=2$。对应于该特征值的特征向量是：

$$v=\begin{bmatrix}0.1961\\ 0.9806\end{bmatrix}$$

（3）求取权系数

由 v 可以求得 m 和 S 对应的权系数：

$a=0.1961/(0.1961+0.9806)=0.17$

$b=0.9806/(0.1961+0.9806)=0.83$

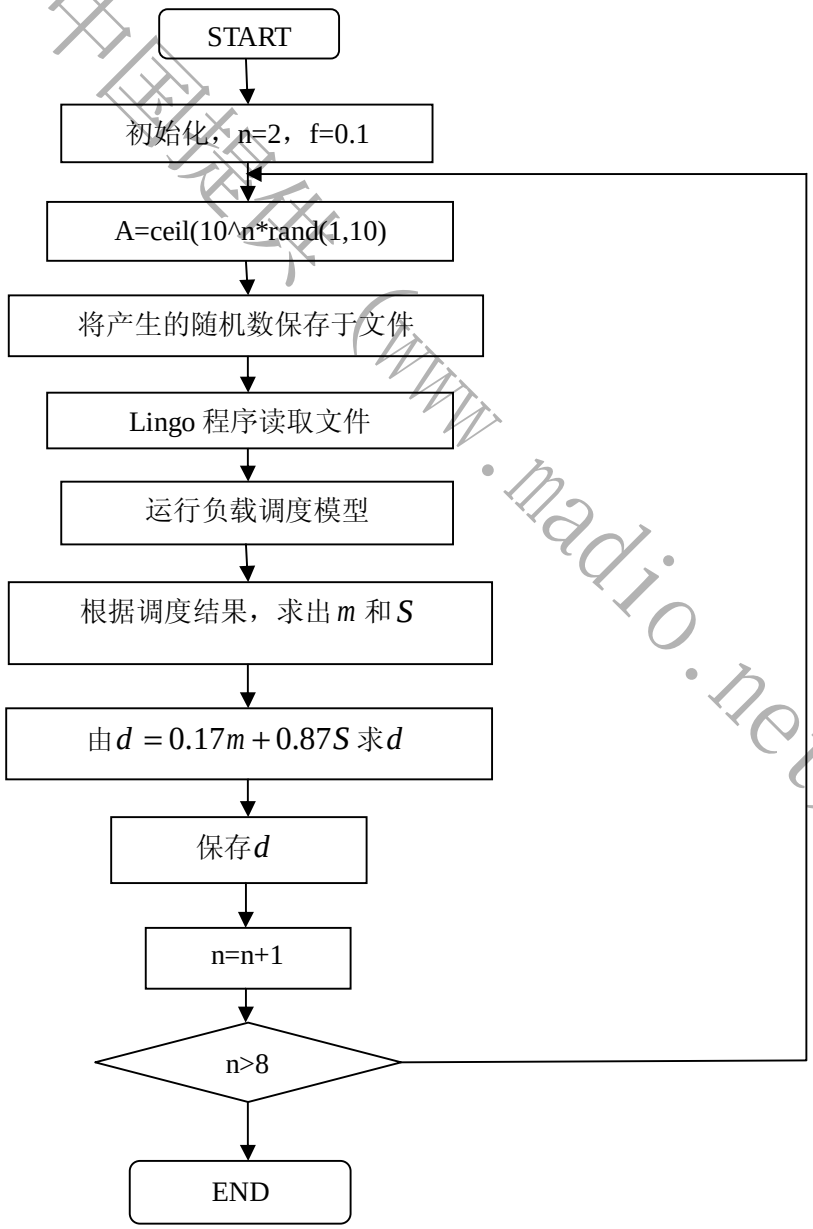
综合以上三步可得综合评价指标模型：$d=0.17m+0.83S$

六、计算机模拟计算

为了检验模型的合理性并为后面的分解算法优化提供依据，我们设计了模拟算例。这一部分我们主要完成了求取综合评价指标d 与计算规模、综合评价指标d 与程序中的不可并行部分的比例 f 之间的关系。

6.1 综合评价指标d 与计算规模的关系

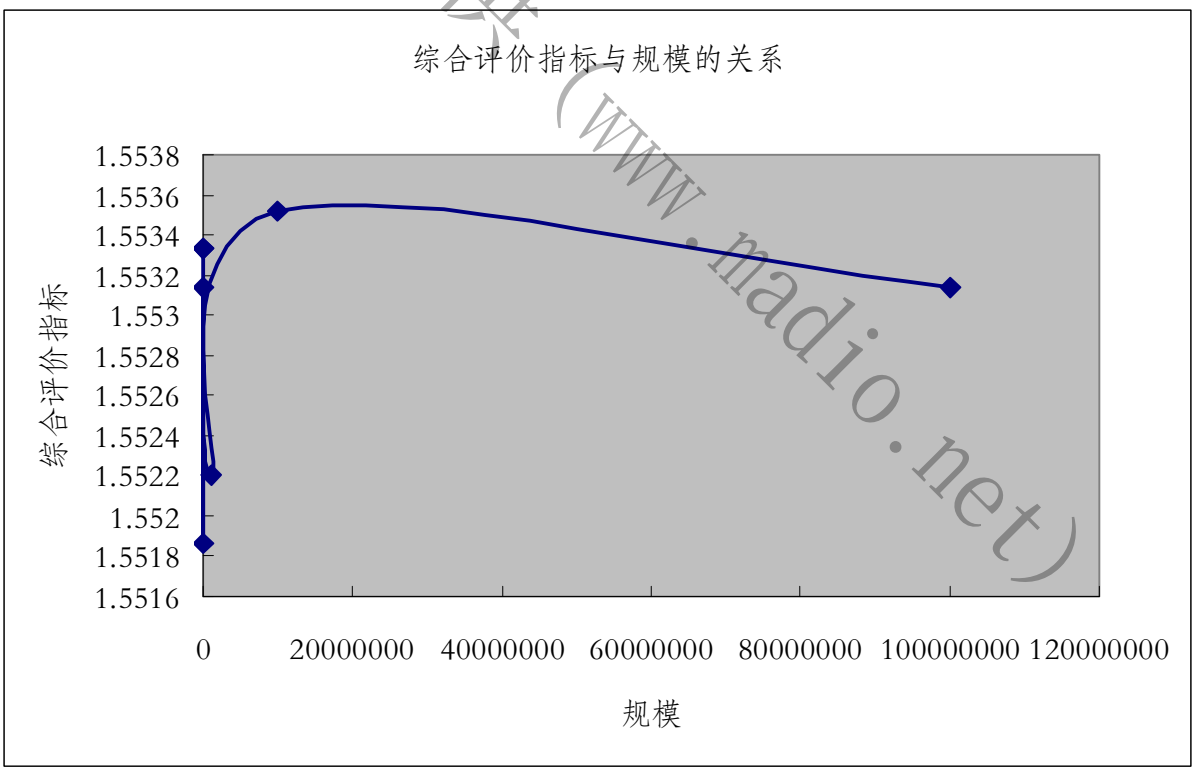
为了简化计算，我们假定程序中不可并行部分的比例为 $f=0.1$，可以并行执行的程序段有 10 个。显然这 10 个可并行执行的程序段的计算量是随机的。我们用 ceil(10^n*rand(1,10))这个语句产生规模为10^n（n=2,3,……8）的 10 个整数，用它们来模拟 10 个可并行执行的程序段的计算量。算法如下：



计算结果如下表所示：

数据 / 规模	C_A	C_B	m	S	d
100	294	294	1	1.666667	1.553333
1000	2811	2811	1	1.666667	1.553333
10000	22808	22763	0.99802506	1.665296	1.55186
100000	283394	283467	0.999742441	1.666488	1.553141
1000000	2593442	2589505	0.998480787	1.665612	1.5522
10000000	26979530	26991600	0.999552724	1.666977	1.553515
100000000	283466400	283390700	0.999732913	1.666481	1.553134

根据以上的数据做出d 与规模的关系图如下：



由图表可以看出：可并行执行的程序段的计算量太小或太大都会导致串行程序并行化后的双核的效能不能得到最大的发挥。当计算量较小（如10^3量级）时，串行算法已经能够满足实时性的要求，没必要进行并行化；当计算量较大（如10^{10}量级）时，双核并行计算也不能有效地达到预期的速度，这时应该增加处理器的数目；当可并行执行

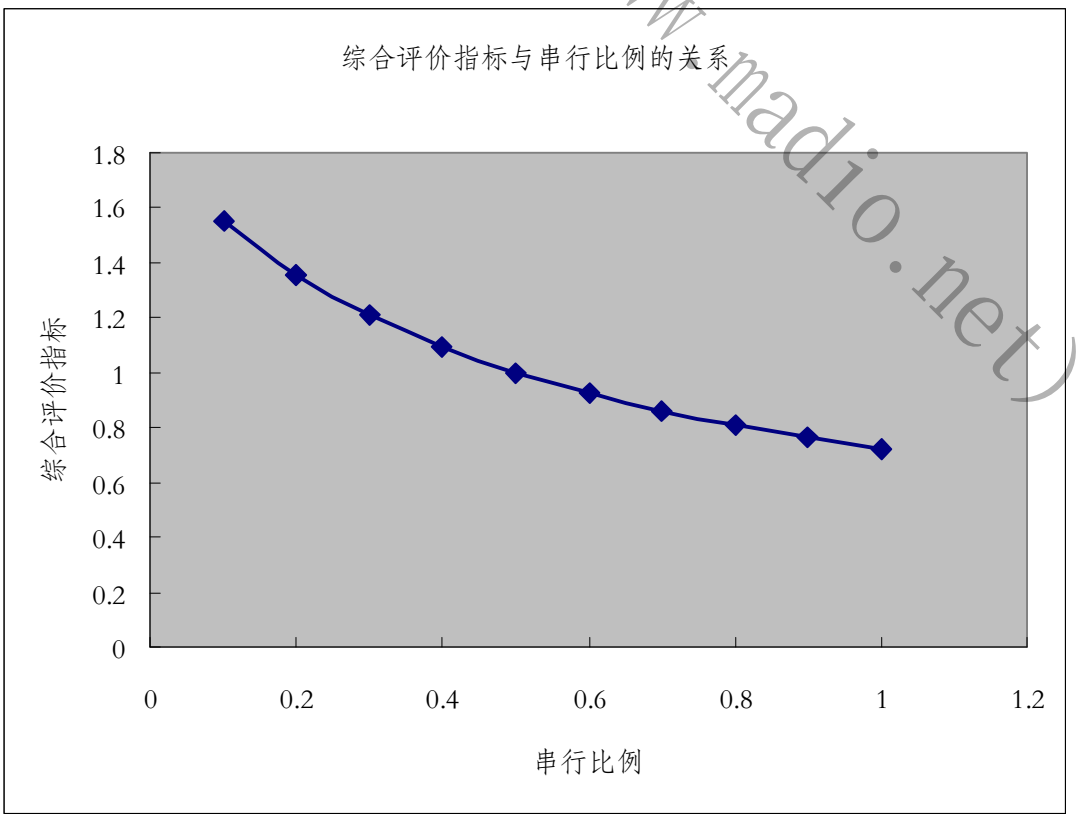
的程序段的计算量是10^7量级时，双核的效能才能达到最大。因而我们的分解算法改进方向之一是：在并行化之前先判断串行程序的计算规模，当规模在10^7量级时再进行后续的并行化操作，否则应合并或分解循环使并发执行程序段的计算规模达到10^7量级。

6.2 综合评价指标d与程序中的不可并行部分的比例f之间的关系

由于在上面已经得到了最适合并行化的问题规模是10^7，所以在这一部分我们就把问题的规模取为10^7，那么C_A、C_B、m都确定了，只有S随不可并行部分的比例f而变化。那么可以等间距地取若干个不同的f，分别计算S和d，然后在做出d与f的关系图。计算结果如下：

数据 / 规模	C_A	C_B	m	f	S	d
10000000	26979530	26991600	0.998480787	0.1	1.666356	1.552817
10000000	26979530	26991600	0.998480787	0.2	1.428343	1.355267
10000000	26979530	26991600	0.998480787	0.3	1.249825	1.207097
10000000	26979530	26991600	0.998480787	0.4	1.110973	1.091849
10000000	26979530	26991600	0.998480787	0.5	0.999888	0.999649
10000000	26979530	26991600	0.998480787	0.6	0.908999	0.92421
10000000	26979530	26991600	0.998480787	0.7	0.833256	0.861344
10000000	26979530	26991600	0.998480787	0.8	0.769165	0.808148
10000000	26979530	26991600	0.998480787	0.9	0.714229	0.762552
10000000	26979530	26991600	0.998480787	1	0.666617	0.723034

根据上面的计算结果做出d与f的关系图如下：



由图表可以看出：不可并行部分的比例越低，双核的效能发挥的越明显。因而我们的分解算法改进方向之二是：对程序的语句进行必要的预处理（如语句重排、无关性处理等），尽量减少不可并行部分的比例。

七、分解算法的优化

7.1 减小不可并行部分的比例

从计算机模拟计算结果可以看出，随着一个程序中串行程序所占比例的增加，转换模型的效率评价指标逐渐降低，所以对于一个程序，我们可以考虑并行化之前增加一些预先处理，进行程序变换，挖掘串行算法的可并行潜力，减少只能串行执行的程序所占的比例，使下一步的串行算法并行化处理达到最佳效果，提高整个模型的转换能力。
我们认为对程序进行无关化处理是一个有效的途径。
例如：

```
e=(ab+c)d+f;
m=(cd-f)g-e;
```

这两条指令按原模型考虑因存在数据相关性而不能进行并行化处理，但是利用中间变量，可以将其无关化为：

```
L1:  x=ab;
L2:  y=cd;
L3:  z=x+c;
L4:  r=y-f;
L5:  s=zd;
L6:  t=rg;
L7:  e=s+f;
L8:  m=t-e;
```

此时再按原模型中对顺序执行程序进行串行化处理的算法，可以进行并行化，形成以下两个队列：

QUEUE1	QUEUE2
L1	L2
L3	L4
L5	L6
L7	

L8 与两队列均数据相关，故 L8 为一个同步点。
这样，经过无关化预处理，挖掘了串行程序的并行化潜力，降低了不可并行程序所占比例，提高了指标，改善了分解算法的能力。

7.2 对可并行执行程序块运算规模的处理优化

另外，因为分解算法的效果与并行化后可并行执行的程序块的规模有一定的关系，

当规模达到10^7级的规模时，模型的转换指标最优，对于一个一般的程序,对其进行代码优化，可以改变程序的并行化处理后可并行程序的规模，以提高并行化效率及转换能力，如：循环合并，也将两个相邻的循环合并为一个循环，在根据实情进行并行化处理，可能增加并行执行的粒度，也减少组织并行执行的开销。

示例：

```
L1：FOR  i=1 TO N  DO
S：      A(i)=C(i)*2
    END
L2: FOR i=1 TO N DO
T：      B(i)=A(i)+2
    END
```

显然，两个循环可合并为一个：

```
L：FOR  i=1 TO N  DO
       A(i)=C(i)*2
       B(i)=A(i)*2
   END
```

在循环合并之前，先对两个循环分别进行可并行性测试，再分别进行并行化处理，但是经过合并后，两个循环可以当作一个循环，对其进行并行化处理，已经改变了并行化后并行执行程序块的规模，当规模合适时，就可以提高程序转换的效率。

当然，循环合并也是有条件的，有的代码中可以把两个相邻的循环合并为一个循环，当两个循环具有相同的迭代空间时，可完全合并为一个循环，当两个循环迭代空间不同时，要按迭代空间小的组织合并的循环，然后再用一个循环将迭代空间大的循环中未执行的那部分迭代空间补上。以下以迭代空间相同的为例进行研究。

循环可合并的条件：

令循环 L_1，L_2的迭代空间相同，L_1 在 L_2之前，对 L_1中的任意语句S_j和L_2中的语句T_k，若S_j与T_k之间不存在依赖方向为-1 的依赖关系，则两个循环可合并为一个。

示例：将上例稍改动：

```
L1：  FOR  i=1 TO N  DO
S：       A(i)=C(i)*2
      END
L2:   FOR i=1 TO N DO
T：       B(i)=A(i+1)+2
      END
```

此时 T 依赖与 S，且依赖方向为-1，故不能合并为一个循环。

这样，在相邻循环可以合并的条件下，根据其规模，若通过对相邻的循环进行循环合并，可以使并行程序的规模达到模型中并行化效果更好的规模，此时进行循环合并可以在一定程度上改善模型的转换效率。

八、模型优缺点

1．**模型优点：**

1）采用改进的 Amdahl 加速比模型确定并行程序的加速比，对并行程序任务分配考虑得更加细致，对并行化算法的加速比评价更加准确；

2）模型中提出的指标精简有效，运用层次分析法求出各个指标的权重，最大程度地减少了主观因素的影响，模型客观有效，更便于评价，同时理论依据更加充足；

3）通过计算机模拟，得出评价指标与程序中能串行执行程序所占比例及各并行程序块的规模的关系，发现串行算法并行化的弱点，合理的提出了可行的并行化算法的优化方案。

2. **模型缺点：**

1）评价指标并不全面，例如并行化算法本身的算法复杂度，算法的转换能力，算法并行化带来额外开销等都没有定量的考虑。

2）因为算法具有很大的随机性，故优化时提出的模拟算例通用性并不是特别强，只能对特殊的程序取得较好效果，还有待拓展。

九、参考文献

【1】沈志宇 《并行编译方法》 国防工业出版社 2000 年

【2】张德富 《并行处理技术》 南京大学出版社 1992 年

【3】陈国良 《并行算法的设计与分析》 高等教育出版社 1994 年

【4】李思昆 曾芷德 《数字系统并行 CAD 技术》 国防工业出版社 2000 年

【5】胡玥 高庆狮 高小宇 《串行算法并行化基础》 科学出版社 2008 年