一种多范例并行应用系统的描述方法和性能估算模型

胡长军 张素琴 田金兰

(清华大学计算机科学与技术系 北京 100084)

摘 要 多范例并行是大规模并行应用系统的本质特征.规范化描述并行应用系统,建立性能估算模型对于提高多范例并行应用系统的开发效率和运行效率具有重要意义.该文提出了一种基于模块及其组合关系的描述方法和系统执行代价计算模型,它不仅能描述并行应用系统的多范例特征,而且将不同并行范例模块的组合时产生的代价引入模型.考虑的代价包括并行执行模式的转换、数据分布方式的转换以及编程范例的转换等,从而使模型更为准确.给出了描述和代价估算的应用实例,说明了规范化描述和代价估算对于确定并行策略的重要性以及模型的精确性.

关键词 并行软件工程;多范例并行计算;性能评价;并行应用系统

中图法分类号 TP311

A Specification and Performance Evaluation Method for Multi-Paradigm Parallel Application Systems

HU Chang-Jun ZHANG Su-Qin TIAN Jin-Lan

(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)

Abstract Multi-paradigm parallelism is the characteristic of large-scale parallel application systems. It is necessary to give the specification of a parallel application system and set up performance evaluation model for improving the developing and executing efficiency. In this paper, a specification method and performance evaluation model of large-scale parallel application systems is advanced, based on modules and their composition. It not only can describe the characteristics of multi-paradigm parallel application systems, but also introduces the composition cost of different parallel paradigm modules into the model. The cost includes the transforms of parallel executive models, data distribution and parallel programming paradigms, which makes the whole model more practical. We also analyze an instance in detail to verify the effect of our performance evaluation model and explain how to specify a multi-paradigm parallel application systems and how to determine parallelization strategies.

Keywords parallel software engineering; multi-paradigm parallel computing; performance evaluation; parallel application system

1 引 言

随着微机性能价格比的不断提高,利用微机群系统实现高性能计算,开发大规模并行应用系统的研究,逐渐引起了人们的重视,并行软件工程方法研究也就应运而生了.它是一个全新的领域,从应用需求的角度看,它至少包括两类内容,一类是如何高

收稿日期:2001-12-03;修改稿收到日期:2002-02-25.本课题得到国家自然科学基金(60083004,60173004)和国家"八六三"高技术研究发展计划项目(863-306-ZT01-02-3)资助.**胡长军**,男,1963 年生,博士,教授,主要研究领域为并行编译与并行计算、领域软件工程与数据工程.E-mail: huchangjun@tsinghua.edu.cn.**张素琴**,女,1945 年生,教授,主要研究领域为编译技术、嵌入式系统、软件工程.**田金兰**,女,1945 年生,副教授,主要研究领域为软件工程、嵌入式操作系统.

效、规范化地并行化现有的大规模串行应用系统;另一类是如何高效地开发新的大型并行应用系统,实现并行系统开发的构件化,实现并行软件的重用.就大规模并行工业应用系统而言,任何系统都是由众多模块组成的,这些模块中既有并行模块又有串行模块,不同并行模块的编程范例和运行范例也不相同,有数据并行也有任务(控制)并行,有全局(global)编程模式也有局部(local)编程模式,因此,一般来说,大规模并行应用系统是一个多范例串并行模块组成的混合系统.从软件工程的角度看,无论是开发大型并行应用系统还是并行化现有的串行系统,如何规范化地描述不同范例的串并行模块及其组合关系,并在此基础上对系统的整体性能进行估算是一个关键问题.它直接关系着系统整体并行策略的确定、模块编程范例和运行范例的选择、整体并行效率的评价等.特别是对于并行化串行系统而言,我们的目标不仅要求并行后的系统在计算结果上要等价于串行系统,并行化系统在效率上要明显优于串行系统;更要求并行化过程尽量简单、尽量重用原来的代码,实现这个目标一要有正确的并行化策略,即根据应用问题的特点选择最佳的并行范例和最大限度地挖掘并行性;二要有恰当的性能评价模型,对并行系统进行性能预估以评价并行策略.实际上,由于并行化会引入额外的计算代价,有些模块的并行化可能不会提高效率;三是要尽量结构化、模块化、尽量屏蔽体系结构的细节.这些要求并不是孤立的,有时会有矛盾,如最大限度地挖掘并行性和尽量简化并行化过程以及提高程序的可读性等就有冲突.建立多范例并行应用系统的规范化描述以及合理的性能评估模型,是解决这一问题的有效途径.

并行系统的规范化描述和性能估算的研究一直是一个热点.Rauber 给出了一种基于模块组合的描述方法[1],该方法针对并行应用系统的分析和设计,强调了并行描述的规范性和可操作性,但是它没有考虑到模块的编程范例和运行范例的问题,实际上这是相当重要的,这直接关系到整体性能模型建立的准确性,因为不同程序范例模块的组合代价是不同的.Luksch 结合并行化大规模串行应用系统的实际,研究了并行化过程的工程化方法[2],他在文章中强调了如何考虑体系结构的特点,提高系统的效率,强调了规范化描述系统的重要性,但并没有给出具体的描述方法.Nyland 给出了从软件需求分析出发,开发并行系统的工程化方法[3],但也没有考虑多范例并行模块的组合等问题,此外,Adve[4],Cremonesi[5]等分别针对不同的并行模型给出了性能估算方法.本文结合利用 *p*-HPF 数据并行语言,多范例并行化大规模串行应用系统的实际,提出了一种以数据并行为核心多范例并行应用系统的描述方法,以此为基础,建立了一个性能评价模型.*p*-HPF 是我们开发的支持多范例并行模块组合的并行编译系统,用不同语言编写的数据并行模块、任务并行模块、并行算法库以及串行模块等可以利用其 Extrinsic 机制有效地集成在一起[7].本文给出的方法特色在于充分考虑了应用系统的多范例并行特点,把多范例模块组合时不同范例间转换的代价引入模型,因此较已有的方法相比,具有更高的实用性.利用这种方法,可以有效地描述多范例并行系统,并对系统的效率进行预估.虽然这种预估可能是不够精确的(精确的估价需要体系结构、网络传输、通信方式、运行支持等详细的细节),但是对于确定系统的整体并行策略,指导串行系统的并行化仍然有很大的意义.本文第 2 节给出多范例并行应用系统的描述方法;第 3 节是性能估算模型;第 4 节给出一个应用实例;最后是结论和进一步的研究工作.

2 多范例并行应用系统的规范描述

我们通过描述一个系统中每一个模块的编程范例和运行范例以及不同范例模块间的组合关系来描述一个多范例应用系统.规范描述包括三个内容:(1)基本模块描述;(2)组合模块描述;(3)模块组合表达式.

2.1 基本模块(BM)描述

所谓基本模块,就是指应用系统中具有特定功能,而功能的实现没有调用其它模块的模块.基本模块的描述方式如下:

$$MODUL^{r}\text{-}NAME^{p}(IN\ inputlist;OUT\ outputlist)$$
$$\{var(m)=\{\cdots\},acc(m)=\{\cdots\}\}.$$

其中,r 为模块运行范例标识:r 取“~”表示模块 Serial 运行,取“//”表示 Local 方式运行,“◇”表示 Fork-join 方式运行.

p 为模块编程范例标识,p 取“s”表示串行高级语言编程,取“l”表示并行 Local 编程范例,如 F77+MPI 编程等,取“g”并行全局编程范例,如 HPF,OpenMP 等.关于不同运行和编程范例的详细论述参见文献[7].

inputlist 为模块的输入参数列表,格式为 $x_1:type_1:\{d\}$, $x_2:type_2:\{d\}$,…, $x_n:type_n:\{d\}$, d 是

可选项,表示分布方式,*type* 为参数类型.

outlist 为模块的输出参数列表,格式同输入参数.

$var(M)$表示模块 M 的可赋值变量集,也就是在模块中可以被修改的变量集合. $acc(M)$为 M 的可引用变量集,也就是在模块中只能引用而不能被修改的变量集合. 显然有

$$inputlist \in acc(M),\quad outputlist \in var(M).$$

2.2　组合模块(CM)描述

组合模块是指具有明确功能,调用基本模块或其它组合模块形成的模块,或者是由循环、分支等程序结构和基本模块组成的模块. 组合模块也是多范例的,记法同基本模块. 组合模块的定义是通过模块表达式的形式进行的,格式为

$$MODUL \overset{r}{-} NAME^{p}(IN\ inputlist;OUT\ outputlist) = \mathrm{mod}\ ul\_expr.$$

递归定义模块组合表达式 F 如下:

$$\begin{aligned} F ::= & \mid BM(IN\ inputlist;OUT\ outputlist) \\ & \mid CM(IN\ inputlist;OUT\ outputlist) \\ & \mid F;F \\ & \mid F \parallel F \\ & \mid b^{*}F \\ & \mid parFor(I=1,2,\cdots,n)\ {}^{*}F \\ & \mid F \lhd b \rhd F' \\ & \mid \{F\}. \end{aligned}$$

其中,";"为组合符号,$M1;M2$ 表示两个模块由于数据相关或控制相关的原因,只能按 $M1,M2$ 的顺序组合.

"$\parallel$"也是组合符号, $M1 \parallel M2$ 表示两个模块可以并行执行.

"$b^{*}F$"表示当布尔表达式 b 成立时,循环执行模块表达式 F.

"$parFor(I=1,2,\cdots,n)^{*}F$"表示对 F 的循环执行,与循环执行顺序无关,即不存在迭代流相关性以及反相关性,这意味着可以将循环差分成若干块在不同接点并行执行.

"$F \lhd b \rhd F'$"表示如果条件 b 成立则执行模块表达式 F,否则执行另一模块表达式 F'.

通过以上定义,可以将一个应用程序通过模块表达式表示出来.

3　性能估算模型

无论是开发新的大规模并行应用系统,还是串行化一个已有的串行系统,一个核心问题是运行效率,特别是串行系统的并行化,并不是所有模块并行化都会带来效益的,因此,及时对并行性能进行估算是极其必要的.

设串行系统的执行代价为 C_s,其对应的多范例并行系统的执行代价为 C_p,只有 $C_s > C_p$时并行化工作才有意义,通常将 C_s/C_p称为加速比.

若一个系统由 N 个模块构成,每一个模块的执行代价为 $C_1, C_2, \cdots, C_n$.

显然有 $C_s = C_1 + C_2 + \cdots + C_n$,即整体代价只与各模块的执行代价相关.

但是,多范例并行系统的整体代价,不仅与各模块的执行代价相关,而且和不同模块的编程范例和运行范例的转换密切相关. 下面是我们给出的一个基于多范例并行规范化描述的多范例并行系统的整体代价估算模型. 值得指出的是,这里给出的模型,其目的并不是精确计算模块的运行时间,而是从整体性能评价和并行策略选择的角度,提供一种度量的方法. 为确定系统的整体策略提供依据.

3.1　并行模块的代价计算

一个模块并行执行的代价,等于并行计算的代价加上因为引入并行所需的代价. 对于以数据并行为核心的多范例并行系统来说,并行引入的代价可以从如下几个方面度量:(1)数据并行引入的数据的初始分布代价;(2)计算过程中产生的数据重分布代价;(3)计算结果的收集代价;(4)为了实现并行引入的变量定义等其它必要代价. 用公式表示为

单个并行模块的代价

$$\begin{aligned} C(\overset{//}{M}) = & C(M\_comput) + C(load\_distribute) + \\ & C(redistribute) + C(gether) + C(others) \end{aligned} \tag{1}$$

其中,$C(M)$是完成必要的计算所需的代价,$C(load_distribute)$是读入数据并实现分布的代价,$C(redistribute)$是当 $acc(M) \cap var(M) \neq \varnothing$时的数据重分布代价,$C(gether)$为数据处理完后的收集输出代价,$C(others)$为并行冗余的代价.

从式(1)可以看出,并行模块的执行代价与由并行化带来的通信代价密切相关,通信代价的估算与硬件设备及体系结构密切相关,在 Cluster 通信环境下,对其通信代价模型的研究文献很多,文献[1]给出的一个模型如下:

单次传送:

$$ts2s(b) = \tau + t_c \times b \tag{2}$$

广播：

$$t_{sb}(p,b)=\log(p)+t_c\times\log(p)\times b \quad (3)$$

规约：

$$t_a(p,b)=\tau\times p+t_c\times p\times b \quad (4)$$

多级广播：

$$t_{mb}(p,b)=\tau_1+\tau_2\times p+t_c\times p\times b \quad (5)$$

gather/scatter：

$$tg(p,b)=\tau_1+\tau_2\times p+t_c\times p\times b \quad (6)$$

这里 p 为参与通信的处理器个数，b 为参与通信的字节数，τ 为常量，表示通信启动时间等，它与具体的体系结构有关，可以通过实验得到. 在以上模型的基础上可以估算并行模块的精确执行代价.

3.2 模块组合的代价计算

在多范例并行计算系统中，模块组合的代价与模块组合关系密切相关，设有模块 M_1，M_2，其运行范例为 $r1$，$r2$，编程范例为 $p1$，$p2$. 下面给出不同组合方式时的代价计算方法.

3.2.1 顺序执行的代价估算

$p1=p2$ 即两模块编程语言相同时顺序执行模块的代价

$$C(\overset{r1}{M1}^{p1};\overset{r2}{M2}^{p2})=C(\overset{r1}{M1}^{p1})+C(\overset{r2}{M2}^{p2})+C_{R\_paradigm\_trans}(r1,r2) \quad (7)$$

若 $r1\neq r2$，则

$$C_{R\_paradigm\_trans}(r1,r2)=\begin{cases}C(scatter), & r1=\approx,\ r2=//\\ C(access\_consistency), & r1=\approx,\ r2=\diamondsuit\\ C(gether), & r1=//,\ r2=\approx..\\ C(remap), & r1=//,\ r2=\diamondsuit\\ C(remap), & r1=\diamondsuit,\ r2=//\\ C(Gether), & r1=\diamondsuit,\ r2=\approx\end{cases} \quad (8)$$

若 $r1=r2$，则

$$C_{R\_paradigm\_trans}(r1,r2)=C_{data\_redistribution}.(M1\rightarrow M2) \quad (9)$$

式(7)说明，在并行环境下，两个模块的顺序执行的总代价大于两模块在单结点上的代价，这是由于不同范例的转换引起的，范例转换代价的大小与两模块的运行模式密切相关，式(8)给出了不同运行范例模块顺序执行时转换的计算方法：从串行范例到并行范例转换时，需要数据从一个接点分发到各接点，所以需要 Scatter 通信；从 Serial 范例转换为 Fork_join 范例时，需要解决内存访问一至性控制问题；当由 Local 范例转换为 Serial 范例时，需要收集接点数据到一个接点上，这就是 Gether 通信的代价；当 Local 范例和 Fork_join 范例相互转换时，则需要数据重分布，这就是 remap 通信的代价[6]；当有 Fork-join 范例转换为 Serial 范例时，需要将数据收集到一个接点上，也就是 Gether 通信的代价. 可见，顺序执行时，范例的转换代价取决于通信的代价，它最终取决于通信量和通信方式. 当两个模块的运行模式相同时，副作用的大小是数据的重分布代价. 由于两个模块运行范例相同，所以不必做范例转换，但是数据的分布可能不同，所以需要考虑重分布的代价如式(9)所示.

当两个模块的编程范例不同时，即 $p1\neq p2$ 时，除了考虑以上运行范例之间的转换代价，还要考虑不同语言间的转换代价，主要是对数据存取方式不同产生的转换代价，和具体的语言有关，如当两个模块顺序分别是 C 语言和 Fortran 语言时，需要将按行优先存储的数组形式转换为列优先存储；当为 HPF 和 F77 语言时，需要压缩存储孔穴转换为 F77 数组形式.

3.3 并行执行的代价计算

并行执行模块的代价.

两个并行执行的多范例并行模块的总代价，等于运行代价最大的一个，如式(10)，

$$C(\overset{r1}{M1}^{p1}//\overset{r2}{M2}^{p2})=\max(C(\overset{r1}{M1}^{p1}),C(\overset{r2}{M2}^{p2})) \quad (10)$$

此式说明，只有两个执行代价相差不多的模块并行执行才有意义，这也是负载均衡的重要性.

3.4 循环执行模块的代价估算

并行循环块的代价计算.

$$C(parFOR(i=1,2,\cdots,n)\overset{r}{M}^{p})=\frac{n}{nump}\max_{1\leq i\leq n}(C(\overset{r}{M}^{p})+C_{R\_paradigm\_trans}(\overset{r}{M}^{p})+C_{P\_paradigm\_trans}(\overset{r}{M}^{p})) \quad (11)$$

$$C_{R\_paradigm\_trans}(\overset{r}{M}^{p})=\begin{cases}C(gether)+C(scatter), & r=\sim\\ C(redistribute), & r=//\\ C(access\_consistency), & r=\diamondsuit\end{cases} \quad (12)$$

$$C_{P\_paradigm\_trans}(\overset{r}{M}^{p})=\begin{cases}0, & \text{当 } p \text{ 与宿主程序编程语言一致时}\\ C(memory\_trans), & \text{当 } p \text{ 与宿主程序的编程语言不一致时}\end{cases} \quad (13)$$

式(11)表明并行循环块的代价与模块的并行范例和其宿主程序的并行范例有关，总代价等于模块

本身的代价加上宿主语言和模块间编程范例和运行范例的转换代价.这里 $nump$ 表示参与计算的接点数,式(12)是运行范例的转换代价,当模块需要串行运行时,转换代价就是数据收发的通信代价,当模块也是并行执行时,代价是数据重分布的代价,而当模块采用 Fork-join 模式运行时,就是内存访问控制代价,这和体系结构及内存管理方式密切相关.式(12)是不同编程范例的转换代价,当模块的实现语言与宿主语言不同时,调用时编译器要负责变量存储方式的转变,所以需要考虑内存转换方式的代价.

串行循环模块的代价计算.

串行循环块的代价计算方法,考虑的因素和并行循环块相同.

$$C(seqFOR(i=1\cdots n)\overset{r}{M^p})=n\times(C(\overset{r}{M^p})+C_{\text{R\_paradigm\_trans}}(\overset{r}{M^p})+C_{\text{P\_paradigm\_trans}}(\overset{r}{M^p}) \tag{14}$$

$$C_{\text{R\_paradigm\_trans}}(\overset{r}{M^p})=\begin{cases}0, & r=\sim\\ C(getther)+C(scatter), & r=//\\ C(access\_consistency), & r=\diamondsuit\end{cases} \tag{15}$$

$$C_{\text{P\_paradigm\_trans}}(\overset{r}{M^p})=\begin{cases}0, & \text{当 } p \text{ 与宿主程序编程语言一致时}\\ C(memory\_trans), & \text{当 } p \text{ 与宿主程序的编程语言不一致时}\end{cases} \tag{16}$$

4　应用实例

下面的实例说明在并行化大型应用系统的过程中,应用估算模型确定并行化的可行性并实际验证模型的精确性.

程序 Test 是石油地球物理专家为了检验地震资料处理系统的并行化的效果,研究在 CLUSTER 环境下并行处理的可行性,在实际应用系统中抽取出来的一个串行程序,它很具有典型性.估算模型指导我们恰当确定各模块的并行范例和运行范例.Test 的整个处理过程可以分成 4 个步骤:预处理、FFT、数据处理、反 FFT.完成的功能分别是:(1)预处理:包括计算测线包含的道数、起始索引号、偏置道数和各道的初始样点号等.目的是对原始数据进行读写定位.(2)计算时窗的参数,如时窗宽度、时窗数和交叉平滑系数等.(3)FFT 变换:以测线为单位读取原始数据,根据时窗参数,通过二维 FFT 计算每(测线,时窗)的波谱数据.(4)数据处理:以(时窗,道)为单位,对其中"线×采样点"的二维数据先后进行正反两次 FFT 变换.反 FFT:以线为单位读取波谱数据,将每一线各时窗中的二维波谱数据通过二维 FFT 转换成二维空间数据,并按时窗顺序进行不同时窗数据的迭合.

首先,根据源程序流程,按照前面的描述定义出若干基本模块,给出规范描述,下面是几个模块的描述示例:

initload($IN\ DFILE:file;OUT\ lines:scal,STAK\_BUFM:mat(8,lines)$)

$KN=1,2,\cdots,$测线数 {

getfirstarrive($IN\ KN,STAK\_BUFM:mat(n,Lines);OUT\ DATA\_BUFM:mat(n,lines),KE:scal$)

{$var(Getfirstarrive)=\{DATA\_BUFM,KE,STBUF,INDEX\}$ acc $(Getfirstarrive)=\{KN,STAK\_BUFM,File3\}$}.

在描述的基础上,如果应用波恩斯坦确定并行化的条件,可以得到如下原则:

(1) 对于顺序模块,设两个顺序模块 $M1;M2$ 的引用变量集为 $acc(M1),acc(M2)$,赋值变量集为 $var(M1)$, $var(M2)$,若 $acc(M1)\bigcap var(M2)=\varnothing$ 且 $acc(M2)\bigcap var(M1)=\varnothing$,则 $M1$ 和 $M2$ 的组合可确定为 $M1\parallel M2$.即 $M1$ 和 $M2$ 可以按任意顺序执行.

若 $acc(M2)\bigcap var(M1)\neq\varnothing$,则只能按 $M1;M2$ 的顺序组合.

(2) 对于在循环体内的模块,如果循环不存在携带流相关和反相关,则可利用并行循环 $parFor$ 描述.例如,设循环体内的模块为 $M_1,M_2\cdots,M_n$,循环变量为 $I_1,I_2,\cdots,I_n$.当 $I_1,I_2,\cdots,I_n\notin acc(M_1)$, $acc(M_2),\cdots,acc(M_n)$ 时就可用 $parFor$ 描述.

(3) 对于条件选择结构,用"$F\lhd b\rhd F'$"描述.

这样,可以给出 $test$ 的描述如下:

$test=initiload;\{preprocess1//preprocess2\};comptw;$
　　$process1;process2,$
$process1=$For$(KN=1,2,\cdots,lines)p1cm1,$
$picm1=getfirstarrive;p1cm2,$
$p1cm2=$For$(KL=1,2,3,\cdots,ntw)(fftdataready;$
　　$SLSET1DSPEC),$
$process2=$For$(KL=1,2,\cdots,ntw)p2cm1,$
$p2cm1=$For$(KN=1,2,\cdots,npi)\{p2cm2;SLFXNOISTAPI;p2cm3\},$
$p2cm2=$For$(KT=1,2,\cdots,lines)gettwdata,$
$p2cm3=$For$(KT=1,2,\cdots,lines)puttwdata.$

遗憾的是,按这种方式确定的并行化策略并不一定带来高性能,因为它没有考虑并行化引入的额外代价.每一个模块的并行范例怎样确定才能取得

高性能还依赖于代价估算的结果.

例如,在上面的描述中,对于几个基本模块的组合关系:

$\{initload;\{preprocess1 \parallel preprocess2\}$.

根据估算模型,总代价为

$C(initload)+C(\max(C(preprocess1),C(preprocess2))+C_{R\_paradigm\_trans}(\approx,//)$.

显然,C_R是因为并行化引入的,只有在 *preprocess*1 和 *preprocess*2 计算时间足够大时,才能抵消这个代价,两者并行执行才有意义.从串行源程序分析知道,两者的计算量都不大,所以两者的并行化无意义.应该把几个基本模块 *initload*, *preprocess*1, *preprocess*2 合并为一个串行模块(设为 *init*)更能取得高性能.所以 *test* 的描述可变为

$test=\{init;comptw;process1;process2\}$.

再看组合模块 *process*1 的计算代价,*process*1 包括两重循环,模块 *getfirstarrive* 在第一重循环中,模块 *fftdataready*, *SLSET*1*DSPEC* 在两重循环中.由相关性分析知,两重循环都是可以并行化的,循环体内的模块均以 Local 模式并行运行,所以由式(10),(11)可得

$$C(\overset{//}{process1})=\frac{n}{nump}\max_{1\le i\le n}(C(\{\overset{//}{getfirstarrive};\overset{//}{p1cm1}\})+C_{R\_paradigm\_trans}(\{\overset{//}{getfirstarrive};\overset{//}{p1cm1}\})+C_{P\_paradigm\_trans}(\{\overset{//}{getfirstarrive};\overset{//}{p1cm1}\})).$$

由于 *process*1 模块均采用 Local 模式执行,而且均采用同一种语言编写,所以编程模式和运行模式的转换代价为 0.进一步计算可得 *process*1 的总代价为

$$C(\overset{//}{process1})=\max_{1\le i\le n}\left(\frac{lines}{nump}\times C(\overset{//}{getfirstarrive})+\frac{lines\times ntw}{nump}C(\overset{//}{fftdataready})+\frac{lines\times ntw}{nump}C(\overset{//}{SDSET1DSPEC})\right).$$

由上式可以看出,由于并行化,模块的执行次数比串行缩小了 *nump*(处理器个数)倍,尽管如此,也并不一定保证得到加速比,因为并行化引入了额外代价,由于 *fftdataready* 和 *SDSET*1*DSPEC* 两个过程参数之间无任何相关性,这里主要是由 *getfirstarrive* 引入的代价,由规范描述知过程 *getfirstarrive* 要从文件中读入数据,为了并行需要将其分发到各接点上去,由式(1)得

$$C(\overset{//}{getfirstarrive})=C(compute)+C(distribute).$$

这里,数据的分布代价是一个广播通信的过程,即将数据从一个接点分发到其它所有接点上去的过程,由式(3)得

$$C(process1)=\max_{1\le i\le n}\left(\frac{lines}{nump}\times C(comput\_getfirstarrive)+\frac{lines}{nump}\times\tau\log(nump)+\frac{lines}{nump}\times t_c\times\log(nump)\times b+\frac{lines\times ntw}{nump}C(\overset{//}{fftdataready})+\frac{lines\times ntw}{nump}C(\overset{//}{SDSET1DSPEC})\right).$$

从此式可以明显看出,并行化引入带来的代价,随处理器个数和数据量的增大呈对数增长,但 *SDSET*1*DSPEC* 的计算量随数据量的增加呈指数增长,所以随着数据量增大,并行化的好处将越来越大,模块并行化是值得的.

为了进一步检验计算模型的精确性,对 Test 中的主要模块进行的估算和实际运行比较,实验数据为胜利油田王一地区的两条地震测线,数据量为 580MB 实验环境为四台双 CPU 的 SMP 机器,主频 550MHz,内存每台 1G,100M 网卡;软件环境为 Linux, *p*_HPF 编译环境.计算中没有引入不同语言模块转换的代价.结果如表 1 所示.

表 1 估算模型估算结果与实际结果的比较

模块名	涉及的通信模式及模块组合模式	用模型估算的时间(s)	实际运行时间(s)	误差(%)
Init:数据加载、时窗参数计算等	模块组合:串 通信模式:广播	910	950	4.2
*Process*1:以线为单位在每条线上以时窗为单元进行 FFT 变换	模块组合:并/串 通信模式:Scatter Gether	3667	3986	8
*Process*2:以时窗为单位在每个时窗内以测线为单元进行 FFT 变换	模块组合:并/串 通信模式:Gether	3498	3720	5.96

从上表可以看出,模型具有一定的精确性,几种通信模式的误差在 10%以内.值得指出的是,计算过程中数据通信时间计算用到的参数值(式(2)～(6))的确定是与具体体系结构和协议实现相关的,是一个实验值.因此这里用估算模型计算的结果可能不具通用性.尽管如此,对于给定的系统环境,该模型对验证应用程序并行化的可行性以及确定模块的并行范例、并行策略、通信瓶颈等问题具有指导意义.

5　结论和下一步的工作

多范例并行是大规模并行应用系统的本质特征,研究多范例并行应用系统的规范化描述方法,建立多范例系统的代价估算模型是并行软件工程研究的重要内容之一,本文给出的基于多范例模块组合的描述方法和代价估算模型,为研究串行系统并行化的工程化方法以及并行应用系统的性能优化提供了基础.下一步的研究工作包括,进一步精化代价估算模型,在此基础上,研究如何确定模块的编程范例和运行范例以及如何组合多范例模块使并行系统性能最优的问题,给出多范例并行化大规模串行应用系统的工程化方法,开发交互式串行系统并行化软件工具,提高多范例并行应用系统的开发效率和运行效率.

参　考　文　献

1　Rauber T. A transformation approach to derive efficient parallel implementations. IEEE Transactions on Software Engineering, 2000, 26(4): 289～292

2　Luksch P. Parallel and distributed implementation of large industrial applications. Future Generation Computer Systems, 2000, 16(6): 649～663

3　Nyland L S. A design methodology for data-parallel applications. IEEE Transactions on Software Engineering, 2000, 26(4): 293～314

4　Adve V. Performance Analysis of Data-Parallel Programs. Rice University, Houston, USA: Technical Report CRPC-TR94405, 1994

5　Cremonesi P. Performance evaluation of parallel systems. Parallel Computing. 1999, 25(13～14): 1677～1698

6　Hu Chang-Jun, Yu Hua-Shan, Jiang Wei *et al*. Compiling methods supporting multi-paradigm parallel computing in *p*-HPF. Chinese Journal of Computers, 2001, 24(7): 685～693 (in Chinese)
(胡长军,余华山,姜　伟等. *p*-HPF 支持多范例并行编程的并行编译技术. 计算机学报,2001, 24(7):685～693

7　Hu Chang-Jun, Yu Hua-Shan *et al*. Multi-paradigm parallel computing of *p*-HPF compiler on cluster environment. Journal of Computer Research and Development, 2001, 38(8): 608-615 (in Chinese)
(胡长军, 余华山等. Cluster 环境下 *p*-HPF 支持的多范例编程模式. 计算机研究与发展,2001, 38(8): 608～615)

HU Chang-Jun, born in 1963, Ph. D., professor. His main research interests focus on parallel compiling and parallel computing, domain software engineering and data engineering.

ZHANG Su-Qin, born in 1945, professor. Her main research interests focus on compiling technology, embedded system and software engineering.

TIAN Jin-Lan, born in 1945, associate professor. Her main research interests focus on embedded operation system and software engineering.