并行推理任务的分配方法及其实验分析

冯刚 章萃

(上海复旦大学计算机科学系)

EXPERIMENTAL ANALYSIS OF METHODS FOR PARALLEL INFERENCE TASKS

Feng Gang and Zhang Cui

(*Computer Science Department, Fudan University, Shanghai*)

ABSTRACT

How to allocate the parallel inference tasks among the parallel inference units is an important problem of improving the efficiency of parallel inference systems. This paper takes some examinations of three available allocating methods by expriments, proposes a new strategy and, analyses it theoretically and exprimentally.

摘 要

如何将并行推理任务分配到并行推理部件上去是解决并行推理系统的执行功效的一个重要问题。本文对已提出的三种分配策略进行实验分析并对我们提出的一个新的策略进行理论和实验分析。

§1. 引 言

1989 年 9 月 25 日收到，1989 年 11 月 25 日定稿，得到国家教委年轻教师基金的资助。

随着人工智能的发展，作为它的一个重要工具的逻辑式程序设计，愈来愈得到人们的关注。我们知道，逻辑式程序设计的目的就是将命题的推理过程计算机化。

理论指出，任何可计算的函数可以用完全一阶谓词演算来解决[5]，在此之后消解原则的提出使逻辑的定理证明的计算机化成为现实。在70年代初，人们就认识到一步推理可以看作是一步计算。

目前，人工智能的研究已从主要针对单纯化、假想的问题空间和游戏等转为主要针对人工智能的实际应用—知识工程。人们要求作为工具的语言不仅具有较充分的知识表示能力，而且具有较高的执行功效，于是对逻辑式程序设计语言的执行功效进行了广泛地研究。人们认识到解决这类语言执行功效的关键是解决其计算模型与现有计算机的计算模型的不一致性，而解决这一问题的重要途径就是要开发逻辑程序设计语言的潜在并行性。如针对连接词"and"和"or"，可分别开发"与(and)"并行性和"或(or)"并行性。开发并行性，并在具体的计算中加以实现，无疑会给我们的计算带来较高的执行功效。

迄今，许多研究者针对逻辑语言的特点，诸如，具有模式匹配的功能，具有回溯机能，能进行截断处理，具有符号处理，不确定性，自我扩充的能力等，研究开发了许多不同的并行执行模型，如运用"信道段"概念。这些策略对并行推理有着不同的划分，但无论怎样，它们都需要将这些可并行执行的任务分配到并行推理部件上去。本文即是对已有的分配策略进行实验分析，并对我们新提出的一种策略进行理论与实验的分析，比较优劣。

§2. 分配策略

我们假设并行推理机是由一组紧密耦合的并行推理部件组成的。我们标记这些推理部件为$u_0, u_1, \cdots, u_{m-1}$。它们在推理过程中并行地产生着并行推理任务，并且并行地依照分配策略将产生的并行推理任务分配在$u_0, u_1, \cdots, u_{m-1}$之中。

将并行推理任务分配到并行推理部件上，其本质是一种单值映射

f:{并行推理任务}$\rightarrow$ {并行推理部件}

如果我们记T_j为并行推理任务，则f可表示为

f: $T_j \rightarrow u_i$, 其中$0 \le i \le m-1$

迄今的研究提出了有效地实现并行推理任务分配的两条较为合理的标准[1]:

(1) 并行推理任务在各u_i上的分配力求均匀，此称为均匀度要求。

(2) 并行量力求最大，即力求尽快地使载有任务的u_i尽可能多，此称为高位分布要求。

此外，颇为重要的一点是，并行推理任务的分配是靠大量的通信实现的，所以应尽可能地减少用于分配并行推理任务的通信。因此在考虑以上两条标准的同时应尽可能地减少从一个部件向另一部件的任务分配。

根据这些原则，现在通常有以下三种分配策略，我们标记为A、B、C:

设产生并行推理任务的推理部件为u_i，产生的并行推理任务为$T_0, T_1, \cdots, T_{n-1}$(在此假设$n \ge 1$, 若$n < 1$, 则不存在分配问题):

A. 把T_0分配在u_i上，T_1分配在u_{i+1}上，依此顺序分配，其映射函数为

$f: T_j \rightarrow u_{[(i+j) \bmod m]}$

B. 把T_0分配在u_{i+1}上，T_1分配在u_{i+2}上，依此顺序分配。其映射函数为

$f: T_j \rightarrow u_{[(i+j+1) \bmod m]}$

C. 若$n=0$, 则依A法分配，否则依B法分配。

显然，策略C的用意在于采用策略B的高位分布性质好的特点，而试图去除它在产生任务数为1时的不必要的通信。

我们提出的新的分配策略为(标记为D):

D. 每一并行推理部件都对应着一个固定的并行推理部件，从该固定部件开始顺序分配其产生的并行推理任务。一并行推理部件u_s与其对应的固定的并行推理部件u_t的对应规则为: $t=(k*s) \bmod m$, 也就是说，u_s对应的部件为$u_{[(k*s) \bmod m]}$, 这里k为常数。

于是该分配策略的映射关系为:

$f: T_j \rightarrow u_{[(((k*i) \bmod m)+j) \bmod m]}$

常数k的选择为各推理任务产生的并行推理任务数的平均值。实际中各任务产生的并行推理任务的个数相差是不大的。

作为例子，我们来看看将两组值代入i、j后映射所反映的关系。

i) 当$i=0, j=0$时，有$T_0 \rightarrow u_0$, 也有$S=0, T=0$, 也就是说推理部件u_0对应的推理部件是它自身，u_0它产生的第一号并行推理任务分配在它的对应推理部件u_0上。

ii) 当$i=1, j=1$时，有$T_1 \rightarrow u_{[((k \bmod m)+1) \bmod m]}$, 也有$S=1, t=k \bmod m$, 也就是说，推理部件$u_1$对应的推理部件是$u_{k \bmod m}$, 它产生的第二号并行推理任务分配在推理部件$u_{k \bmod m}$的下一号推理部件$u_{[((k \bmod m)+1) \bmod m]}$上。

策略A、B、C已有了理论分析[1], 现在我们来分析我们提出的策略D。

为便于分析，我们沿用人们对策略A、B、C进行分析时所提出的假设:

1. 运用动态计算树(DCT)的概念[3][6], 各非叶节点产生K个子节点。同时我们假设叶子节点只在最深一层。至于叶节点可能在非最深一层的问题，我们认为，因为在同层节点中，这些叶节点是等可能地分布的，而使我们的假设合情合理;

2. 各并行推理任务的运行时间相同，设为t。

在时刻$n_1 * t$, 这里n_1满足$K^{n_1-1} < m$及$K^{n_1} \geq m$, 各并行推理部件所载的任务数为

$$|u_i| = \sum_{j=0}^{n_1-1} S_g(K^j - i) + \left[\frac{K^{n_1}}{m}\right] + S_g((K^{n_1}/m) - i)$$

$$= \sum_{j=0}^{n_1} S_g((k^j/m) - i) + \left[\frac{K^{n_1}}{m}\right]$$

其中

$$s_g(x) = \begin{cases} 1, & x > 0 \\ 0, & x \leq 0 \end{cases}$$

"[ ]" 表示整除

"|" 表示取余

我们记此值为C_i，即$C_i=\sum_{j=0}^{n_1}S_g((k^j/m)-i)+[\frac{K^{n_1}}{m}]$，我们看到对给定的$i$，$C_i$为定值。

在时刻$n*t$ $(n>n_1)$，各并行推理部件所载的并行推理任务数为

$$|u_i|=(n-n_1)*K+C_i$$

我们注意到上式右边的项$(n-n_1)*K$与i无关，也就是说在时刻n_1之后，分布是完全均匀的；同时由C_i的表达式看到该分配策略对高位推理部件是以指数级增长的，而前三种策略是以线性级增长的，也就是说它对高位公布是更好地满足的。

因此，我们认为在理论上策略D较策略A、B、C，无论是均匀度还是高位分布都是较优的。

§3. 实验摸拟

本实验是以前述的两条分配原则为准绳，用顺序执行来摸拟并行推理的实现过程，比较分析A、B、C、D四种分配策略。

对均匀度的检查，我们是对每一逻辑时刻计算以下值

$$\text{并行推理部件的负载差}=\sum_{i=0}^{m-1}abs\left(|u_i|-\frac{\sum_{j=0}^{m-1}|u_j|}{m}\right)$$

也就是计算各部件所载任务数与平均任务数的绝对差值之和。此值越大说明均匀性越差。

对高位分布的检查是通过计算各时刻没有载有任务的推理部件的数目来衡量的。此值越大说明高位分布性越差。

在以往的研究中通常假设[3][6]：

1. 各父节点产生K个子节点，即每个推理部件一步完成后，若产生并行推理任务则产生的数目为K；

2. 各并行推理任务的运行时间相同。

而我们的实验允许各父节点产生的子节点数不相同，可以为0，也可以为一较大的数，另外也允许各并行任务的运行时间不同。再者，我们的模拟只分析动态计算树在其生长过程中并行推理机的均匀度情况和高位分布情况。

当一并行推理部件因载有任务而开始工作时，我们赋予它当前运行的任务以一个将产生的并行推理任务数和它自身的运行时间，这两个数值都取自随机数产生器(在实验中我们对随机数的取值范围作了不失一般性的限制)，其中运行时间是整数值，即是逻辑时钟的整数单位。

在每一逻辑时刻运行完被处理的推理任务的推理部件完成可将被产生的并行推理任务的分配。我们此时做的工作为：计算并行推理部件的负载差，计算没有载有任务的推理部件的数目。

我们分别选择了512、256和16作为并行推理部件数进行摸拟分析，512和256对应着VLSI技术构成的并行推理系统，16对应着用现有机器模拟构成并行推理系统。

我们的模拟结果由图1-6表示。

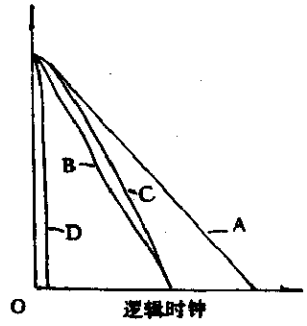

图 1　m=512 时的高位分布

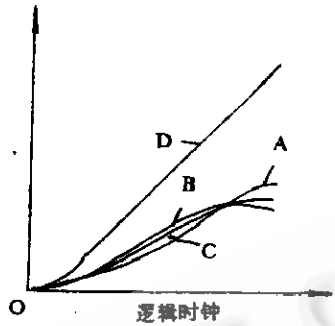

图 2　m=512 时的均匀度

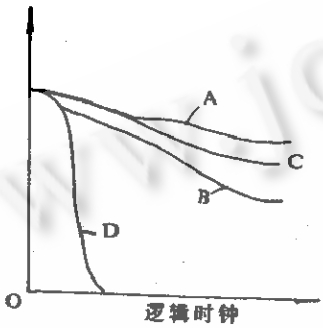

图 3　m=256 时的高位分布

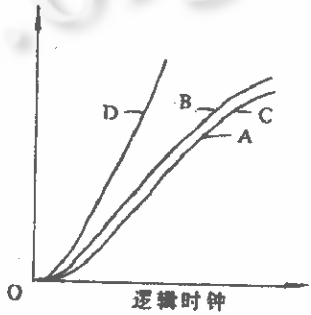

图 4　m=256 时的均匀度

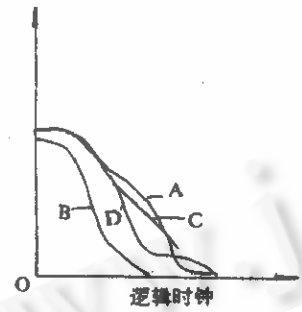

图 5　m=16 时的高位分布

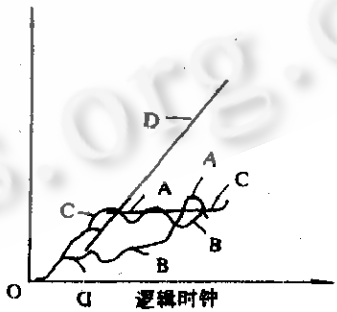

图 6　m=16 时的均匀度

§4. 分析和结论

我们来看图1。这里的条件是推理部件数$m = 512$。我们看到各策略使得没有任务的推理部件数减少的速度是有明显的差别的：策略D减少的速度最快，策略B、C次之(它们相差无几)，而A减少的速度最慢。

至于均匀度方面，我们可看图2。这里m=512。我们看到这四种策略所反映出的

性质相差无几。我们注意到这与我们在前面的分析结果有些不同，在那里我们指出策略D 在时刻$n_1 * t$ 之后，分配将完全均匀。这不同的原因在于我们在实验中抛弃了理论分析的假设(各任务的运行时间相同和产生的并行任务数相同)使得我们有现在的结果。

其它各图(分别对应着m=256, m=16 的情形)表明的事实与m=512 时相同，除了图5。在那里由于m=16, 推理部件太少而未明显体现出策略D 的优越性。

总之，我们认为策略D, 无论就均匀度．还是高位分布都具有很好的性质。

策略A 的功效性最差。策略B 和C 在均匀度和高位分布两方面的性质类似，几乎没有什么差别。直观意义上，策略B 比策略C 在均匀度和高位分布方面都应是较好的，但实验中并行推理任务数的随机性使这一点不明显。

因此，我们认为，策略D 的执行功效最高，它不仅能很好地符合我们前述的两条原则，同时由于它的分配地址的不变性(各推理部件u_i 对应着不变的起始分配的推理部件)并不增加通信的复杂。

至于策略B 和C, 我们认为C 较优，因为它需要的通信比B 少，而其他方面不差于B。

策略A 在高位分布和均匀度两方面虽较差，但它有通信量少这样一个优点。这是由于它使得推理部件产生的并行推理任务始终由自身开始分配，这同其它无此特点的策略相比较，它减免了将任务往自身分配的那次通信。

本文关于并行推理任务分配的实验分析具有普遍意义，尽管有些系统采取不同的开发并行性的策略。

参考文献

[1] 赵沁平，“从抽象ENGINE 到KLND—ENGINE”，《计算机研究与发展》，1988 年，第25 卷，第3 期。

[2] 徐家福、章萃、赵沁平，“并行推理系统NDPIS 的设计”，《计算机研究与发展》，1988 年，第25 卷，第3 期。

[3] 赵沁平，“并行推理系统NDPIS 中抽象机KLND—ENGINE 的设计与分析”，南京大学博士学位论文，1986. 6。

[4] 高全泉，“新颖的人工智能语言—PROLOG”，《计算机科学》，1983 年，第6 期。

[5] Sten—Åke Tärnlund, “Logic Programming—from a Logic Point of View”, 《Proceedings of Luternational Symposium on Logic Programming》, 1986。

[6] 章萃，“并行推理系统NDPIS 中抽象机KLND—AM 的设计与分析”，南京大学博士学位论文，1986.6。