随机算法异步并行化的效率分析*

徐 云[1+], 陈国良[1], 张强峰[1], 顾 钧[1,2]

[1](中国科学技术大学 计算机科学与技术系,安徽 合肥 230027)

[2](香港科学技术大学 计算机科学系,香港)

Efficiency Analysis of Asynchronic Parallelization of Randomized Algorithms

XU Yun[1+], CHEN Guo-Liang[1], ZHANG Qiang-Feng[1], GU Jun[1,2]

[1](Department of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)

[2](Department of Computer Science, Hong Kong University of Science and Technology, Hong Kong, China)

+ Corresponding author: Phn: 86-551-3603145, Fax: 86-551-3601013, E-mail: xuyun@ustc.edu.cn

http://nhpcc.ustc.edu.cn

Received 2002-04-22; Accepted 2002-12-27

Xu Y, Chen GL, Zhang QF, Gu J. Efficiency analysis of asynchronic parallelization of randomized algorithms. *Journal of Software*, 2003,14(5):871~876.

http://www.jos.org.cn/1000-9825/14/871.htm

Abstract: The uncertainty of running time of randomized algorithms provides a better opportunity for asynchronized parallelization. There are many computing experiments verifying that the asynchronized parallelizating acceleration of randomized algorithms are linear or even superlinear. For randomized algorithm RDP solving for SAT (satisfiability) problem, the relation among efficiency of asynchronized parallelization, distribution of running time and number of processors are investigated. In this paper, a model of piecewise-linear distribution is applied to simulate the running time distribution of randomized algorithms. This model of distribution is a kind of single peak. Both theoretical analysis and computing experiment indicates that asynchronied parallelization of randomized algorithms are of near linear acceleration when the processors are less and the single peak is located near the front of running time distributions.

Key words: randomized algorithm; asynchronic parallelization; distribution of running time; SAT problem; NP-complete problem

摘 要: 随机算法的执行时间具有不确定性,这种不确定性为随机算法的异步并行提供了良好的基础,已有许多计算实验表明了随机算法的异步并行可以达到线性甚至超线性的加速.对于求解 SAT 问题的随机算法 RDP,研究了异步并行效率与运行时间分布和处理器数目之间的关系.应用一种单峰分布——分段线性分布模型来模拟随机算法的运行时间分布.理论分析和计算结果均表明:当处理器数目 k 较小和单峰位于分布的前部时,随

* Supported by the National Grand Fundamental Research 973 Program of China under Grant No.G1998030403 (国家重点基础研究发展规划(973)); the High-Level University Construction Foundation of the Chinese Academy of Sciences under Grant No.KYZ2706 (中国科学院高水平大学建设项目)

第一作者简介: 徐云(1960－),男,浙江宁波人,博士,副教授,主要研究领域为高性能计算,并行算法设计与分析.

机算法的异步并行具有近线性加速.

关键词: 随机算法;异步并行;运行时间分布;SAT 问题;NP 完全问题
中图法分类号: TP301 文献标识码: A

并行计算和随机化方法是算法设计中的重要方法和手段,它们的结合可以实现串行确定性算法难以达到的计算效率,在一些实际问题中得到了迄今最好的算法.例如,匹配问题(matching problem)目前尚不存在 NC 类的确定性算法,而 Karp 等人首次构造了 NC 类的并行随机算法[1].对于像 SAT 问题一类的 NP 完全问题,并行计算尽管不能改变最坏情形的时间复杂度,但是可以显著地增加求解问题的规模[2],并且是一些大规模实时问题的惟一解决方案(如机器人的路径规划、机器翻译、任务调度等).随着计算机技术的发展和经济能力的提高,并行计算和随机化方法在 NP 完全问题和 NP 难解问题中的应用会越来越普及.

异步并行化具有易于实现、结构简单等良好特性,成为近几年的研究热点[3~11].这种方法主要是由 p 个相同或不同的核心算法同时地独立运行,充分利用了竞争、重启和随机化方法等技术.大量的计算实验表明,随机算法的异步并行效率与运行时间分布之间是密切相关的,究竟何种运行时间分布有利于异步并行,以及最优的异步并行处理器数目的确定等是其中的关键问题.Luby,Hogg 和 Gomes 等人研究了对数正态分布和尾重(tailor heavy)分布的随机算法异步并行化[3~8],并指出,当处理器数目较小时,可以达到较高的加速比.金人超和黄文奇在文献[10]中使用了一个非常简单的运行时间分布模型——线性分布,其理论分析指出,异步并行化可以产生线性加速,并行效率约为 2/3,并在处理器数目较小时(k≤10)得到了实验验证.

实际上,随机算法在求解 SAT 等问题时,其运行时间分布具有多样性和复杂性,以往的工作只研究了一些典型的运行时间分布,并没有从整体上进行探讨.本文提出了一个分段线性分布模型,这是一种简单的单峰分布,可以近似地模拟随机算法的各种运行时间分布.像对数正态分布和线性分布,只是随机算法运行时间分布较特殊的情形,在分段线性模型中,它们分别对应于单峰位置位于分布的前部和起始点的情形,我们的研究推广了已有的工作结果,不仅给出了异步并行的有效区,同时也指出了异步并行的无效区,为随机算法的异步并行提供了理论依据和实验验证.

1 随机算法 RDP 及其异步并行化

本节首先描述求解 SAT 问题的随机算法 RDP,然后给出其异步并行化的方法.由于确定性算法 DPLL[12,13]仍然是最有效的完全算法之一,所以当前许多有效的完全算法是 DPLL 算法的变种.我们在 DPLL 算法中引入随机拆分文字策略(randomized splitting literal strategy),便产生了随机版的 DPLL 算法 RDP.随机算法 RDP 的算法描述如下:

Procedure RDP(formula F)
Begin
1. **if** F is an empty formula **then return** Yes;
2. **if** F contains an empty clause **then return** No;
3. (Unit propagation) **if** F contains a unit clause $\{l\}$ **then**
return RDP(F [l/true]);
4. (Pure literal) **if** F contains a literal l but not the negation $\neg l$ **then**
return RDP(F [l/true]);
5. select an unassigned literal l and an Boolean value T at random**;**
6. (Splitting) **if RDP**(F [l/T])=Yes **then return** Yes;
else return RDP(F [l/$\neg T$]);
End

随机算法 RDP 只是在第 5 和第 6 句与 DPLL 算法不同,其他语句完全相同.随机算法 RDP 的异步并行化就

是将 k 个随机算法 RDP 置于 k 个处理器中同时执行,如果有某个处理器求出 SAT 问题的解,则通知其他处理器终止整个算法的执行.这种计算模式可以在单机上同时使用多个进程或重复使用单个进程来真实地模拟.有时为了方便起见,并行算法的加速比常常与某个对应的串行算法进行比较,而不是与最快的串行算法相比较,本文中提出的异步并行随机算法就是与随机算法 RDP 在平均运行时间的意义上进行比较的.

2 理论分析

随机算法 RDP 每次执行对应的搜索树都是不同的,运行时间也不同.对于一个固定的 SAT 实例,运行时间服从某个分布,但是对于不同的 SAT 实例,它们的运行时间分布一般是不同的.由于运行时间分布的形状非常复杂,为了能够进行运行时间分布与异步并行效率的分析,我们对运行时间分布作了适当的简化.用一个分段线性分布近似地模拟随机算法 RDP 的运行时间分布,这种分段线性分布要比线性分布和对数正态分布包含更多的实际运行情况.分段线性分布是一种单峰分布,文献[5,10]中的计算实验结果也说明了这种分布,对于线性分布和对数正态分布可以视为分段线性分布的特殊情况(分别对应于单峰位置在起始点和前部).下面给出随机算法 RDP 运行时间分布的示意图(如图 1 所示).

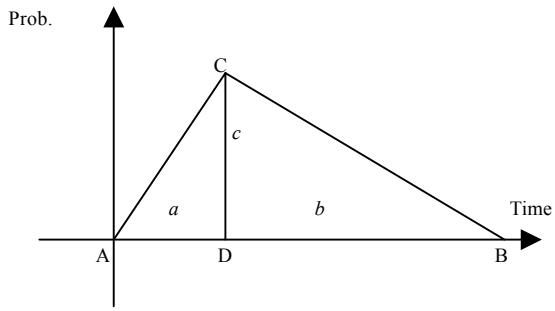

Fig.1 Running time distribution of randomized algorithm RDP

图 1 随机算法 RDP 运行时间分布

在图 1 中,横轴表示执行时间,纵轴表示时间的分布概率,c 是分布的峰高,a,b 为峰的位置(a 为前偏移量,b 为后偏移量).

首先,图 1 所示的两段函数为

$$\text{AC: } y=\frac{c}{a}x,\ 0\le x\le a, \tag{1}$$

$$\text{BC: } y=-\frac{c}{b}x+\frac{ac}{b}+c,\ a\le x\le a+b. \tag{2}$$

由概率密度函数 $f(x)$的特性,有

$$1=\int_0^{a+b}f(x)\mathrm{d}x=\int_0^a\frac{c}{a}x\mathrm{d}x+\int_a^{a+b}\left(-\frac{c}{b}x+\frac{ac}{b}+c\right)\mathrm{d}x=\frac{1}{2}ac+\left(-\frac{c}{2b}x^2+\frac{ac}{b}x+cx\right)\Big|_a^{a+b}=\frac{c}{2}(a+b).$$

因此

$$c(a+b)=2. \tag{3}$$

由式(1)~(3),概率密度函数 $f(x)$ 可以写为

$$f(x)=\begin{cases}\dfrac{c}{a}x, & 0\le x\le a\\ -\dfrac{c}{b}x+\dfrac{2}{b}, & a<x\le a+b\end{cases}, \tag{4}$$

则一次执行算法 RDP 的运行时间不超过 t 的概率为

$$\Pr[x\le t]=\int_0^t f(x)\mathrm{d}x=\begin{cases}\displaystyle\int_0^t\frac{c}{a}x\mathrm{d}x, & 0<t\le a\\ \displaystyle\int_0^a\frac{c}{a}x\mathrm{d}x+\int_a^t\left(-\frac{c}{b}x+\frac{2}{b}\right)\mathrm{d}x, & a<t\le a+b\end{cases}=\begin{cases}\dfrac{c}{2a}t^2, & 0<t\le a\\ -\dfrac{c}{2b}t^2+\dfrac{2}{b}t-\dfrac{a}{b}, & a<t\le a+b\end{cases}. \tag{5}$$

进一步地,k 次执行算法 RDP 的运行时间(即 k 次中最短的一次执行时间)不超过 t 的概率为

$$\Pr^{(k)}[x\le t]=1-(1-\Pr[x\le t])^k, \tag{6}$$

则 k 次执行算法 RDP 的运行时间为 t 的概率为

$$f^{(k)}(t)=\frac{\mathrm{d}}{\mathrm{d}t}(\Pr^{(k)}[x\le t])=k\cdot(1-\Pr[x\le t])^{k-1}\cdot\frac{\mathrm{d}}{\mathrm{d}t}(\Pr[x\le t])=\begin{cases}\dfrac{c}{a}kt\cdot\left(1-\dfrac{c}{2a}t^2\right)^{k-1}, & 0\le t\le a\\ k\cdot\left(-\dfrac{c}{b}t+\dfrac{2}{b}\right)\cdot\left(\dfrac{c}{2b}t^2-\dfrac{2}{b}t+\dfrac{a}{b}+1\right)^{k-1}, & a<t\le a+b\end{cases}. \quad (7)$$

因此,执行一次算法 RDP 的运行时间的期望是

$$E[t]=\int_0^{a+b}t\cdot f(t)\mathrm{d}t=\int_0^a\frac{c}{a}t^2\mathrm{d}t+\int_a^{a+b}\left(-\frac{c}{b}t^2+\frac{2}{b}t\right)\mathrm{d}t=\frac{2}{3}a+\frac{1}{3}b, \quad (8)$$

执行 k 次算法 RDP 的运行时间的期望是

$$E^{(k)}[t]=\int_0^{a+b}t\cdot f^{(k)}(t)\mathrm{d}t, \quad (9)$$

则 k 台处理器异步并行执行的加速比为

$$S_k=E[t]/E^{(k)}[t]=\left(\frac{2}{3}a+\frac{1}{3}b\right)\Big/\int_0^{a+b}t\cdot f^{(k)}(t)\mathrm{d}t. \quad (10)$$

由式(10)可知,并行加速比涉及 a,b,c,k 这 4 个参数.为了研究单峰位置对并行加速比的影响,我们设

$$a=\lambda b, \quad (11)$$

这里, $0<\lambda<1$, $\lambda=1$ 和 $\lambda>1$ 分别对应于单峰位置在分布的前部、中部和尾部的情况.结合式(3),a,b 可以用 λ 和 c 来表示

$$\begin{cases}a=\dfrac{2\lambda}{(1+\lambda)c}\\ b=\dfrac{2}{(1+\lambda)c}\end{cases}. \quad (12)$$

将式(12)代入式(10),并进行积分得到

$$S_k=\frac{(1+2k)\cdot(1+\lambda)\cdot(1+2\lambda)}{3(1+\lambda)\cdot\left(\left(\dfrac{1}{1+\lambda}\right)^{k-1}-2k\lambda\left(\dfrac{1}{1+\lambda}\right)^{k}\right)+2k\lambda^2(1+2k)\cdot Hypergeometric2F1\left[\dfrac{3}{2},1-k,\dfrac{5}{2},\dfrac{\lambda}{1+\lambda}\right]}. \quad (13)$$

这里,*Hypergeometric*2F1 是一种超几何函数,其积分形式是

$$Hypergeometric2F1(a,b,c;z)=\Gamma(c)/[\Gamma(b)\Gamma(c-b)]\times\int_0^1t^{b-1}(1-t)^{c-b-1}(1-tz)^{-a}\mathrm{d}t. \quad (14)$$

由计算结果式(13)可以看出,加速比的值与 c 无关,而只与 λ 和 k 相关.我们分别画出了加速比的变化曲面(如图 2 所示)和 λ=1/16 以及 λ=1 时加速比曲线对照图(如图 3 所示).

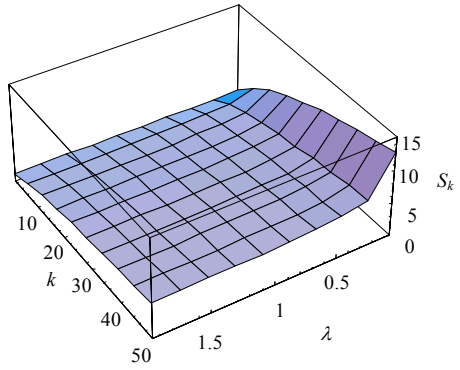

Fig.2 Variation of speedup

图 2 加速比的变化曲面

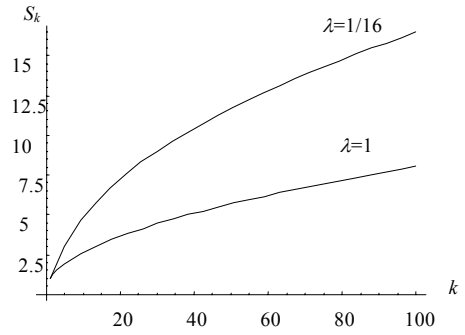

Fig.3 Comparison of speedup (λ=1/16, λ=1)

图 3 加速比曲线对照(λ=1/16 和λ=1)

从图 2 可以看出,加速比受 λ 的影响很大,当 λ 较小时(λ<0.25)有显著的加速,而 λ 的其他区域加速缓慢且变化不大.由图 3 可以看出,单峰位于前部的加速比要明显好于位于中部和尾部时的情形,对于 k<10 且 λ 较小时,有近线性的加速.

3 计算实验

我们随机生成了两个 3-SAT 实例(m=430,n=100),实例 1 和实例 2 分别代表了单峰在前部和中部的情形.按照处理器数目为 1,5,10,20,30,40 和 50,分别对实例 1 和实例 2 进行了异步并行计算,得出的加速比和并行效率见表 1 和表 2.

Table 1 Parallel efficiency and speedup of instance 1

表 1 实例 1 的加速比和并行效率

Number of processors	k=1	k=5	k=10	k=20	k=30	k=40	k=50
Average running time	0.363 1	0.075 3	0.056 4	0.053 2	0.051 6	0.050 0	0.050 0
Speedup		4.822 3	6.438 3	6.825 6	7.037 2	7.262 4	7.262 4
Parallel efficiency		0.964 5	0.643 8	0.341 3	0.234 6	0.181 6	0.145 2

Table 2 Parallel efficiency and speedup of instance 2

表 2 实例 2 的加速比和并行效率

Number of processors	k=1	k=5	k=10	k=20	k=30	k=40	k=50
Average running time	10.494 4	9.011 3	8.534 6	4.105 8	2.554 0	1.881 4	1.540 8
Speedup		1.164 6	1.229 6	2.556 0	4.109 0	5.578 0	6.811 0
Parallel efficiency		0.232 9	0.123 0	0.127 8	0.137 0	0.139 4	0.136 2

由表 1 和表 2 中的计算结果可以看出:

(1) 实例 1 的加速比或并行效率明显好于实例 2,说明单峰位置靠前的有较好的并行效率;

(2) 两个实例都说明,当处理器数目较大时(k>20),加速比增长缓慢,并行效率较低.

4 结 语

本文的计算实验验证了理论分析的结果,因此对于随机算法 RDP 的异步并行化,可以得出以下结论:

(1) 当单峰位置较前以及处理器数目较少时,有较好的并行加速,即当 λ≤0.25 和 k≤20 时有接近线性的加速,为加速有效区;

(2) 当单峰位置靠后或处理器数目较大时,并行加速较差,即对于 λ>0.25 和 k>20 的区域加速缓慢,为加速无效区.

本文应用分段线性分布来模拟随机算法的运行时间分布,要比文献[10]中的线性分布代表更多的实际情况,所得结论对于随机算法的异步并行化有着实际的指导意义.由于随机算法运行时间分布的多样性和复杂性,分段线性模型还是一种简单的近似模型,需要进一步研究可行的更准确和更符合实际的模型.

致谢 在此,我们向微软亚洲研究院的谢幸博士以及对本文的工作给予支持和建议的同行表示感谢.

References:

[1] Karp RM, Upfal E, Wigderson A. Constructing a perfect matching is in random NC. Combinatorica, 1986,6(1):35~48.

[2] Gu J, Purdom PW, Franco J, Wah BW. Algorithms for satisfiability (SAT) problem: A survey. DIMACS Series in Discrete Mathematics and Theoretical Computer Science, American Mathematical Society, 1997,35:19~151.

[3] Luby M, Sinclair A, Zuckman D. Optimal speedup of Las Vegas algorithms. Information Processing Letters, 1993,47:173~180.

[4] Luby M, Ertel W. Optimal parallelization of Las Vegas Algorithms. Symposium on Theoretic Aspects of Computer Science, 1994, 463~475.

[5] Hogg T, Williams C. Expected gains from parallelizing constraint solving for hard problems. In: Proceedings of the AAAI-94. Seattle: AAAI Press. 1994. 331~336.

[6] Gomes C, Selman B, Crato N, Kautz H. Heavy-Tailed phenomena in satisfiability and constraint satisfaction problems. Journal of Automated Reasoning, 2000,24:67~100.

[7] Gomes C, Selman B, Crato N. Heavy-Tailed distributions in combinatorial search. In: Proceedings of the Constraint Programming (CP'97). Linz: Springer-Verlag, 1997. 121~135.

[8] Gomes C, Selman B, Kautz H. Boosting combinatorial search through randomization. In: Proceedings of the AAAI-98. Madison: AAAI Press, 1998. 430~437.

[9] Chen GL, Xie X, Xu Y, Gu J. Designing of restart strategy for randomized algorithms and its application in solving the TSP. Chinese Journal of Computers, 2002,25(5):514~519 (in Chinese with English abstract).

[10] Jin RC, Huang WQ. Parallel computing: An effective method for improving the efficiency of solving SAT problems. Journal of Software, 2000,11(3):398~400 (in Chinese with English abstract).

[11] Xu Y. Studies on randomized algorithms for SAT problem and its phase transition phenomenon [Ph.D. Thesis]. Hefei: University of Science and Technology of China, 2002 (in Chinese with English abstract).

[12] Davis M, Putnam H. A computing procedure for quantification theory. Journal of the ACM, 1960,7:201~215.

[13] Davis M, Logemann G, Loveland D. A machine program for theorem proving. Communication of the ACM, 1962,5(7):394~397.

附中文参考文献:

[9] 陈国良,谢幸,徐云,顾钧.随机算法重启策略的构造及其在 TSP 中的应用.计算机学报,2002,25(5):514~519.

[10] 金人超,黄文奇.并行计算:提高 SAT 问题求解效率的有效方法.软件学报,2000,11(3):398~400.

[11] 徐云.SAT 问题的随机算法及其相变现象研究[博士学位论文].合肥:中国科学技术大学,2002.

2003 全国软件与应用学术会议(NASAC 2003)

征 文 通 知

由中国计算机学会软件工程专业委员会主办,上海交通大学计算机系承办,北京大学、北京航空航天大学、复旦大学、国防科技大学协办的 2003 全国软件与应用学术会议将于 2003 年 11 月 14~16 日在上海召开。届时将进行软件工程等方面的技术与应用交流,会议将出版正式论文集,并将优秀论文推荐到核心学术刊物(EI 检索源)发表。欢迎大家踊跃投稿。

一、征文范围(包括但不限于)

需求工程、软件过程、质量保障、软件工具与环境、软件工程实践、软件工程教育、操作系统、中间件、软件复用、软件语言、应用软件,等。

二、论文要求

1．论文未曾在其它杂志、会议上发表或录用

2．论文长度：每篇限定在 6 页(A4)内

3．请以 PDF 或者 PS 格式提交论文。有关文章的版心、字号、题目、各级标题、格式及参考文献格式与《软件学报》相同，具体模板请参阅如下网址 http://www.jos.org.cn 中的“相关网站”一栏

三、重要日期

文稿截止日期:2003 年 8 月 15 日

论文录用通知日期:2003 年 9 月 20 日

四、联系方式

200030 上海交通大学计算机系 李明禄

E-mail: li-ml@cs.sjtu.edu.cn

关于会议更详细内容请访问:http://www.cs.sjtu.edu.cn/nasac2003/