一种基于优先级表的实时调度方法

李 毅[1], 武君胜[1], 樊 晨[2]
(1. 西北工业大学 软件与微电子学院, 西安 710072; 2. 中山大学 翻译学院, 广东 珠海 519082)

摘 要: 针对单处理器实时系统动态调度问题进行了研究,分析任务的到达时间、执行时间、截止时间和空闲时间等任务属性的敏感度和影响度,提出了一种基于优先级表的调度算法 PTBM,使相对截止期越小、空闲时间越大的任务优先级越高。对关于实时任务属性敏感度和影响度的结论验证和与传统的 EDF、LLF 和 PTD 算法的对比进行仿真实验,仿真结果表明基于优先级表设计的实时调度算法 PTBM 具有较高的调度成功率。该方法可应用于实时系统的实时任务的动态调度中。

关键词: 实时系统; 动态调度; 任务属性; 敏感度; 影响度; 优先级表; 调度成功率
中图分类号: TP18;TP301.6 **文献标志码**: A **文章编号**: 1001-3695(2014)05-1410-04
doi:10.3969/j.issn.1001-3695.2014.05.030

Real-time scheduling algorithm based on priority table

LI Yi[1], WU Jun-sheng[1], FAN Chen[2]
(1. *School of Software & Microelectronics, Northwestern Polytechnical University, Xi'an* 710072, *China*; 2. *School of International Studies, Sun Yat-sen University, Zhuhai Guangdong* 519082, *China*)

Abstract: Focused on dynamic scheduling of single-processor real-time system, this paper defined and analyzed the sensitivity and influence of task attributes including arrival time, execution time, deadline and laxity, then it proposed a scheduling algorithm PTBM based on priority table, which made that a task with small deadline and large laxity had higher priority. Compared with PTBM with EDF, LLF and PTD, simulated results verify CMRN of the sensitivity and influence, and show PTBM outperforming on scheduling success ratio. The proposed algorithm can be applied to dynamically schedule real-time tasks in real-time systems.

Key words: real-time system; dynamic scheduling; task attributes; sensitivity; influence; priority table; scheduling success ratio

实时系统是一个用来处理具有定时限制工作负载的任务处理系统[1]。典型的实时系统应用包括飞行控制、信号处理和远程通信系统等。例如,在战场环境下,要求通信系统能够及时地传递战场信息,接收和发送不同作战方的指挥信息,信息传递不及时可能导致战役的失利。

实时调度算法主要分为两大类,即静态调度和动态调度。典型的静态调度算法有 rate monotonic(RM)和 deadline monotonic(DM),但其调度成功率在理论和实践中都低于动态调度。关于动态调度的研究有最早截止期最优先(earliest deadline first,EDF)[2,3]、最小空闲时间最优先(least laxity first,LLF)[3]、最早放行最优先[3]、最早可达截止期最优先[3]、最高价值最优先[4]、最大价值密度最优先[4]等算法。其中,EDF 和 LLF 是理论上最佳的单处理器调度算法,且它们是等价的[5]。文献[6]综合考虑了截止期和空闲时间来设计任务优先级,提出了一种基于优先级表的调度算法 PTD(priority table design)。在实践中,EDF、LLF 应用和讨论得最为广泛[7~10]。

上述算法除了 PTD 算法都只考虑了一个特征属性,虽然在特定情况下调度性能比较理想, 但有时不能唯一确定任务优先级,总体调度性能不高,算法适应能力不强,应用价值偏低[11];PTD 算法考虑了两个特征属性对优先级的影响,但仅在定性上进行了分析,没有定量计算。为此,本文给出了在影响度最大(即调度成功率最大)情况下的一种优先级表设计方法 PTBM。

1 敏感度和影响度

定义 1 在单处理器系统中,可以对任务模型进行统一描述为 τ(arri,deadline,exec)。其中,arri 表示任务 τ 的到达时间,deadline 表示任务 τ 的截止时间,exec 表示任务 τ 的执行时间。在本文中,假设一旦任务到达即可以被系统进行调度执行,即任务的就绪时间等于到达时间。因此任务 τ 的属性里不包括就绪时间。

定义 2 对于一个特定的实时调度算法,算法的调度成功率对于任务属性的敏感程度不尽相同,简称敏感度 s(attr)。其中,attr 表示任务的属性。敏感度是指任务属性呈不同规律变化时对调度成功率影响的剧烈程度。

定义 3 对于一个特定的实时调度算法,不同的任务属性对于调度成功率的影响程度不同;即使对于相同的任务属性,不同的变化规律对算法的调度成功率的影响程度也不相同。

收稿日期: 2013-07-10; **修回日期**: 2013-08-21

作者简介:李毅(1988-),男,陕西咸阳人,硕士研究生,主要研究方向为企业应用、计算方法(437595856@qq.com);武君胜(1962-),男,陕西西安人,教授,博士,主要研究方向为应用软件、计算方法、科学计算可视化、软件工程;樊晨(1989-),女,河南周口人,硕士研究生,主要研究方向为英语翻译.

定义任务属性的影响度为 f,$f(\overrightarrow{attr})$ 和 $f(\overleftarrow{attr})$ 分别表示属性 attr 增大和减小时对算法调度成功率的影响程度。其中→表示增大,←表示减小。

1.1　敏感度分析

任务的截止时间属性对调度成功率影响最大。任务的执行时间属性不是孤立存在的,它与截止时间和到达时间联系紧密,实际上受截止时间和到达时间的隐性影响,即任务必须满足 exec < = deadline - arri 的约束。任务到达时间只表明了任务可以被调度执行的条件,它与任务其他属性的联系最为松散,对任务能否被成功调度影响较小。基于上述分析,给出不同属性对调度成功率的敏感度对比为

$$s(\text{deadline}) > s(\text{exec}) > s(\text{arri}) \tag{1}$$

在仿真实验部分,将会证明上述结论。

1.2　影响度分析

对于一个特定的实时调度算法,任务的不同属性对调度成功率的影响程度也不相同;即使相同的任务属性,不同变化规律对算法的调度成功率的影响程度也存在差异。当任务的截止时间相同时,任务的执行时间最短优先执行,在保证尽快完成任务的条件下可以节省处理器时间。而在相同截止时间下,后到达的任务更加紧迫,而且执行时间也较短。基于上述分析,给出不同属性、不同变化规律下任务属性对调度成功率的影响度:

$$f(\overleftarrow{\text{deadline}}) > f(\overleftarrow{\text{exec}}) > f(\overrightarrow{\text{arri}}) > f(\overleftarrow{\text{arri}}) > f(\overrightarrow{\text{exec}}) > f(\overrightarrow{\text{deadline}}) \tag{2}$$

影响度越高,调度成功率越大,证明见仿真实验部分。

2　动态调度与优先级表

2.1　动态调度

动态优先级调度是指任务的优先级在调度期内随着调度序数动态变化的。在可抢占式单处理器调度环境下,EDF 算法在调度任意独立的实时任务集合时,是一种最佳的调度算法。EDF 算法在每个调度时刻将最高优先级赋予截止期最小的活动任务,并且如果一个任务集合是可调度的,则 EDF 调度算法可以保证该集合中所有任务的截止时间[5]。LLF 算法是另外一种最佳调度算法;一个任务的空闲时间定义为该任务可以继续等待的最大时间。LLF 算法将最高优先级赋予空闲时间最小的任务。单处理器实时系统常采用二维优先级表作为任务优先级的数据存储结构,动态优先级调度的核心是优先级表的设计。优先级表的设计需要考虑实时任务的不同属性,如截止时间、执行时间和到达时间等。在第 1 章,得到不同的任务属性对调度成功率的影响程度不同。此外,任务的优先级仅由某一个任务属性确定是不够的[12],采用多特征综合调度算法可以提高任务的可调度性和稳定性[7]。

每个调度周期内,调度算法一般直接比较任务的相对截止时间。相对截止时间 dl = deadline - curr,其中 curr 是调度时刻。对于所有任务来说,curr 是相同的。所以,dl 与 deadline 的值是一致的,使用相对截止时间并不会破坏算法的正确性。对于任务的执行时间和到达时间应该直接使用绝对时间。对于选定的两个任务属性作为优先级表的参考因素,一般情况下应将参数的取值范围划分为若干个不同的取值区间,每个区间选择一个典型值代表该区间。假设选定的任务属性是相对截止时间 d 和执行时间 c,截止时间 d 有 m 个典型值 $d_1, d_2, \cdots, d_m$,其中 $d_1 < d_2 < \cdots < d_m$,执行时间 c 有 n 个典型值 $c_1, c_2, \cdots, c_n$,其中 $c_1 < c_2 < \cdots < c_n$。对于不同的截止时间 d_i 和执行时间 c_j,任务的优先级表示为 $p(\tau, d_i, c_j)$,简记为 p_{ij}。遍历所有的 d_i 和 c_j,可以得到确定的优先级表。当任务的相对截止时间和执行时间是典型值时,可以查表得到对应的优先级;当有一个属性值不是典型值时,可以参考文献[5]中的方法插值得到其优先级。

2.2　优先级表

图 1 是综合考虑相对截止时间和执行时间的 EDF 算法优先级表设计。在图 1 中,数字越小,任务对应的优先级越高。相同截止时间内,任务的优先级随着执行时间的增大而降低;对于截止时间不同的任务,截止时间越小优先级越高。图 2 是考虑了任务截止时间和到达时间的优先级表设计。可以看出,相同截止时间内,任务的到达时间越晚,其优先级越高;而不同截止时间内,截止时间越小,优先级越高。这里给出的是 EDF 算法的两种不同的变形。同理,也可以在综合考虑 deadline 和 exec 时,使相同 deadline 的任务优先级随着 exec 的增大而升高,也可以使相同 deadline 的任务优先级随着 arri 的增大而降低。因此,EDF 算法有四种不同实现,在仿真实验部分,实现了这四种策略,分别对应 $f(\overleftarrow{\text{exec}})$,$f(\overrightarrow{\text{arri}})$,$f(\overrightarrow{\text{exec}})$ 和 $f(\overleftarrow{\text{arri}})$,并得到对调度成功率为 $f(\overleftarrow{\text{exec}}) > f(\overrightarrow{\text{arri}}) > f(\overleftarrow{\text{arri}}) > f(\overrightarrow{\text{exec}})$。

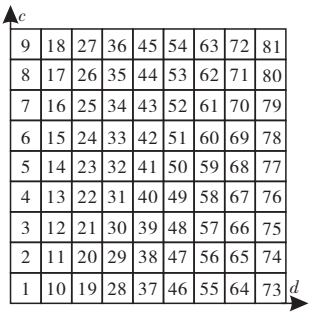

图 1　优先级表

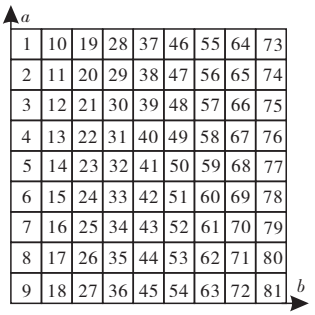

图 2　优先级表

对于 EDF 算法,也可以同时考虑任务的三种属性,即在优先考虑属性 deadline 的条件下,优先考虑属性 exec,再考虑属性 arri;或者优先考虑属性 arri,再考虑属性 exec。因此,EDF 算法共有八种变形。同理,在 LLF 算法下,当任务具有相同的空闲时间时,可供考虑的属性有 deadline、exec 和 arri,因此 LLF 算法有八种变形。仿真实验时,舍弃了一些调度成功率明显较差的策略,优先级表的设计与 EDF 算法类似。

3　优先级调度算法

每种动态优先级调度算法 PTBM(priority table based method)都对应一种优先级表设计,即当给定调度算法时,可设计出对应的优先级表。基于逆向思维,当设计出一种优先级表时(不管是否有效),也就形成了对应的算法。如图 3 所示的优先级表设计,理论上也对应一种调度算法,但是无法从算法的角度理解它的意义,因而采用了从设计优先级表推断算法的方式。

3.1　PTD 算法

PTD(priority table design)算法就是基于上述思想提出的算法。该算法综合考虑相对截止时间和空闲时间两个任务属性,其设计原则是截止时间越小和空闲时间越短的任务优先级越高。任务的优先级等级与相对截止时间、空闲时间之间的关系用一种线性关系来描述:

$$p = d_i + k \times s_j \tag{3}$$

优先级等级的大小随着 k 的改变而改变,而且还考虑了属性的变化规律。图 4 为 PTD 优先级表的一种实现,其中 k = -1,箭头所指的方向表示优先级降低。本文基于任务属性对调度成功率影响度分析,提出了一种新的优先级表设计方法 PTBM。

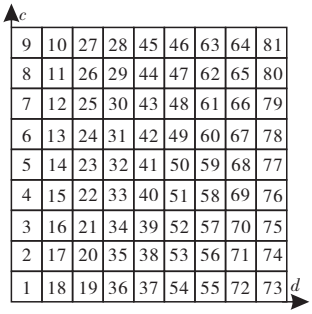

图 3 优先级表

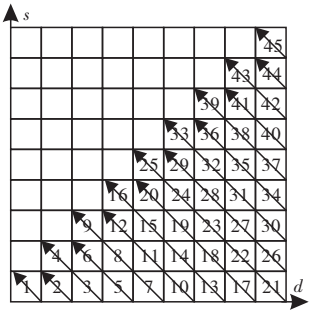

图 4 PTD 策略优先级表

3.2 PTBM 算法

在调度成功率对任务属性的敏感度分析中,任务的截止时间具有最高的敏感度,其次是执行时间和到达时间,因此,截止时间应该作为优先级表设计考虑的第一因素。一个实时任务除了以上三种静态属性外,还有动态属性,如空闲时间。PTBM 算法正是额外考虑了空闲时间这个动态属性,因而任务的表示可以扩展为 $\tau(\text{arri},\text{deadline},\text{exec},\text{laxity})$。优先级表设计考虑的另一种属性由 exec、arri 和 laxity 的敏感度大小决定,选择敏感度大的属性。已知 $s(\text{exec})>s(\text{arri})$,又已知 laxity = (deadline $-t$) $-$ (exec $-et$),其中 t 是当前时刻,et 是任务已经执行的时间。当任务的截止时间相同时,deadline $-t$ 也相同;所以 $s(\text{laxity})$ 的大小由 $s(\text{exec}-et)$ 决定。又因为 et 是不可预测的,可以认为是随机变量,所以 exec $-et$ 的不确定性大于 exec,因此 $s(\text{laxity})>s(\text{exec})$。基于上述敏感度分析,采用 deadline 和 laxity 作为优先级表设计的考虑属性。

在满足式(3)的条件下,设计如图 5 所示的优先级表 PTBM。当任务的相对截止时间相同时,任务的优先级随着空闲时间的增大而升高;当任务的截止时间不同时,截止时间小的任务具有更高的优先级。可以发现,这是一种新的 EDF 算法策略。此外,优先级表也可以设计为:当任务的相对截止时间相同时,任务的优先级随着空闲时间的增大而降低,此时 k 取无穷大。实验证明,图 5 所示的设计具有较高的调度成功率。

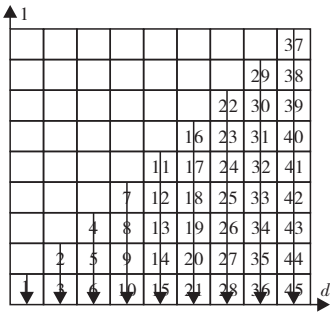

图 5 PTBM 策略优先级表

4 仿真实验

4.1 敏感度验证

记 $U(a,b)$ 表示定义域为$[a,b]$上的均匀分布函数。取 arri = $U(0,3)$,deadline = $U(5,10)$,exec = $[1+\text{rand}\times(\text{deadline}-\text{arri}-1)]$,rand 为$[0,1]$上的随机数。共仿真 10 批,每批 100 组,每组 10 个任务。EDF 算法有六种策略:R_1 先到先调度;R_2 后到先调度;R_3 短任务先调度;R_4 长任务先调度;R_5 短空闲时间先调度和 R_6 长空闲时间先调度。如图 6 所示,$|R_6-R_5|>|R_4-R_3|>|R_2-R_1|$,即 $s(\text{laxity})>s(\text{exec})>s(\text{arri})$,结果如表 1 所示。可以看出,算法调度成功率对于空闲时间的敏感度远大于执行时间和到达时间;且对执行时间的敏感度略高于到达时间。

表 1 属性敏感度

算法	敏感度
$\lvert R_6-R_5\rvert$	0.032 5
$\lvert R_4-R_3\rvert$	0.017 8
$\lvert R_2-R_1\rvert$	0.015 0

与 EDF 算法类似,对 LLF 算法也六种不同的策略。仿真验证发现,调度成功率对任务截止时间的敏感度远高于执行时间和到达时间,即 $s(\text{deadline})>s(\text{exec})>s(\text{arri})$。综上所述,调度成功率对任务属性的敏感度对比为

$$s(\text{deadline})>s(\text{laxity})>s(\text{exec})>s(\text{arriv})$$

4.2 影响度验证

采用 4.1 节中的实验设计,分别对六种 EDF 算法和六种 LLF 算法进行实验,对比 $f(\overleftarrow{\text{deadline}})$、$f(\overrightarrow{\text{deadline}})$、$f(\overleftarrow{\text{exec}})$、$f(\overrightarrow{\text{exec}})$、$f(\overleftarrow{\text{arri}})$、$f(\overrightarrow{\text{arri}})$、$f(\overleftarrow{\text{laxity}})$ 和 $f(\overrightarrow{\text{laxity}})$ 的大小。

实验所得结果与理论分析一致,即 $f(\overleftarrow{\text{deadline}})>f(\overrightarrow{\text{laxity}})>f(\overleftarrow{\text{laxity}})>f(\overleftarrow{\text{exec}})>f(\overrightarrow{\text{arri}})>f(\overleftarrow{\text{arri}})>f(\overrightarrow{\text{exec}})>f(\overrightarrow{\text{deadline}})$。在 EDF 算法中,$f(\overleftarrow{\text{exec}})$ 略微大于 $f(\overrightarrow{\text{arriv}})$,而 $f(\overrightarrow{\text{exec}})$ 略微小于 $f(\overleftarrow{\text{arriv}})$,说明 exec 与 arri 对调度成功率的影响几乎相等。在 LLF 算法中,deadline、exec 和 arri 对调度成功率的影响有比较明显的差别。

4.3 PTBM、EDF、LLF、PTD 的调度成功率比较

不失一般性,在比较算法 PTBM、EDF、LLF 和 PTD 的仿真实验中,仍然采用 4.1 节中任务的生成方法,用调度成功率衡量算法的优劣性。为了更全面地比较算法 PTBM 与 PTD,设计了多种优先级表策略:R_7 斜率 $k=-1$ 左上递增,即算法 PTD;R_8 斜率 $k=-1$ 交叉递增 1;R_9 斜率 $k=-1$ 交叉递增 2;R_{10} 斜率 $k=-1$ 右下递增;R_{17} 斜率极大向上递增;R_{18} 斜率极大交叉递增 1;R_{19} 斜率极大交叉递增 2;R_{20} 斜率极大向下递增,即 PTBM 算法。仿真实验不仅对比了 R_{13} 与 R_{20} 的调度成功率,而且对比了优先级表的不同设计。

如图 7 所示,R_{16} 的调度成功率最高;R_{13},即算法 PTD 最低;R_{14} 和 R_{15} 居中。因此,当斜率 $k=-1$ 时,优先级数右下递增比左上递增有更高调度成功率,R_{14} 和 R_{15} 正是因为交叉递增(既包含左上递增,又包含右下递增),其调度成功率介于 R_{13} 与 R_{16} 之间。

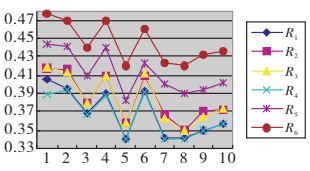

图6 EDF不同策略调度成功率

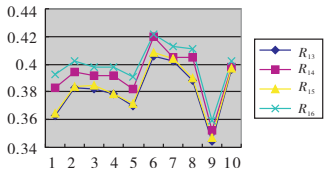

图7 PTD调度成功率

如图 8 所示,R_{20},即算法 PTBM 的调度成功率最高;R_{17} 最低;R_{18}、R_{19} 介于 R_{17} 与 R_{20} 之间。R_{20} 的优先级表设计如图 5 所示。因而推断,在斜率极大时,优先级数向下递增的调度成功率高于向上递增;R_{18} 和 R_{19} 由于既存在向上递增又存在向下递增(即交叉递增),所以其调度成功率介于 R_{17} 与 R_{20} 之间。其实,$R_i(i=17,18,19,20)$ 是 EDF 算法的不同策略。

R_3 采用截止期相同时短任务优先的调度策略,是 EDF 算法中调度成功率最高的策略;R_7 是空闲时间相同时截止期最小优先调度的策略,是 LLF 算法中调度成功率最高的策略;R_{13} 是 PTD 算法;R_{16} 是 $k=-1$ 时调度成功率最高的优先级表设计;R_{17} 是 k 极大时调度成功率最低的优先级表设计;R_{20} 是 k 极大时调度成功率最高的优先级表设计,即 PTBM。仿真实验对比这六种算法,证明 PTBM 的较高调度成功率。如图 9 所示,调度成功率的大小为 $R_{20}>R_{17}>R_{16}>R_{13}>R_7>R_3$。可以看

出,考虑属性截止时间和空闲时间的 EDF 算法,在最坏情况下(R_{17})调度成功率也高于传统 EDF、LLF 和 PTD 算法;而采用最大空闲时间优先的 EDF 算法(R_{20})具有最高的调度成功率。原因是当任务的截止时间相同时,空闲时间越大,表示任务还需执行的时间越短,优先执行长空闲时间的任务可以在保证有一个任务执行完的基础上节省更多的时间。

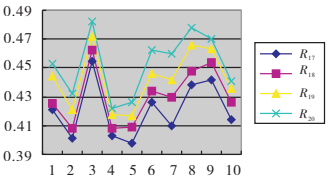

图8　PTBM调度成功率

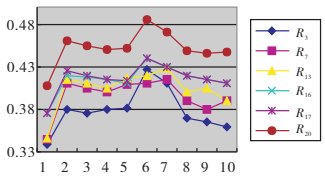

图9　EDF、LLF、PTD、PTBM调度成功率

5　结束语

本文首先讨论了算法调度成功率对于实时任务属性的敏感度,提出了不同算法对于任务属性有不同的相对敏感度。影响算法调度成功率的因素不仅包含任务属性,还包含属性的变化规律。基于敏感度的理论,分析了任务属性和变化规律对算法调度成功率的影响度,得出任务的截止时间和空闲时间是最有影响的两个因素。综合考虑这两个属性,使得任务的截止时间越小,任务的优先级越高;相同截止时间下,空闲时间越大的任务优先级越高。这是因为空闲时间越大,表示任务还需执行的时间越短,优先执行长空闲时间的任务可以在保证有一个任务执行完的基础上节省更多的时间。最后设计了 20 种不同的算法或者策略来验证仿真验证敏感度和影响度的理论,实验结果证明 PTBM 算法具有较高的调度成功率。

参考文献:

[1] ZHOU Shi, MONDRAGON R J. Accurately modeling the Internet topology[J]. Physical Review E,2004,70(6):96-108.

[2] LIU C L, LAYLAND J W. Scheduling algorithms for multiprogramming in a hard-real-time environment[J]. Journal of the ACM, 1973,20(1):46-61.

[3] LIU Yun-sheng, HE Xin-gui, TANG Chang-jie, *et al*. Special type database technology[M]. Beijing: Science Press, 2000.

[4] JENSEN E D, LOCKE C D, TODUCA H. A time-driven scheduling model for real-time operating systems[C]//Proc of IEEE Real-time Systems Symposium. Washington DC: IEEE Computer Society Press, 1985:112-122.

[5] GOOSSENS J, RICHARD P. Overview of real-time scheduling problems[C]//Proc of the 9th International Workshop on Project Management and Scheduling. 2004:13-22.

[6] 金宏,王宏安,王强,等. 一种任务优先级的综合设计方法[J]. 软件学报, 2003,14(3):376-382.

[7] 金宏,王宏安,王强,等. 改进的最小空闲时间优先调度算法[J]. 软件学报,2004,15(8):1116-1123.

[8] 李琦,巴巍. 两种改进的 EDF 软实时动态调度算法[J]. 计算机学报, 2011,34(2):943-950.

[9] 檀明,魏臻,韩江洪. 非抢占式 EDF 算法下周期性任务的最小相对截止期计算[J]. 计算机应用研究, 2012,29(2):722-724.

[10] 袁晷,檀明,周晶晶. EDF 调度算法可调度性分析方法的改进研究[J]. 计算机应用研究,2013,30(8):2429-2431.

[11] 王多强,鲁剑锋,李庆华. 实时调度中基于多特征参数的任务优先级设计方法[J]. 计算机工程与科学, 2008,30(1):73-78.

[12] BIYABANI S R, STANKOVIC J A, RAMAMRITHAM K. The Integration of deadline and criticalness in hard real-time scheduling[C]//Proc of the 9th IEEE Real-time Systems Symposium. Washington, DC: IEEE Computer Society Press, 1988:152-160.

[13] 穆阿里,吴仲光,张昭瑜,等. 一种多特征综合的实时[J]. 四川大学学报:自然科学版,2005,42(3):621-623.

(上接第 1409 页)

表 1　测试图的基本信息

Graph	Vertex	edges	Graph	Vertex	edges
Graph4096	4 096	24 576	New York	264 346	733 846
Graph65536	65 536	393 216	Florida	1 070 376	2 712 798

表 2　SSSP 算法实验结果比较

Graph	Dijkstra_ heap/ms	SSSP GPU/ms	Advanced_ Atomics_ SSSP/ms	Graph	Dijkstra_ heap/ms	SSSP GPU/ms	Advanced_ Atomics_ SSSP/ms
Graph4096	6.28	0.98	1.32	New York	157	102	118
Graph65536	21.53	4.15	3.76	Florida	1 086	1 651	1 593

表 3　APSP 算法实验结果比较

Graph	CPU_ APSP/s	APSP USING SSSP/s	Heap_ APSP/s	Graph	CPU_ APSP/s	APSP USING SSSP/s	Heap_ APSP/s
Graph4096	545	12.44	9.6	New York	23 518	1 569	511
Graph65536	13 104	743	252	Florida	65 950	2 652	1 403

5　结束语

本文针对图论中最短路径问题提出了 Advanced_Atomics_SSSP 和 Heap_APSP 算法。通过实验验证,Advanced_Atomics_SSSP 算法对每个节点的度大于 6 时,其加速效果更加明显;Heap_APSP 可以取得 40 ~ 50 倍的加速效果。根据 Heap_APSP 的特点,也可以实现有限个节点到其他部分节点的最短路径搜索。因此,笔者后续工作将把该算法应用到 FPGA 的布线算法中,期望能够缩短 FPGA 的布线时间。

参考文献:

[1] HARISH P, NARAYANAN P J. Accelerating large graph algorithms on the GPU using CUDA[C]//Proc of the 14th International Conference on High Performance Computing,2007:197-208.

[2] 张凌洁,赵英. 基于 GPU 的并行 APSP 问题的研究[J]. 电子设计工程,2012,20(17):15-18.

[3] OKUYAMA T, INO F, HAGIHARA K. A task parallel algorithm for computing the costs of all-pairs shortest paths on the CUDA-compatible GPU[C]//Proc of International Symposium on Parallel and Distributed Processing with Applications. 2008: 284-291.

[4] MATSUMOTO K, NAKASATO N, SEDUKHIN S G. Blocked all-pairs shortest paths algorithm for hybrid CPU-GPU system[C]//Proc of IEEE Interenational Conference on High Performance Computing and Communications. 2011:145-152.

[5] OKUYAMA T, INO F, HAGIHARA K. A task parallel algorithm for finding all-pairs shortest paths using the GPU[J]. International Journal of High Performance Computing and Networking,2012, 7(2):87-98.

[6] https://dev. ece. ubc. ca/projects/gpgpu-sim/browser/ispass2009-benchmarks/BFS/datalast visited[EB/OL]. (2013-04-24).

[7] JOSEPH T, Jr KIDER. GPU as a parallel machine: sorting on the GPU, CIS 700/010[EB/OL]. http://www. cis. upenn. edu/ ~ suvenkat/700/lectures/19/sorting-kider. pdf.

[8] 9th DIMACS implementation challenge-Shortest paths[EB/OL]. (2013-04-27). http://www. dis. uniroma1. it/challenge9/download. shtml, last visited.

[9] TRAN Q N. Designing efficient many-core parallel algorithms for all pair shortest-paths using CUDA[C]//Proc of the 7th International Conference on Information Technology: New Generations (ITNG). 2010:7-12.