可靠性代价驱动的实时任务调度算法

张君雁　杨国纬　罗旭斌
（电子科技大学计算机科学学院　成都 610054）

Reliability-Driven Tasks Scheduling for Real-Time System

ZHANG Jun-Yan　YANG Guo-Wei　LUO Xu-Bin
(College of Computer Science, UESTC, Chengdu 610054)

Abstract　In this paper, we describe a two-phase scheme to determine a scheduling of tasks with precedence constraints that employ a reliability measure as one of the objectives in a Real-Time and heterogeneous distributed system. The simulation results show that, for task graphs with precedence constraints in a heterogeneous Real-Time system, the method performs significantly better than the two algorithms presented that do not consider reliability cost. Furthermore, the results suggest that higher computational heterogeneity is conducive to improving the schedulability of the reliability cost-driven (RCD)algorithm, while the opposite is true for the two non-RCD algorithms.

Keywords　Reliability cost, Real-time system, Real-time scheduling, Heterogeneous distributed systems, Performance

1　概述

分布式系统越来越广泛地用于重要的实时系统应用程序中，关键问题在于必须保证每个任务在其截止时间之前完成。在许多实时调度算法中调度性是需要最大化的功能目标之一[1,2,4,5,7]。为了使实时调度算法更实用，必须考虑任务优先权限制[9,10]。文[11]中提出将离线分析和在线保证结合使用的方案。文[12]提出了一个分布式实时系统中的最佳任务调度算法。上述算法都是为同构分布式系统设计的，均假定系统中的处理器都是一样的，所以不能直接应用于异构分布式系统。分布式异构计算系统包括多个异构模块，彼此交互作用解决问题[3,8,16]。文[13]中阐述了一种将异构任务图映射到异构系统图的一般方法。文[14]讨论了异构环境中的调度专家顾问。文[15]研究一种用于异构系统的实时调度算法，但都是非实时的。

本文在异构系统中加入任务优先权限制，目标是研究在异构分布式系统中具有优先权限制的实时任务调度算法。其中，任务由有向非循环图(DAG)表示，而异构分布式系统是指多个处理器以不同的速度运作，处理器之间以不同的带宽链接。此外，使用一个可靠性量度——可靠性代价(reliability cost)作为实时任务调度的目标函数，以达到驱动调度进程的目的。

2　系统模型

作为分布式系统输入的实时作业由用 DAG 描述的一组交互作用的实时任务组成。本文研究一种二阶段法来调度实时任务。

定义 1　一个实时 DAG 定义为 RG＝{T,E}。其中：

(1)T＝{$t_1,t_2,\cdots,t_n$}：表示实时任务集合；

每个任务 t_i 是一个 6 元组模型：t_i＝(C,d,s,f,g,ρ)，其中：

C：代表执行时间向量，C_i＝[c(i,1),…,c(i,m)]。c(i,j)表示任务 t_i 在处理器 j 上的执行时间；

d、s 和 f：分别表示截止时间和开始时间以及完成时间；

g：代表入度(in-degree)；

ρ：表示任务 t_i 被分派到的处理器。

(2)E ＝{$e_1,e_2,\cdots,e_k$}：表示任务中加权有向边的集合。

E 中的每个元素 e_i(i＝1,2,…,k)代表从一个任务传递到另一个任务的消息，且 e_i＝(t_s,t_r,s,f,c)。这里：

t_s 和 t_r：分别是用于发送和接收 e_i 的两个任务；

s 和 f：分别表示开始时间和结束时间；

c：表示传输的数据量。

本文将异构分布式系统模型化为一个处理器集合：Ω＝{$P_1,P_2,\cdots,P_m$}，P_i＝(Δ_i,M_i,λ_i)。λ 表示失败率，M_i 是消息列表集合：M_i＝{$M_{i1},\cdots,M_{ij},\cdots,M_{im}$}，j≠i，其中的消息按照开始时间的增序排列。在处理器 p_i 和 p_j 之间的边上的权值 w_{ij} 表示处理器间发送/接收一个单位长度的消息的延迟。

定义 2　处理器 P_j 上执行的任务 t_i 的可靠性代价是 P_j 的失败率和任务 t_i 在 P_j 上执行时间的乘积。异构系统 Ω 中，一个给定任务调度的可靠代价 RC(Ω)等于基于该调度的所有任务的可靠性代价之和。可靠性代价由 $e^{-RC(\Omega)}$给出。

$$RC(\Omega)=\sum_{j=1}^{m}\sum_{t_i\in\Delta_j}\lambda_j c(i,j)$$

3. 实时调度

实时调度方法分两步：第一步，决定调度顺序；第二步，将任务调度到分布式系统中。本文设计一种允许灵活增加和删除的排序算法和调度算法，形式化为：MSSP＝{O,OL,S}，O 表示排序算法的集合，S 表示调度算法的集合，OL 是任务的有序列表。

3.1　排序算法

最早截止时间优先排序算法 SORT_EDF，作用是使得具有最早截止时间的任务具有最高优先级。

SORT_EDF ALGORITHM (Input: RG, Output: OL)

张君雁　博士生，研究方向为计算机系统结构。杨国纬　教授，博导，研究方向为计算机系统结构、网络计算、自然语言处理。罗旭斌　硕士生，研究方向为网络计算、计算机安全。

```
    OL←Φ;
    ZDL←Create_A_LIST();
    FOR each t_i.g∈T DO /* Initialize the list ZDL */
        IF t_i.g = 0 THEN ENLIST (ZDL, t_i);
    WHILE list ZDL is not empty DO
Select t from ZDL, where ∀ t_i∈ZDL(t.d≤t_i.d);
        OL←Append_OL(t); /* Append task t to OL */
        v←First_Son(RG, t); /* Get first son of t */
        WHILE v≠NULL DO
            v.g←v.g−1; /* In-degree of v decrease 1 */
            IF v.g = 0 THEN Select t from ZDL,
                where ∀ t_i∈ZDL(t.d≤t_i.d);
            v←Next_Son(RG, t, v);
        END WHILE
    END WHILE
END
```

3.2 调度算法

3.2.1 *尽早调度算法(AEAP)* 排序后要决定每个实时任务的开始时间。为此要计算一个任务在每个处理器上的最早开始时间 EST。尽早调度算法 AEAP 基于三个过程：get_EAT 过程：计算 P_j 上的任务 t_i 的最早可用时间；get_MST 过程：决定在集合 $MS(t_i)$ 中的每个消息的开始时间；get_EST 过程：计算在处理器 P_j 上的任务 t_i 的最早开始时间。

```
AEAP ALGORITHM (Input: OL,Ω; Output: T)
    FOR (each task t_i in OL)DO
        est←∞;/* Initialize the earliest start time */
        FOR each processor P_j in Ω
        DO
            IF (get_EST(t_i,P_j)≤est)
            THEN( est←get_EST(t_i,P_j));
        END FOR
        IF (est+ t_i.c(i,j)> t_i.d)THEN RETURN(FAIL);
        t_i.s←est; t_i.f←est + t_i.c(i,j); t_i.ρ←P_j;
        Insert task t_i into set P_j.Δ;
        Update information of each message;
    END FOR
    RETURN (SUCCESS);
END
```

3.2.2 *尽晚调度算法(ALAP)* 尽晚调度算法 ALAP 保证实时任务是尽晚开始的。get_LST 过程专门用于计算任务 t_i 在处理器 P_j 上的最晚开始时间。

```
ALAP ALGORITHM (Input: OL,Ω; Output: T)
    FOR (each task t_i in OL)DO
        lst←0; /* Initialize the latest start time */
        schedule←NO; processor_id←0;
        FOR each processor P_j in Ω DO
            (st, find)←get_LST(t_i, P_j,);
            IF (find = YES)THEN
                schedule←YES;
                IF (st > lst)THEN (lst←st);
                    processor_id←j;
                END IF;
            END IF
        END FOR
        IF (schedule = NO)THEN RETURN (FAIL);
        t_i.s←lst; t_i.f←lst +t_i.c(i,j);t_i.ρ←P_j;
        Insert task t_i into set P_j.Δ;
        Update information of each message;
    END FOR
    RETURN (SUCCESS);
END
```

3.2.3 *可靠性代价驱动调度算法(RCD)* AEAP 和 ALAP 是两个不考虑可靠性的算法。本节给出可靠性代价驱动调度算法(Reliability Cost Driven scheduling，RCD)。RCD 算法中每个实时任务被分派到具有最小可靠性代价的处理器上，并尽可能早地被调度。

```
RCD ALGORITHM (Input: OL,Ω; Output: T)
    RC←0;
    FOR each task t_i on OL DO
        st←∞; find←NO; rc←∞;
        FOR each processor P_j in Ω DO
        est←get_EST(T_i,P_j);rc_j←λ_j·c(i, j);
            IF (est + c(i, j)≤t_i.d)THEN
                Find←YES;
                IF ((rc_j<rc)OR (rc_j=rc and est < st))THEN
                    st←est; p←p_j;rc←rc_j;
                ENDIF
            ENDIF
        ENDFOR
        IF find=NO THEN RETURN (FAIL);
        t_i.s←est; t_i.f←est + t_i.c(i,j);ti.ρ←P_j;
        Insert task t_i into set P_j.Δ;
        Update information of each message;
        ENDFOR
        RETURN (SUCCESS);
END
```

4. 性能评测

4.1 工作量

本节给出用于性能评测的仿真试验结果。该仿真程序为实时任务随机地产生两种 DAG，分别是二叉树和网状结构。

定义3 工作量参数

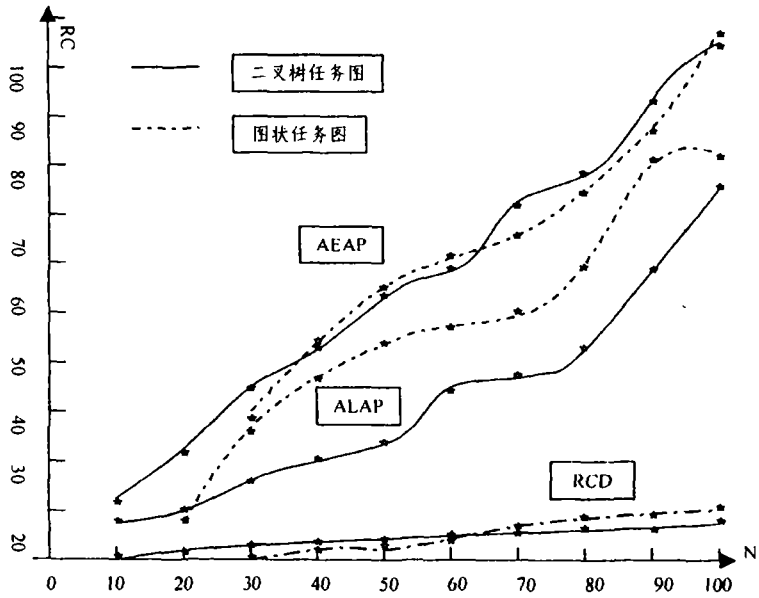

图1 不同算法的可靠性代价比较

(1)任务的执行时间(ET)：属于特定范围的值，对每个执行时间向量 C 中的元素，ET 是随机选择的。

(2)系统规模(SZ)：系统中的处理器数目，用 N 表示。

(3)通信权值(CW):属于特定范围的值,对于每个 w_{ij},CW 是随机选择的。

(4)通信量(CV):对于每个消息 e.c,e∈E,CV 是随机选择的。

(5)处理器故障率:每个处理器产生故障的概率在0.95×10^{-6}/hour 到1.05×10^{-6}/hour 之间[8]。

(6)任务截止时间:给出任务 t_i,t_i∈T,假定 t_i 在 P_k 上,t_j 在 P_l 上,那么 t_i 的截止时间可以被模型化为如下形式:

$t_i.d=MAX(t_j.d)+1+e.c\times w_{lk}+MAX[C(i,k)]+\delta$

其中 e=(t_j,t_i)∈E,k∈[1,n],δ 是根据均匀分布被计算出来的[8]。

4.2 可靠性代价

本节将提供仿真运行 AEAP、ALAP 和 RCD 的可靠性代价结果。工作量参数:N=∞;CW=[0.5,1.5];CV=[1,10];ET=[1,200];δ=[1,50]。从图1可以看出:依据数量级的规则,可靠性驱动的 RCD 算法在性能上远优于在调度目标中不考虑可靠性代价的 ALAP 算法和 AEAP 算法。

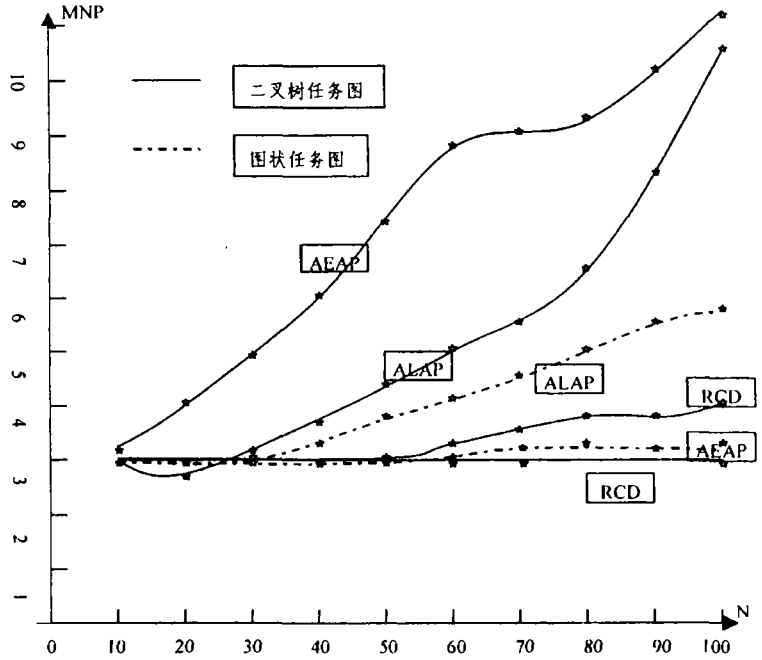

图2 不同算法的 MNP 比较

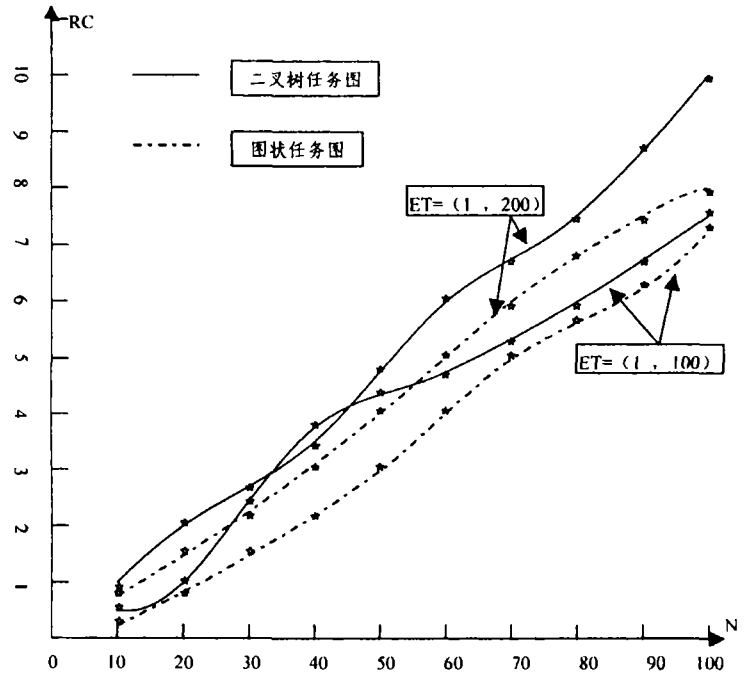

图3 执行时间和系统可靠性代价的比较

4.3 处理器的最小数目

查找最小处理器数目的算法(FMNP)是用来查找能使所有任务都在特定的截止时间之前被调度的最小处理器数目(MNP)。

```
FMNP (Input: RG,Ω, Sort_Alg, Sch_Alg; Output: k,Ω)
    OL←Sort_Alg(RG);/* Sort_Alg 代表用于产生 OL 的排序
                       算法 */
Lower=1;Upper=n;k=⌊(Lower + Upper)/ 2⌋;
    WHILE (Lower = k)DO
        success←Sch_Alg(OL,Ω);/* Sch_Alg 表示被调用来调度
```

```
                                    任务集的算法 * /
        IF (success = SUCCESS)THEN Upper = k;
        ELSE Lower = k;
        k=⌊(Lower + Upper)/ 2⌋;
    END WHILE
    RETURN (k + 1;Ω);
END
```

找寻 MNP 仿真实验工作量参数是:失败率是1×10^{-6}/hour;CW=[0.5,1.5];CV=[1,10];ET=[1,200];δ=[1,50]。图2显示:随着任务数目的增加,大多数算法的最小处理器数目也随之增加,而采用图状 DAG 的 RCD 则始终保持3这个值。

4.4 RCD 算法中执行时间和可靠性代价之间的关系

下述试验分析 RCD 的执行时间和可靠性代价之间的关系。ET 有两个范围:[1,100]和[1,200]。从图3可以得出结论:当每个向量 C_i 中的执行时间增加时,系统的可靠性代价也随之增加。

结论 在实时任务调度研究领域中,现有的大部分工作要么就是没有考虑容错性和可靠性问题,要么就是仅考虑系统的异构性,或者假定任务之间是独立的。本文设计了一种可靠性代价驱动的算法 RCD,它能在异构实时系统中进行具有优先权限制的实时任务调度。该算法将可靠性代价用作实时任务调度的一个目标函数,并由可靠性驱动整个实时任务调度过程。同时列举了没有考虑可靠性代价的 AEAP 和 ALAP 算法作为对比。仿真试验的结果表明,依据可靠性代价和作为调度性量度的最小处理器数目,RCD 远远优于 AEAP 和 ALAP 算法。而且该结果还表明较高的异构程度有利于提高 RCD 算法的调度性,而对于 AEAP 和 ALAP 算法则相反。该仿真试验是基于两种类型的实时任务图(DAG),即二叉树任务图和网状任务图,它们是很有代表性的实时任务图。工作量参数要么是基于文献,要么是选择那些能合理代表实际工作量的数值,并能为算法提供一些强化测试。

参 考 文 献

1 Berman P, DasGupta B. Improvements in Throughput Maximization for Real-Time Scheduling, DIMACS: [Technical Report, TR-99-52]. 1999

2 Chang G L, Joosun H, Yang M S, et al. Analysis of Cache-Related Preemption Delay in Fixed-Priority Preemptive Scheduling. IEEE Trans. on Computers, 1998, 47(6):700～713

3 Chen Z, et al. An Architecture to Support Collaborative Distributed Heterogeneous Computing Applications: [Technical Report, TR97-26]. Department of Computer Science, University of Bristol, UK, 1997

4 Liu H, Zarki M E. Adaptive Source Rate Control for Real-Time Wireless Video Transmission. Mobile Networks and Application, 1998, 3:49～60

5 Manimaran G, Murthy C S R. An Efficient Dynamic Scheduling Algorithms for Multiprocessor Real-Time Systems. IEEE Trans. On Parallel and Distributed Systems, 1998, 9(3):312～319

6 Shatz S M, Wang J P, Goto M. Task Allocation for Maximizing Reliability of Distributed Computer Systems. IEEE Trans. Computers, 1992, 41(9):1156～1168

7 Son S H, Chaney C. Supporting the Requirements for Multilevel Secure and Real-time Databases in Distributed Environments. Database Security: Status and Prospects, Chapman and Hall Publishing, 1998. 73～91

8 Srinivasan S, Jha N K. Safty and Reliability Driven Task Allocation in Distributed Systems. IEEE Trans. Parallel and Distributed Systems, 1999, 10(3):238～251

9 Chetto H, Silly M, Bouchentouf T. Dynamic Scheduling of Real-Time Tasks under Precedence Constraints. J. Real-Time Systems, 1990, 2(3):181～194

10 Ramamritham K. Allocation and Scheduling of Complex Periodic Tasks. In: Proc. Int'l Conf. Distributed Computing Systems, 1990. 108～115

11 Natale M D, Stankovic J A. Dynamic End-to-End Guarantees in Distributed Real-Time Systems. In: Proc. Real-Time Systems Symp., 1994. 216～227

12 Abdelzaher T F, Shin K G. Combined Task and Message Scheduling in Distributed Real-Time Systems. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(11)

13 Eshaghian M M, Wu Y C. Mapping heterogeneous task graphs onto heterogeneous system graphs. In: IEEE Proc. 6th Heterogeneous Computing Workshop (HCW '97), Geneva, SWITZERLAND April 1, 1997. [14] Sirbu, M. G.; Marinescu, D. C, A scheduling expert advisor for heterogeneous environments. In: IEEE Proc. 6th Heterogeneous Computing Workshop (HCW '97), Geneva, SWITZERLAND April 1, 1997

14 Qin Xiao, Han Z F, Jin H, Pang L P, Li SL. Real-time Fault-tolerant Scheduling in Heterogeneous Distributed Systems. In: Proc. of the 2000 Intl. Workshop on Cluster Computing- Technologies, Environments, and Applications (CCTEA' 2000), Las Vegas, Nevada, USA, June, 2000

15 Maheswaran M, Siegel H J. A Dynamic Matching and Scheduling Algorithm for Heterogeneous Computing Systems. In: IEEE Proceedings of the Seventh Heterogeneous Computing Workshop, Orlando, Florida, 1998

(上接第41页)

据检索、并行计算等实例。结果表明,Cogent 系统提供了一种新的基于构件的应用软件系统开发方法。

参 考 文 献

1 Tanenbaum A S. Computer Networks. Prentice Hall, 1989

2 Stevens R. Unix Network Programming, Second Edition, Volume 1. Networking APIs: Sockets and XTI. Prentice Hall, 1998

3 Armstrong J, et al. Concurrent Programming in Erlang. Prentice Hall, 1999

4 Burghart T. Distributed Computing Overview. QUOIN, Cambridge, Massachusetts, 1998

5 Object Management Group. The Common Object Request Broker: Architecture and Specification, revision 2.2, 1998

6 Microsoft. DCOM Technical Overview. white paper, November 1997. http://www.microsoft.com/ntserver/library/dcomtec.exe

7 Microsoft. DCOM Architecture. white paper, July, 1997. http://www.microsoft.com/com/wpaper/default.asp

8 Sunsoft. Java RMI Tutorial, Revision 1.3. JDK 1.1 FCS, February 10, 1997

9 Lu Jian. Some Research on Componentware Frameworks based on Mobile Agent Technology: [technical report of State Key Laboratory for Novel Software Technology]. Nanjing Univ., Nanjing, P. R. China, 1999