第 36 卷 第 2 期 计 算 机 工 程 2010 年 1 月
Vol.36 No.2 Computer Engineering January 2010
·博士论文· 文章编号：1000—3428(2010)02—0015—03 文献标识码：A 中图分类号：TP311.2

一种新的分布式控制系统容错调度算法

孟庆春，刘云卿
(中国科学院力学研究所，北京 100190)

摘 要：目前多数容错调度算法在调度非周期任务时采用预留时间的方法，非周期任务无法得到充分响应。针对该问题，提出一种新的分布式控制系统容错调度算法，采用任务集划分的方法在不同处理机上运行不同的周期任务子集，使每个处理机具有不同的非周期任务预留时间，当非周期任务发生时，即可得到有效响应。结果表明，该方法能提高容错调度的效率。

关键词：分布式控制系统；容错；混合任务调度

New Fault-tolerant Schedule Algorithm for Distributed Control System

MENG Qing-chun, LIU Yun-qing
(Institute of Mechanical, Chinese Academy of Sciences, Beijing 100190)

【Abstract】 Existing algorithms adopt the method based on the reservation of the time when scheduling non-periodic tasks. This method can not fully response the non-periodic tasks. In order to solve the problem mentioned above, this paper proposes a new fault-tolerant schedule algorithm for distributed control system. It uses the method of partitioning the task set to execute different periodic task subset in different processors and to guarantee every processor has different reserved time. When the non-periodic task starts, the method can schedule it effectively. Result manifests that new method can improves the efficiency of the fault-tolerant schedule.

【Key words】 distributed control system; fault-tolerant; hybrid task schedule

1 概述

随着控制系统复杂性的不断提高，分布式控制系统越来越多地应用于各类控制领域，如工业控制系统、武器防御控制系统、飞行控制系统、铁路控制系统等，但是随着控制器数目的增加，系统出现故障的可能性也相应增大。在控制系统中，每个任务都有严格的时间约束(时限)，如果控制器发生故障使某些任务不能在其时限内完成，就有可能造成很大的损失。为了解决上述问题，需要为分布式控制系统提供一定的容错能力，以保证在某些控制器出现故障的情况下仍能有效完成任务，从而提高整个控制系统的可靠性。

容错调度算法是一种通过软件解决分布式控制系统容错问题的有效方法。该方法的优点是不需要额外的硬件代价。分布式系统目前采用的容错技术主要有：分布式投票技术[1]，反转恢复技术[2]和基/副版本技术[3]，然而这些技术没有考虑任务的实时性问题。现有的实时容错技术主要是在基/副版本技术的基础上发展起来，根据待处理的实时任务性质进一步产生了处理周期任务的 RM 算法[4]、RMFF 算法[5]、处理非周期任务的 PB-Overloading 算法[6]和处理混合任务的 SS 算法[7]、RB 算法[8-9]、HFTS 算法[10]。

在处理混合任务调度的过程中，调度算法针对非周期任务一般采用预留时间的方法，预留时间是指任务周期内划分出的一部分固定处理非周期任务的时间，但是为了保证周期任务的可调度性，人们选择单位时间内所有处理机上周期任务运行完毕后剩余的时间作为预留时间。由于预留时间是在周期任务运行最坏情况下得到的，因此当非周期任务在某个周期内频繁到来时无法得到充分响应。

针对上述问题，本文将周期任务集根据处理机的数量划分为周期任务子集，每个处理机调度不同的周期任务子集，由于每个处理机上运行的周期任务不同，因此计算出的预留时间也不同。当非周期任务由于频繁到来而难以充分响应时，可以将这些任务调度到那些周期任务负担较轻的处理机运行，这样既可以有效保证周期任务的调度，也可以充分响应非周期任务的要求，从而提高了控制系统容错调度的效率。

2 系统模型描述

考虑一个由多台处理机构成的分布式控制系统，该系统需要处理多个周期任务和非周期任务，其形式化描述如下：

(1)一组周期任务

$$\Gamma = \{\tau_1, \tau_2, \cdots, \tau_N\}, \tau_i = (C_i, T_i),\ i = 1, 2, \cdots, N \quad (1)$$

其中，N 代表周期任务数；C_i 代表任务 τ_i 在一个周期内的最大执行时间；T_i 代表任务 τ_i 的最小执行间隔—即周期。设任务的截止时限等于其周期。

(2)一组非周期任务

$$J = \{j_1, j_2, \cdots, j_N\}, j_i = (a_i, c_i, D_i, d_i),\ i = 1, 2, \cdots, K \quad (2)$$

其中，K 代表某一时刻存在于系统中的非周期任务数；a_i 是非周期任务随机到达系统的时间；c_i 为任务 j_i 最大执行时间；

基金项目：科研院所技术开发专项基金资助项目(国科计字[2000]056 号)
作者简介：孟庆春(1979－)，男，助理研究员、博士，主研方向：机动车检测，应用于机动车检测的网络技术；刘云卿，研究员
收稿日期：2009-06-20 **E-mail**：mqc_794388@hotmail.com

D_i 为任务 j_i 相对截止时限； $d_i = a_i + D_i$ 为任务 j_i 绝对截止时限。

(3)一组处理机

$$p = \{p_1, p_2, \cdots, p_M\},\ m = 1,2,\cdots,M \tag{3}$$

其中， M 代表所需处理机数。设处理机类型相同，从而一个实时任务即使运行在不同处理机上，其执行时间仍然相同。

主/副版本技术将实时任务的主版本和副版本分别调度在 2 个不同的处理机上。式(1)中的周期任务和式(2)中的非周期任务的副版本可以表述为

$$B\Gamma = \{\beta_1, \beta_2, \cdots \beta_N\}, \tau_i \models (B_i, T_i),\ i = 1,2,\cdots,N \tag{4}$$

$$BJ = \{v_1, v_2, \cdots, v_N\}, v_i \models (a_i, c_i, D_i, d_i),\ i = 1,2,\cdots,K \tag{5}$$

其中， β_i 为周期任务 τ_i 的副版本； B_i 为 β_i 的最大执行时间，一般来说 $B_i \leqslant C_i$ ，因为副版本既可以是主版本的完全备份，实现主版本需要完成的所有功能，也可以是简化版本，只完成故障状态下主版本必不可少的功能； T_i 为 β_i 的最短执行间隔，与 τ_i 的周期相等。 v_i 为非周期任务 j_i 的副版本， b_i 是 v_i 的执行时间，基于同样的原因 $b_i \leqslant c_i$ ，相对截止时限 D_i 和绝对截止时限 d_i 与非周期任务主版本中的对应参数相等。

3 预留时间机制

上文提到的处理混合任务的调度算法中主要有 SS 算法、RB 算法和 HFTS 算法，其中，SS 算法通过建立被动任务取得未被周期任务使用的处理机时间来执行非周期任务，由于该算法利用试凑的递推方法来寻找最优方案，具有很大的系统开销;RB 算法和 HFTS 算法通过在单位时间内预留时间来处理非周期任务，上述 2 种算法中给出了不影响周期任务可调度性的最大预留时间，它们也能获得与 SS 算法同样的最优处理时间，因此，本文延用 RB 算法和 HFTS 算法中采用的预留时间的做法。

预留时间 R_t 主要与单位时间的长度 l 和单位时间内周期任务所占用的处理机时间 Γ_t 有关，具体可表示为 $R_t = l - \Gamma_t$ ，其中， l 定义为处理机上各周期任务执行周期的最大公约数， Γ_t 的定义比较复杂，因为不同时刻周期任务对处理机的占用率不同，有时高达 100%，有时只有 0%(见文献[9])，所以 Γ_t 定义为周期任务在单位时间内占用处理机所需的最长时间。

在分布式容错系统中，当某处理机 p_m 发生故障时，设处理机 p_k 执行周期任务时的利用率为 $\rho_k (k = 1,2,\cdots,M, k \neq m)$ ，则在极端情况下 p_k 的利用率为 $\rho = \max(\rho_1, \rho_2, \cdots, \rho_M)$ ，根据文献[10]的推导 Γ_t 就可以通过 $l \cdot \rho_{\max}$ 求得。

4 改进的调度算法

在本文提出的改进算法中，每个处理机上运行的任务包括两类：周期任务和非周期任务，其中周期任务是由周期任务子集中包含的周期任务元素规定的，周期任务子集在调度算法执行前根据周期任务子集划分算法获得；非周期任务则在预留时间内执行，预留时间由 $l - l \cdot \rho_{\max}$ 计算所得。

4.1 周期任务子集的划分

在叙述周期任务子集划分的算法前，首先定义一些相关的概念：

(1)对于前文提到的周期任务集 Γ ，计算 $R_i = C_i / T_i$ ， $i = 1,2,\cdots,N$ ，令 $R_{\min} = \min(R_1, R_2, \cdots, R_N)$ ， $R_{\max} = \max(R_1, R_2, \cdots, R_N)$ ，同时计算 $Dis = (R_{\max} - R_{\min} + 0.01)/N$ 。

(2)定义集合 $SET = \{SET_i (1 \leqslant i \leqslant N)\}$ ，子集 SET_i 中的元素 e_{ij} 满足 $R_{\min} + (i\text{-}1) \times Dis \leqslant e_{ij} < R_{\min} + i \times Dis$ 。

(3)根据 R_i 的大小将其插入适当的集合 SET_i ，定义集合 $COUNT = \{count_i (1 \leqslant i \leqslant N)\}$ ，其中，元素 $count_i$ 表示集合 SET_i 中元素的累加和。

(4)定义集合 $SIFT = \{SIFT_i (1 \leqslant i \leqslant 2)\}$ ，对于集合 $COUNT$ 中的元素 $count_i$ ，如果其值不小于 1 ，则将其对应的子集 SET_i 放入集合 $SIFT$ 的子集 $SIFT_0$ 中，否则放入子集 $SIFT_1$ 中。

(5)定义周期任务集合 $TASK = \{TASK_i (1 \leqslant i \leqslant N)\}$ ，其中， $TASK_i$ 表示运行在处理机 i 上的周期任务子集。

根据上述形式化定义，选择周期任务子集的算法如下：

(1)检查集合 $SIFT_0$ 是否为空集，如果为空表明找到周期任务集合(集合 $SIFT_1$ 即为 $TASK$)算法结束，否则若算法迭代达到 N^3 次依然无法结束则表明无法找到周期任务集合算法退出，若以上条件均不满足则转步骤(2)。

(2)从集合 $SIFT_0$ 中随机选出元素 SE_j ，同时从集合 $SIFT_1$ 中随机选出元素 ST_k ，随机交换集合 SE_j 和 ST_k 中的元素。

(3)根据集合 SE_j 中经过交换后新的元素值重新计算累加和 SUM_1 ，如果 $SUM_1 \geqslant 1$ 则撤消前述交换过程，否则根据集合 SE_j 中经过交换后新的元素值重新计算累加和 SUM_2 ，如果 $SUM_2 < 1$ ，则将从集合 $SIFT_1$ 中删除元素 ST_k ，同时将元素 ST_k 插入集合 $SIFT_0$ ，上述过程执行结束转步骤(1)。

4.2 非周期任务的调度

非周期任务通过分配预留时间得到有效响应，其调度采用 EDF 算法[11]，该算法为截止时限最近的非周期任务分配最高的执行优先级，高优先级的任务到来时，低优先级的任务必须立即让出处理机的使用权。由于 EDF 算法属于动态调度算法，因此可以有效保证处理机的利用率。

某一时刻有大量非周期任务到来时，为了及时响应任务的请求，调度程序执行下述处理过程：

(1)调度程序首先将到来的非周期任务尝试分配到周期任务负担较轻的处理机，如果上述处理机都被更高优先级的非周期任务占用，则执行过程(2)；

(2)将非周期任务尝试分配到其他处理机，如果无法满足分配，则执行过程(3)；

(3)调度程序仅调度优先级较高的非周期任务，延时调度某些低优先级的非周期任务，如果某些非周期任务不能满足延时响应的要求，则执行过程(4)；

(4)调度程序显示无法有效响应非周期任务的错误，控制系统停止所有任务执行。

4.3 实例分析

定义周期任务集为 $\Gamma = \{\tau_1, \tau_2, \cdots, \tau_5\}, N = 5$ ，且 $\tau_1 = (0.2, 0.5), \tau_2 = (0.3, 0.5), \tau_3 = (0.1, 1.0), \tau_4 = (0.7, 1.0), \tau_5 = (0.3, 1.0)$ ，计算 $R_1 = 0.4, R_2 = 0.6, R_3 = 0.1, R_4 = 0.7, R_5 = 0.3$ ，其中， $R_{\min} = 0.1$ ， $R_{\max} = 0.7$ ，计算 $Dis = (0.7\text{-}0.1\text{+}0.01)/5 = 0.122$ 。

定义集合 $SET = \{SET_i (1 \leqslant i \leqslant 5)\}$ ，其中， $SET_1 = \{0.1\}$ ， $SET_2 = \{0.3\}$ ， $SET_3 = \{0.4\}$ ， $SET_4 = \{\}$ ， $SET_5 = \{0.6, 0.7\}$ 。定义集合 $COUNT = \{count_i (1 \leqslant i \leqslant 5)\}$ ，其中， $count_1 = 0.1$ ， $count_2 = 0.3$ ， $count_3 = 0.4$ ， $count_4 = 0$ ， $count_5 = 1.3$ 。

定义集合 $SIFT = \{SIFT_i (1 \leqslant i \leqslant 2)\}$ ，其中， $SIFT_0 = \{SET_5\}$ ， $SIFT_1 = \{SET_1, SET_2, SET_3, SET_4\}$ 。

执行周期任务子集划分算法，将集合 $SET_5=\{0.6,0.7\}$ 中的元素 0.6 和集合 $SET_1=\{0.1\}$ 中的元素 0.1 交换，产生新的集合 $SET_1=\{0.6\}$ 和集合 $SET_5=\{0.1,0.7\}$。由于集合 SET_5 中的元素累加和小于 1，因此从集合 $SIFT_0$ 删除元素 SET_5，同时将元素 SET_5 加入集合 $SIFT_1$，由于集合 $SIFT_0$ 为空，算法结束。

经过周期任务子集划分，各个处理机上执行的周期任务如下：处理机 1 上执行周期任务 τ_2，周期任务占用处理机时间 60%；处理机 2 上执行周期任务 τ_5，周期任务占用处理机时间 30%；处理机 3 上执行周期任务 τ_1，周期任务占用处理机时间 40%；处理机 4 上不执行周期任务；处理机 5 上执行周期任务 τ_3,τ_4，周期任务占用处理机时间 80%。这样处理机 4 完全空闲可以充分响应非周期任务，相对于原来在每个处理机上预留 70% 处理机时间作为周期任务时间的算法效率有所提高。

5 结束语

本文针对分布式控制系统的应用需求，提出了同时调度周期任务和非周期任务的容错调度算法。该算法首先将周期任务集根据处理机的数量划分为周期任务子集，并将每个子集中的周期任务分配给相应的处理机，根据不同处理机上运行的周期任务参数确定处理非周期任务请求的预留时间，当某个周期中非周期任务频繁到来时可以将这些任务分配给一些周期任务负担较轻的处理机，进一步增加了处理机的利用率，同时也提高了控制系统的容错调度效率。尽管如此，上述容错调度算法的计算复杂度有所增加，因此，如何在计算复杂度增加不大的前提下，进一步提高调度算法性能，是今后研究中需要解决的主要问题。

参考文献

[1] Xu Lihao, Bruck J. Deterministic Voting in Distributed Systems Using Error-correcting Codes[J]. IEEE Transactions on Parallel and Distributed Systems, 1998, 9(8): 813-824.

[2] Lin K H, Shin K G. Damage Assessment for Optimal Rollback Recovery[J]. IEEE Transactions on Computers, 1998, 47(5): 603-613.

[3] Davoli R, Giachini L A, Amoroso A, et al. Parallel Computing in Networks with Parallax[J]. IEEE Transactions on Parallel and Distributed Systems, 1996, 7(4): 371-384.

[4] Liu C L, Layland J W. Scheduling Algorithms for Multi-programming in a Hard-real-time Environment[J]. Journal of the Association for Computing Machinery, 1973, 20(1): 46-61.

[5] Bertossi A A, Mancini L V, Rossini F. Fault-tolerant Rate-monotonic First-fit Scheduling in Hard-real-time Systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(9): 934-945.

[6] Al-Omari R, Somani A K, Manimaran G. A New Fault-tolerant Technique for Improving Schedulability in Multiprocessor Real-time Systems[C]//Proc. of the 15th IEEE Parallel and Distributed Processing Symposium. San Francisco, USA: IEEE Press, 2001.

[7] Luchozky J P, Ramos T S. An Optimal Algorithm for Scheduling Soft Aperiodic Tasks in Fixed-priority Preemptive Systems[C]//Proc. of the 13th Real-time System Symposium. Phoenix, USA: [s. n.], 1992.

[8] Shin K G, Chang Y C. A Reservation-based Algorithm for Scheduling Soft-aperiodic Tasks in Fixed-priority Preemptive System[C]//Proc. of the 13th Real-time System Symposium. Phoenix, USA: [s. n.], 1992.

[9] Shin K G, Chang Y C. A Reservation-based Algorithm for Scheduling Both Periodic and Aperiodic Real-time Tasks[J]. IEEE Transactions on Computer, 1995, 44(12): 1409-1415.

[10] 阳春华, 桂卫华, 计　莉. 基于多处理机的混合实时任务容错调度[J]. 计算机学报, 2003, 26(11): 1479-1486.

[11] 刘　怀, 费树岷. 基于EDF的分布式控制系统容错调度算法[J]. 软件学报, 2003, 14(8): 1371-1378.

编辑　金胡考

(上接第 14 页)

在 DAS 系统实现 SBBR 算法过程中，主要对 Oracle, Mysql 和 SQL Server 3 种数据库分别进行了测试，在误差允许的范围内，基本上都能成功地从查询结果集中按照设定的内存块大小取出数据，图 3 给出了在 eclipse3.2 环境中测试含有 LOB 数据的 Oracle 数据库时，当设定内存块大小为 10 KB 时执行 SBBR 算法的情况。

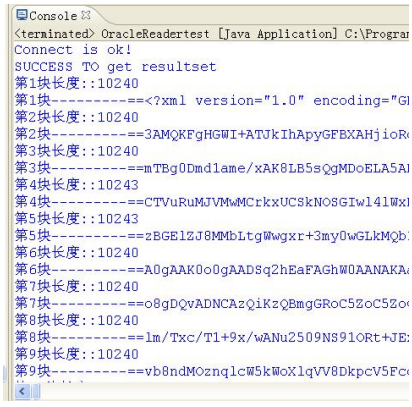

图 3　SBBR 读取 Oracle 数据库的测试结果

4 结束语

本文对 ResultSet 对象的读取策略进行了分析，提出一种数据库结果集块状读取算法。本文的工作将对基于 Web 数据库应用系统更好地处理针对包含 LOB 对象的数据库的查询提供帮助。

参考文献

[1] Sun Microsystems, Inc.. JDK 5.0 Documentation[EB/OL]. (2004-10-30). http://java.sun.com/j2se/1.5.0/docs/.

[2] OGSA-DAI: Using OGSA-DAI with Binary Large Objects(BLOBs) [EB/OL]. (2006-10-20). http://www.ogsadai.org.uk/documentation/scenarios/working_with_blobs/.

[3] Monjid A. Working with Binary Large Objects(BLOBs)[EB/OL]. (2008-03-20). http://www.c-sharpcorner.com/UploadFile/Ashush/UsingBlobs03222008142343PM/UsingBlobs.aspx.

[4] 陈启转, 陈　坚. 基于 JDBC 的 Oracle 大对象(LOBs)操作[J]. 微型电脑应用, 2000, 16(12): 45-47.

[5] 张西广, 谢建军, 杨德婷, 等. 科学数据网格中数据传输技术的研究与改进[J]. 微电子学与计算机, 2007, 24(12): 118-122.

编辑　索书志