第41卷 第6期 西 安 交 通 大 学 学 报 Vol.41 №6
2007年6月 JOURNAL OF XI'AN JIAOTONG UNIVERSITY Jun. 2007

基于预测表的容错实时调度算法

刘东，张春元，李瑞，黄影，李毅
（国防科技大学计算机学院，410073，长沙）

摘要：为了更精确地预测软件容错模型中的容错实时调度算法主部分可否执行，提出了基于预测表的容错调度算法(PTBA).该算法充分考虑了当前时间至替代部分通知时间之间的任务执行状况，以该时间段内通知时间的先后顺序精确地预测主部分的执行情况，并为待执行主部分构建预测表.当主部分不发生错误时，算法将依照预测表进行任务调度.模拟结果表明，利用PTBA可获得更多的主部分执行时间，降低了CPU消耗.PTBA适用于软件错误率较低的应用环境，特别是当任务的周期较短时，能够以较小的调度开销获得较高的调度性能.

关键词：软件容错；容错调度；预测表
中图分类号：TP316.2 **文献标识码**：A **文章编号**：0253-987X(2007)06-0697-05

Fault-Tolerant Real-Time Scheduling Algorithm Based on Prediction-Table

Liu Dong, Zhang Chunyuan, Li Rui, Huang Ying, Li Yi
(School of Computer, National University of Defense Technology, Changsha 410073, China)

Abstract: In order to predict more accurately whether the primary version of the fault-tolerant scheduling algorithm in software fault-tolerant model is executable, a new algorithm named prediction-table based algorithm (PTBA) is put forward, in which the executing situation of tasks between the current time and the notification time of alternate version is fully considered. The executing situation of primary version is accurately predicted according to the sequence of notification time in that time interval, and the prediction table is created for the primary version to be executed. If primary versions do not fail, the task scheduling will be carried out based on the table. Simulation result shows that PTBA can acquire more execution time of the primary version and decrease CPU cost. PTBA is applicable for the circumstance with low software fault rate, especially for short-period tasks.

Keywords: software fault-tolerance; fault-tolerant scheduling; prediction-table

为了提高实时系统的可靠性，实时系统中的任务除了需要满足功能和时限要求之外，还需要保证系统在任务发生单个或多个错误时仍旧正常运行.目前，针对此问题已提出了多种容错模型.分布式容错调度模型[1-2]为实时系统提供了软、硬件级别的容错能力，模型中的任务具有基版本和副版本，正常情况下执行基版本，而当基版本失效时，将调度其他处理器上的副版本.软件容错模型[3-5]不需要多处理器来支持，只为每个任务提供主部分和替代部分2个版本，正常情况下系统运行主部分，并为替代部分预留足够的时间片断，而当主部分执行失败时，系统执行替代部分.

软件容错调度算法是主部分、替代部分的调度机制，相关研究包括BCE调度算法，其包含了Basic、CAT (Check Available Time)和EIT (Eliminate Idle Time) 3部分内容，以及BCE改进算法PKSA和CUBA[4-5]，当处理器利用率较高时，前者的主部分丢失率比较高，而后者的预测精度不高.

收稿日期：2006-10-08. 作者简介：刘东(1981～)，男，博士生；张春元(联系人)，男，教授，博士生导师. 基金项目：国家自然科学基金资助项目(60573103).

为了更精确地预测软件容错模型主部分的执行情况，本文提出了基于预测表(PT)的容错实时调度算法(PTBA). 在当前时间至通知时间之内，预测抢占待执行主部分所有任务的运行状态，从而构建反映任务执行情况的预测表，若该时段内的主部分发生错误失效，调度算法将依照预测表执行任务.

1 软件容错模型

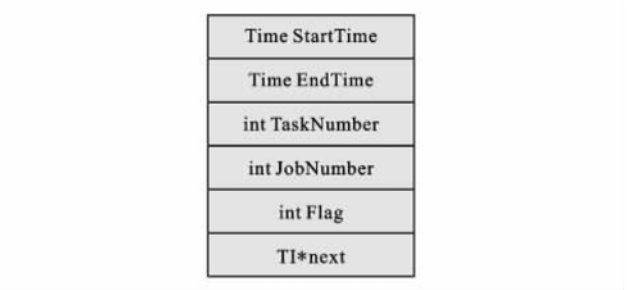
在软件容错模型中，实时系统包含周期性任务集合 $\tau=\{\tau_1,\tau_2,\cdots,\tau_n\}$，每个任务 τ_i 的周期为 T_i. τ_i 的每次执行称为一个请求，每个请求的执行时间为 e_i. τ_i 的第 j 个请求用 J_{ij} 表示，其中 $1\leqslant i\leqslant n, j\geqslant 1$. J_{ij} 的到达时间为 $(j-1)T_i$，且必须在 jT_i 之前完成. 定义 $r_{ij}=(j-1)T_i$ 为 J_{ij} 的请求时间，$d_{ij}=jT_i$ 为 J_{ij} 的截止期；定义计划周期 $T=\mathrm{LCM}(T_1,T_2,\cdots,T_n)$. 任务每经过一个 T 将会重复执行，不失一般性应仅考虑所有任务在一个计划周期内的调度情况，并设所有任务从0时刻开始执行. τ_i 具有2个独立的程序版本：主部分 P_i 和替代部分 A_i. P_i 的执行时间为 p_i，A_i 的执行时间为 a_i，通常 $p_i\geqslant a_i$，则 $\tau_i=(T_i, p_i, a_i)$ 可以惟一确定任务 τ_i. 相应地，J_{ij} 包含主部分 P_{ij} 和替代部分 A_{ij}. 定义软件错误概率 F 来表示执行完毕后的任务主部分发生错误的概率.

在计划周期内，J_{ij} 必须在 d_{ij} 内完成 P_{ij} 或 A_{ij}. A_{ij} 的开始时间称为通知时间 v_{ij}，若 P_{ij} 在 v_{ij} 之前全部完成，则撤销执行 A_{ij}，否则撤销执行 P_{ij}，转而开始执行 A_{ij}. 主部分包含了更多的功能，如果执行成功，将会获得更佳的操作结果，但主部分的执行复杂度比较高，所以难以保证其可靠性. 替代部分只具有最简单的功能，它产生最基本、可接受的操作结果，并易于测试和验证，因此它的执行被认为是完全正确的. 由于主部分操作结果的质量更高，因此调度算法应当尽可能地满足主部分优先执行.

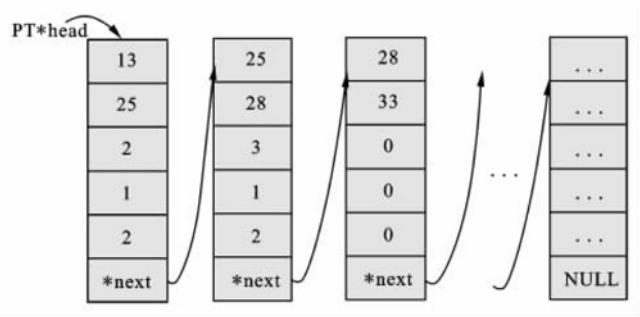
2 基于预测表的容错实时调度算法

主部分的执行存在复杂度、可靠性问题，通常情况下 $F>0$，这样就无法确定主部分和替代部分的最终调度结果. 假如 $F=0$，也就是主部分不发生错误失效，则可以确定调度算法在任意时刻 t 的调度状态，那么针对任务集 τ_i 的调度是确定的. 因此，在预测主部分 P_{ij} 可否执行时，如果认为抢占 P_{ij} 的主部分 P_{mn} 的执行过程不会发生失效，则可以精确预测 P_{ij} 的执行情况，预测结果也将是 P_{ij} 及抢占 P_{ij} 的主部分和替代部分组成的时间表，可以称之为预测表，而基于预测表的容错实时调度算法称为PTBA算法.

预测表是由多个时间段组成的单向链表，如图1所示. 每个时间段包含了该时间段的使用信息，其中包括时间段的占有者(任务编号)和时间段的起始/终止时间. 若时间段被主部分占有，则占有者标志 Flag 为2；若时间段被替代部分占有，则 Flag 为1；若时间段为空闲，则 Flag 为0.



(a)时间段的结构定义



(b)预测表的结构

图1 预测表的结构

定理1 设当前时间为 t，且主部分 P_{ij} 已经到达，替代部分 A_{ij} 的通知时间为 v_{ij}，在 $[t,v_{ij}]$ 区间已经被占用的时间总和为 I，则 P_{ij} 的可用时间为 $T_A=v_{ij}-t-I$.

证明 由于 P_{ij} 必须在截止期 v_{ij} 之前完成，因此可供 P_{ij} 使用的时间最大为 $v_{ij}-t$. 除 P_{ij} 之外，占用 $[t,v_{ij}]$ 区间的任务还包括抢占 P_{ij} 的主部分 P_{mn}，以及由于主部分被撤销或发生错误而必须执行的替代部分. 在构建预测表时，如果 P_{mn} 没有足够的执行时间，则撤销 P_{mn} 而保留 A_{mn}，因此在 $[t,v_{ij}]$ 区间可以抢占 P_{ij} 的主部分已经事先被检测完毕，而检测 P_{ij} 的可执行性是精确检测. 上述主部分和替代部分的执行时间总和为 I，因此最终可供 P_{ij} 使用的空闲时间总和 $T_A=v_{ij}-t-I$，证毕.

定理1为主部分的可执行性提供了精确的判定方法. 如果定理1计算出 P_{ij} 的可用时间 $T_A\geqslant P_i$，则 P_{ij} 一定可以执行；如果定理1计算出 P_{ij} 的可用时

间 $T_A < p_i$，则 P_{ij} 一定不可执行.

如果在[t, v_{ij}]区间内的主部分不发生错误失效，则预测表在[t, v_{ij}]区间内可反映任务的实际执行情况. 因此，下一调度时间位于[t, v_{ij}]，任务调度将可以依照事先制订好的预测表操作，而不需要再次对[t, v_{ij}]区间内的主部分进行预测，从而节省了调度开销.

设当前时间为 t，待预测的主部分为 P_{ij}，其替代部分 A_{ij} 的通知时间为 v_{ij}，预测表构建算法描述如下.

步骤 1：预测表初始化为[t, v_{ij}]，且不被任何任务占用，将[t, v_{ij}]区间内由反向调度算法[3]分配的替代部分执行时间复制到预测表，再将替代部分按通知时间先后排序，依次检测表中的替代部分.

步骤 2：设预测表中存在未被检测的替代部分 A_{mn}，其对应的主部分为 P_{mn}，如果 P_{mn} 发生错误或已经被撤销，转步骤 2 检测预测表中下一个替代部分，否则下转步骤 3. 如果预测表中没有未被检测的替代部分，下转步骤 4.

步骤 3：利用定理 1 检测预测表中的 P_{mn} 可用时间是否可供 P_{mn} 执行，如果 P_{mn} 可以执行，将 P_{mn} 的执行时间添加到预测表中，并撤销 A_{mn} 占用的执行时间；如果 P_{mn} 没有足够的执行时间，撤销 P_{mn}，保留 A_{mn}，上转步骤 2.

步骤 4：利用定理 1 检测预测表中的空闲时间是否可供 P_{ij} 执行，如果 P_{ij} 可以执行，将 P_{ij} 的执行时间添加到预测表中，返回“预测表构建成功”；如果 P_{ij} 没有足够的执行时间，撤销 P_{ij}，返回“预测表构建失败”.

定理 2　在上述预测表的构建算法中，若 P_{mn} 可以执行且撤销了 A_{mn} 占用的时间段，则不需要调整其余替代部分的执行时间.

证明　设 A_{mn} 的通知时间为 v_{mn}，当撤销 A_{mn} 时，在 v_{mn} 之前的替代部分为 A_{pq}，在 v_{mn} 之后的替代部分为 A'_{pq}. 在构建预测表时，由于要依据替代部分的通知时间作为检测顺序，因此可知 A_{pq} 已经被检测完毕，这意味着 P_{pq} 已经被撤销，所以调整 A_{pq} 的执行时间对 P_{pq} 已经没有意义. 又由于 A'_{pq} 在 v_{mn} 之后，因此 A_{mn} 的撤销并不影响 A'_{pq}，证毕.

由定理 2 知，本文依据替代部分的通知时间顺序构建预测表的方法，不仅减小了调度开销，同时也精确地反映了[t, v_{ij}]区间内的任务执行状态，同时避免了 PKSA 算法中 K 步试探策略不能调整替代部分执行时间的缺陷，进而达到了精确预测主部分可执行性的目的. PTBA 调度算法描述如图 2 所示.

```
Procedure PTBA(host-primary)
  static: t,当前时间，初始值为0;
          S,当前调度状态，初始值为"not PT";
  利用反向速度单调调度算法分配所有替代部分;
  while (1)
    if S="not PT" then
      if 替代部分到达 then 执行该替代任务;
      else if 处理器空闲 then 使用EIT算法[3] 消除空闲时间
      else if 主部分到达 then
           在所有已经释放的主部分中（不包括已经完成、撤销
           或失效的主部分），设P_ij的替代部分具有最小的通知时间;
        if PT(P_ij)="预测表构建成功" then S="PT";
        else 撤销P_ij;
        end
        continue;
      end
    else if S="PT" then
      设P_ij为 host-primary;
      if 预测表中当前时间段被P_ij占用 then
       调度P_ij;
      if P_ij完成 then
        if P_ij执行中未发生错误 then 撤销A_ij;
        else 保留A_ij;
        end
        S="not PT";
        根据需要调整其他替代部分的执行时间;
      end
    else if 预测表中当前时间段被P_mn(P_mn≠P_ij)占用 then
      调度P_mn;
      if P_mn完成 and P_mn执行过程中出现错误 then
        S="not PT";
        根据需要调整其他替代部分的执行时间;
      end
    else if 预测表中当前时间段为A_mn then 调度A_mn;
    end
  end
  设置下一时间t;
end
```

图 2　PTBA 调度算法

3　模拟结果

3.1　评价标准

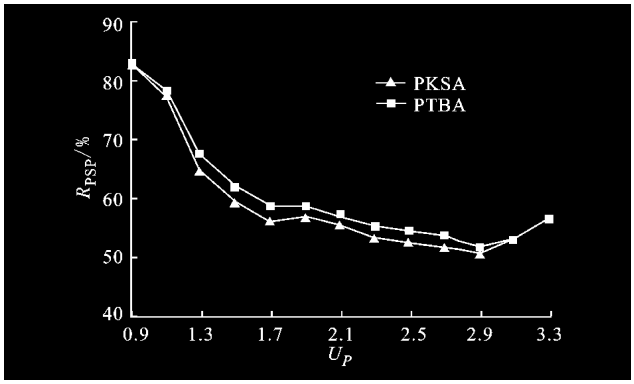
本文采用成功主部分使用时间比率 R_{PSP}（Percent for Successful Primaries，PSP）和浪费的 CPU 时间 W 作为最终调度结果的评价标准. R_{PSP} 可用执行成功的主部分所用时间占模拟时间的百分比来衡量，以评价调度算法的调度性能. 若主部分因执行时间不够而被取消，此时主部分已经占用的 CPU 时间为 W，该参数可表示主部分可执行性的预测精度. 由于 PKSA 算法的调度效果优于 CUBA[5]，因此本文主要对 PTBA 和 PKSA 进行比较.

3.2　不同 U_P 下 PTBA 与 PKSA 的比较

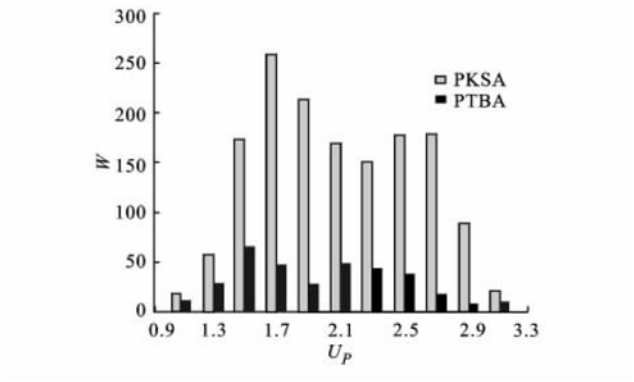
当 $F=0.05$，PKSA 的试探步数 $K=4$ 时，模拟不同 T_i、p_i 和 a_i 的任务集，每个任务集含有 5 个任务，其中 T_i 位于[10，150]时间单位区间，a_i 的值为[0.1，0.2]倍的 T_i. 通过调整主部分执行时间相对于任务周期所占用的比例 p_i/T_i，来改变主部分利用率 U_P，从而间接调整 CPU 利用率 U. 通常，任务

集的计划周期比较大，因此在实际模拟中采用的模拟时间为10 000个时间单位.

对每种条件模拟200次，以便获得最一般的模拟结果. 对每一种U_P条件下模拟得到的200个R_{PSP}和W进行求和平均，从而得到各种U_P条件下的PTBA与PKSA的比较结果，如图3所示.



(a)R_{PSP}



(b)W

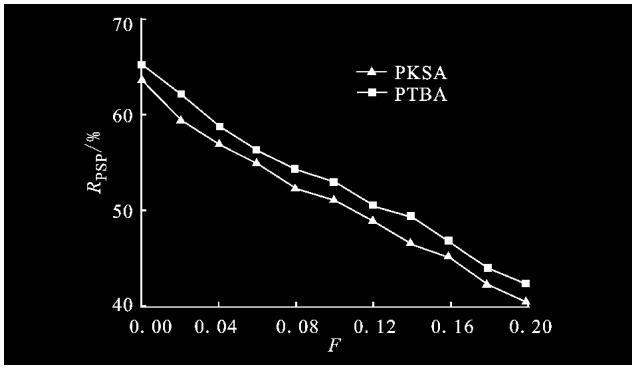
图3 不同U_P下的PTBA与PKSA的比较

模拟结果表明，随着U_P的增高，PTBA与PKSA的性能均逐渐降低. 这是因为，可供主部分调度的空闲时间有限，而主部分执行时间的增加将导致主部分无法获得足够的执行时间. 当这2种算法预测到主部分无法完成时，将取消执行主部分，因此主部分总的执行时间降低.

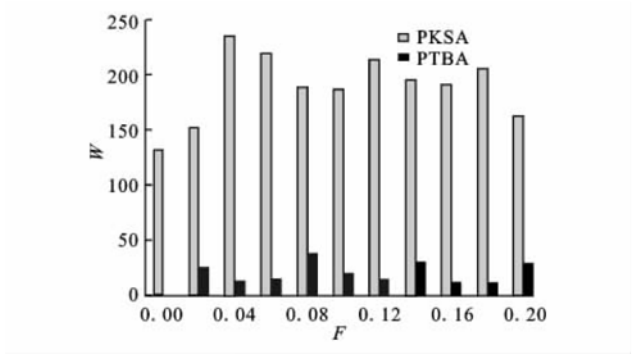
当$U_P<1.1$时，2种算法的性能接近，此时主部分的执行时间较短，均具有足够的完成时间. 当$1.1<U_P<3$时，PTBA比PKSA的调度性能更好. 这是因为：PKSA在预测主部分时并不调整替代部分的执行时间，因此当CPU利用率增加，其预测精度降低；PTBA以替代部分的通知时间作为选择顺序，从而避免了调整替代部分的执行时间，由此获得较高的预测精度，并为更多主部分提供了可执行的条件. 当$U_P>3$时，2种算法的性能又再次接近. 这是因为，主部分的执行时间增大到一定程度，2种预测算法均可以预测出主部分无法完成的情况，从而获得相近的调度结果.

3.3 不同F下PTBA与PKSA的比较

PKSA的试探步数$K=4$，当$p_i/T_i=0.37(1\leqslant i\leqslant 5)$，即$U_P=1.23$时，在不同$F$下的PTBA与PKSA的模拟结果如图4所示(每种$F$均模拟100次).



(a)R_{PSP}



(b)W

图4 不同F下PTBA与PKSA的比较

模拟结果表明，对于不同的F，PTBA比PKSA均获得较多的主部分执行时间. 随着F的增加，这2种算法的主部分成功执行所占用的时间均逐渐下降，这是因为F的增加导致主部分执行失效也在增加，从而需要执行更多的替代部分，造成主部分可以使用的时间减少. 当$F<0.02$时，由于主部分发生错误的概率比较低，因此PTBA获得了更佳的预测结果，其$W=0$，而PKSA并不调整替代部分的执行时间，因此仍旧浪费较多的CPU时间. 随着F的增加，PTBA的W逐渐增加，但是并没有出现次线性增长的趋势，这是因为影响PTBA的因素除了F之外，还包括执行过程中任务集自身的因素. 在PKSA中，F的增加对W的影响并不大，这是因为统计结果只考虑已成功执行的主部分，而F的增加将导致成功执行的主部分逐渐减少，因此尽管F的增加造成了主部分可执行性预测精度越来越小，但总体上

W 保持稳定.

4　结束语

本文针对软件容错模型提出了 PTBA 算法，该算法根据替代部分通知时间的先后顺序构建预测表，避免了在撤销替代部分时需调整其他替代部分的情况，同时能够精确预测主部分的执行情况. 当主部分不发生错误失效时，算法依据预测表进行任务调度，减小了调度开销. 经过模拟验证和同类算法的比较得知，PTBA 能够获得更多的主部分执行时间和更少的 W. 目前，本文的算法已经在 RT-Linux 中进行了前期移植，并通过了功能验证. 下一阶段拟采用故障注入法，进一步验证实际应用算法的有效性. PTBA 算法适用于软件错误率较低的应用环境，当任务的周期较短时，预测表构建过程所花时间比较少，而且能够以较小的调度开销获得较高的调度性能.

参考文献：

[1] Al-Omari R, Somani A K, Manimaran G. Efficient overloading techniques for primary-backup scheduling in real-time systems [J]. Journal of Parallel and Distributed Computing, 2004, 64: 629-648.

[2] de Araújo Lima G M. Fault tolerance in fixed-priority hard real-time distributed systems [D]. York, England: Department of Computer Science, University of York, 2003.

[3] Han Ching-Chih, Shin K G, Wu Jian. A fault-tolerant scheduling algorithm for real-time periodic tasks with possible software faults [J]. IEEE Transactions on Computers, 2003, 52(3): 362-372.

[4] 韩建军，李庆华，Essa A A. 基于软件容错的动态实时调度算法 [J]. 计算机研究与发展，2005，42(2)：315-321.
Han Jianjun, Li Qinghua, Essa A A. A dynamic real-time scheduling algorithm with software fault-tolerance [J]. Journal of Computer Research and Development, 2005, 42(2): 315-321.

[5] 李庆华，韩建军，Essa A A，等. 硬实时系统中基于软件容错的动态调度算法 [J]. 软件学报，2005，16(1)：101-107.
Li Qinghua, Han Jianjun, Essa A A, et al. Dynamic scheduling algorithms with software fault-tolerance in hard real-time systems [J]. Journal of Software, 2005, 16(1): 101-107.

（编辑　苗凌）