异构分布式系统中实时周期任务的容错调度算法

罗 威 阳富民 庞丽萍 涂 刚
(华中科技大学计算机科学与技术学院 武汉 430074)

摘 要 提出一个基于抢占性实时周期任务的可靠性调度模型,该模型与现有可靠性模型相比充分考虑了单处理机故障容错情况下的系统可靠性,因而更加接近现实和精确.在此基础上,提出一个基于异构分布式系统的实时容错调度算法 IRDFTAHS,IRDFTAHS 算法以提高系统的可靠性为目标来进行任务的分配,从而在不增加硬件代价的前提条件下通过调度增加了系统的可靠性.该算法同时支持主动和被动两种方式的副版本,使得容错调度算法具有更大的灵活性.最后,通过仿真实验对 IRDFTAHS 和现有的调度算法在几个方面进行比较.实验结果表明,IRDFTAHS 算法的综合性能优于现有算法.

关键词 实时周期任务;容错;主/副版本;异构分布式系统;可靠性
中图法分类号 TP306

A Real-Time Fault-Tolerant Scheduling Algorithm of Periodic Tasks in Heterogeneous Distributed Systems

LUO Wei YANG Fu-Min PANG Li-Ping TU Gang
(*School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan* 430074)

Abstract This paper proposes a novel reliability model based on preemptive periodic tasks. Compared with existing reliability models in literature, the proposed reliability model can be one-processor-failed fault-tolerant, which makes it more realistic and precise. Moreover, a real-time fault-tolerant scheduling algorithm based on heterogeneous distributed systems, namely IRDFTAHS, is presented. IRDFTAHS tries to assign tasks copies in a way to improve reliability of system. In addition, IRDFTAHS considers backup copy in both active and passive status, which makes the proposed algorithm more flexible than existing algorithms. Finally, simulation experiments are carried out to compare the algorithm with existing ones in several aspects. The experiments results show that the IRDFTAHS generally performs significantly better than existing algorithms.

Keywords real-time periodic tasks; fault-tolerance; primary/backup copy; heterogeneous distributed systems; reliability

1 引 言

随着高速网络和高性能 PC/工作站的发展,异构分布式系统已经广泛地应用于各种安全关键系统(safety critical systems),如飞行控制系统、核电站控制系统等.这些系统中的任务都具有实时性,即所有任务必须在截止期(期限)内完成,如果任务没有在规定的截止时限前完成,往往会引起灾难性的后果.为了保证实时任务在出现系统软/硬件故障后仍

收稿日期:2007-04-17;修改稿收到日期:2007-08-08.本课题得到国家自然科学基金(60603032)资助.**罗 威**,男,1980 年生,博士研究生,主要研究方向为实时嵌入式操作系统、实时调度算法、容错计算.E-mail: free_xingezi@163.com. **阳富民**,男,1966 年生,教授,主要研究领域为实时嵌入式操作系统. **庞丽萍**,女,1944 年生,教授,博士生导师,主要研究领域为并行分布式系统、实时系统. **涂 刚**,男,1976 年生,博士,副教授,主要研究方向为实时嵌入式操作系统、实时调度算法.

能在截止期限前完成，需要为分布式实时系统提供一定的容错能力，从而提高系统的可靠性. 为了在异构分布式系统中获得高性能，近年来很多学者致力于研究异构分布式系统中的调度算法[1-3]. 虽然这些算法都以可靠性为调度目标，但是它们既不支持实时任务，也不支持容错.

主/副版本备份技术(Primary/Backup copy，P/B)是广泛应用于分布式系统的容错调度方法，它通过在备份处理机上执行备份任务来实现容错. 传统的副版本有三种执行方式：主动方式(active backup copy)[4]、被动方式(passive backup copy)[5]和主/副版本间的重叠方式(P/B overlapping backup copy)[6]. 主动方式副版本的特点是：它与主版本同时运行，两者间无同步关系. 被动方式副版本仅仅在主版本执行失败的时候才启动执行，其优点在于在系统无故障的情况下，无需执行冗余任务(即 backup-copy de-allocation 技术)，同时它允许不同故障状态下的任务的副版本重叠使用(即 backup copy overlapping 技术)，从而提高处理机的利用率. 因此从提高系统可调度性的角度出发，我们希望更多任务的副版本以被动方式运行. P/B 重叠方式的副版本其执行方式介于主动方式和被动方式之间，如文献[6]提出的一种适应性的容错调度算法，该算法通过调节主/副版本间的重叠长度在系统可靠性和性能间取得折衷. 文献[7]提出一种容错调度算法 FTRMFF(Fault-Tolerant Rate-Monotonic First-Fit)，该算法基于抢占性的周期任务，并且同时考虑主动副版本和被动副版本，然而 FTRMFF 算法是基于同构的分布式系统，并且不考虑调度算法的可靠性.

很多学者深入研究了基于异构和同构分布式系统的实时容错调度算法[8-12]. 在同构分布式系统中，实时容错调度算法大多以可调度性作为系统的调度目标[5,12]，例如文献[12]提出一种基于固定优先级调度算法的延迟主动副版本备份技术，该技术通过尽量向后调度主动方式的副版本，并在主版本成功执行时终止副版本的执行来减少备份的冗余度，而本文提出的调度算法是基于异构分布式系统的. 在异构系统中，除了可调度性以外，还必须考虑系统的可靠性. 文献[8]中提出了基于周期任务模型的实时容错调度算法，但是该模型要求所有任务的周期完全相同；文献[10-11]提出的基于异构分布式系统的实时容错调度算法都要求任务是非抢占的非周期任务，不适合抢占性的周期任务，因此在实际应用中有很大的局限性. 为了克服这个问题，最近我们提出一个基于实时容错周期任务的可靠性模型[9]，并在此基础上提出一个可靠性驱动的实时容错调度算法 RDFTAHS，RDFTAHS 以可靠性为驱动来调度实时容错周期任务，从而在不增加硬件代价的情况下通过调度来增加系统的可靠性. 虽然该模型能够在一定程度上度量系统的可靠性，但是该模型是基于系统无故障的情况下的；另一方面 RDFTAHS 算法忽略了被动副版本在系统故障后的可靠性问题. 换句话说，这个模型假设系统故障后在系统恢复故障前不会发生任何其它故障. 显然，在实际应用中，这个模型的要求太苛刻了.

针对文献[9]中提出的可靠性模型和调度算法的缺陷，本文首先提出一个基于抢占性实时周期任务的可靠性调度模型，与现有可靠性模型相比，该模型充分考虑了单处理机故障容错情况下的系统可靠性. 在此基础上，提出一个基于异构分布式系统的实时容错调度算法 IRDFTAHS(Improved Reliability-Driven Fault-Tolerant Algorithm for Heterogeneous Distributed System). IRDFTAHS 算法以提高系统的可调度性为目标进行任务的分配，从而在不增加硬件代价的前提下通过调度增加了系统的可靠性. 本算法同时支持主动和被动两种方式的副版本，使得容错调度算法具有更大的灵活性. 最后，通过仿真实验对 IRDFTAHS 和现有的调度算法在几个方面进行比较. 实验结果表明，IRDFTAHS 算法的综合性能优于现有算法.

2　容错调度模型

本文考虑一个典型的由多处理机和实时任务集构成的异构分布式系统，由以下定义进行形式化描述.

定义 1.　异构分布式系统可描述为一组处理机集合. $\Omega=\{P_1,P_2,\cdots,P_M\}$ 以及处理机的失效向量 $\boldsymbol{R}=(\lambda_1,\lambda_2,\cdots,\lambda_M)$，$\lambda_i$ 为处理机 P_i 的失效率，假设各个处理机之间出错过程相互独立，且符合泊松过程.

定义 2.　一组实时周期任务的主版本集合：

$$\Gamma=\{\tau_1,\tau_2,\tau_3,\cdots,\tau_N\},$$

$$\tau_i=(\boldsymbol{C}_i,T_i),\ i=1,2,\cdots,N,$$

其中，Γ 为主版本的集合，τ_i 为任务 i 的主版本，N 为周期任务的数目，T_i 为任务的周期，这里任务的周期都等于其期限，$\boldsymbol{C}_i$ 为任务主版本的执行时间向量，即 $\boldsymbol{C}_i=[c(i,1),c(i,2),\cdots,c(i,M)]$，所有任务在各个处理机上的执行时间向量构成了执行时间矩阵 $\boldsymbol{C}_{N\times M}$，$c(i,j)$ 表示 τ_i 在处理机 P_j 上的执行时间. 假

设任务之间相互独立,且可抢占.每一个处理机上的任务集按照单调速率算法进行调度[13].

定义 3. 一组实时周期任务的副版本集合:

$$B\Gamma = \{\beta_1, \beta_2, \beta_3, \cdots, \beta_N\},$$

$$\beta_i = (\boldsymbol{D}_i, T_i),\ i = 1, 2, \cdots, N.$$

每一个任务 τ_i 都对应一个副版本 β_i. β_i 的周期与 τ_i 相同,都为 T_i. β_i 在每一个处理机上的执行时间也是一个向量 $\boldsymbol{D}_i$,本模型中,假设主版本与副版本执行时间完全相同,即 $\boldsymbol{D}_i = [c(i,1), c(i,2), \cdots, c(i,M)]$. 主/副版本被调度到不同的处理机上执行,以保证当主版本执行失败的时候,副版本仍能在截止期前产生正确的结果.

定义 4. 本模型中的副版本 β_i 有两种执行方式:主动副版本(active backup-copy)和被动副版本(passive backup-copy). 对于每一个任务的分配,首先分配其主版本,再分配其副版本. 副版本的类型 $Status(\beta_i)$ 的判定由以下公式计算,即

$$Status(\beta_i) = \begin{cases} passive, & T_i - R_{ij} > D_{ik} \\ active, & T_i - R_{ij} \le D_{ik} \end{cases}, \quad (1)$$

$$P(\tau_i) = P_j \text{ 且 } P(\beta_i) = P_k$$

其中 R_{ij} 为主版本 τ_i 在处理机 P_j 上的最坏响应时间(Worst Case Response Time, WCRT). 可以看出,副版本方式的确定与文献[7]类似. 为方便统一描述,设 γ_i 为主版本或者副版本,即 $\gamma_i = \tau_i$ 或者 β_i. $P(\gamma_i)$ 表示 γ_i 所在的处理机.

为了集中讨论所关心的问题,同文献[7],还作出以下的一些假设:

(1) 任何时刻只存在一个处理机故障,即不可能等于或大于两个处理机同时出现故障.

(2) 处理机的故障是"失败-停止模式"(fail-stop mode),即处理机的工作状态为正常或者故障停止.

(3) 采用接受测试 AT(Acceptance Test)方法来进行处理机的故障检测[14]. 每一个任务的主版本会在其执行完毕的时候向对应的副版本发送消息,通知主版本执行完毕,如果副版本在指定的时间内仍未收到主版本发来的消息,则认为主版本所在的处理机出现故障. 本模型中,设置该指定时间为主版本的 WCRT,并设消息传播延迟开销可以忽略不计.

为了方便进行形式化描述,同文献[7],给出以下标记:

集合 $Primary(P_j)$ 和 $Backup(P_j)$ 分别代表分配到处理机 P_j 上的主版本任务和副版本任务;集合 $active(P_j)$ 包括分配到处理机 P_j 上的方式为 $active$ 的副版本任务;集合 $passiveRecover(P_j, P_f)$ 包含分配到 P_j 上的被动副版本,其对应的主版本被分配到 P_f 上;相对应的 $activeRecover(P_j, P_f)$ 包含分配到 P_j 上的主动方式副版本,其对应的主版本被分配到 P_f 上;最后 $Recover(P_j, P_f)$ 为以上两个集合的合集,即

$$\begin{cases} Primary(P_j) = \{\tau_i \mid P(\tau_i) = P_j\} \\ Backup(P_j) = \{\beta_i \mid P(\beta_i) = P_j\} \\ active(P_j) = \{\beta_i \mid \beta_i \in Backup(P_j),\ Status(\beta_i) = active\} \\ passiveRecover(P_j, P_f) = \{\beta_i \mid \beta_i \in Backup(P_j),\ Status(\beta_i) = passive,\ P(\tau_i) = P_f\} \\ activeRecover(P_j, P_f) = \{\beta_i \mid \beta_i \in Backup(P_j),\ Status(\beta_i) = active,\ P(\tau_i) = P_f\} \\ Recover(P_j, P_f) = passiveRecover(P_j, P_f) \bigcup activeRecover(P_j, P_f) \end{cases} \quad (2)$$

3 周期任务的可靠性模型

3.1 问题的提出

异构分布式系统的重要特点是每个处理机的性能和可靠性有很大的差别,因此在设计实时调度算法时不仅要考虑任务的可调度性和实时性等特点,还需要考虑系统的可靠性. 文献[2]定义系统的可靠性为系统中的任务集无故障运行的概率,并给出了可靠性代价的定义. 然而该可靠性模型仅适用于不可抢占的非周期任务集. 对于实时周期任务,如无故障干扰或者用户主动终止其执行,其执行时间会无限长(如人造卫星上的实时数据采集系统). 因此,基于非周期实时任务的可靠性模型并不适合于实时性的周期任务. 为了克服这个问题,我们在文献[9]中提出一个基于实时容错周期任务的可靠性模型. 虽然该模型能够在一定程度上度量系统的可靠性,但是该模型是基于系统无故障的情况下的. 换句话说,这个模型假设如果系统出现故障,那么在系统恢复故障前不会发生任何其它故障. 显然,这个模型在实际应用中的要求太苛刻了. 本文提出的可靠性模型则考虑到系统出现一个处理机故障后系统能恢复到正常情况下的概率.

3.2 基于周期任务集的可靠性调度模型

注意到周期任务集在每一个超周期 H(超周期为周期任务集中所有任务的周期的最大公倍数,即

$H=lcm\{T_i|\tau_i\in\Gamma\}$)内的执行是完全相同的；另一方面，处理机的故障符合 Poisson 过程，而 Poisson 过程又是平稳增量过程，所以研究周期任务集在一个超周期内的可靠性可以作为衡量系统可靠性的量度标准.

为了简化问题而不失一般性，在第 2 节提出的 3 个假设基础上，进一步提出以下 3 点假设：

(1)处理机在不处理任务时(即空载时)，如出现故障，将由备份处理机立即进行替换，因此不影响系统的可靠性[3].

(2)处理机在非空载出现故障后，同样由一定的恢复和替换机制(我们这里采用文献[7]提出的 FTRMFF-Replacing 算法)来保证系统在一段时间 MTTF(Mean Time To Failure)内能恢复到无故障时的状态，并保证系统恢复到无故障后处理机数目不变.

(3)文献[3]中已经提出并证明，在系统故障后，FTRMFF-Replacing 算法需要最多三倍于最大任务周期的时间来恢复系统的故障. 为了简化问题，假设 $T_{\max}$是任务集中周期最长任务的周期，并且任何一个处理机故障后，都能在 $3T_{\max}$内采用 FTRMFF-Replacing 算法来恢复到出错前的状态.

假设某一处理机的出错故障符合参数为 λ 的 Poisson 分布，那么 $Pr_n(t)=\dfrac{e^{-\lambda t}(t\times\lambda)^n}{n!}$给出了在时间段$[0,t]$内系统出现 n 个故障的概率. 假设 W 是表示处理机出现故障数目的随机变量. $NF_i(t)$表示处理机 P_i在时间段$[0,t]$内无故障的概率. 那么 $NF_i(t)$可表示为

$$NF_i(t)=Pr[W=0]=e^{-t\lambda_i} \tag{3}$$

假设 K 为表示系统的状态的随机变量且 K 属于集合$\{0,1,2,\cdots,M\}$. $K=0$ 表示没有处理故障，$K=k$ $(1\le k\le M)$表示处理机 P_k出现故障.

在不考虑单处理机失效的假设前提条件下，随机变量 K 的概率 $probability[K=k]$可以由以下公式计算：

$$probability[K=k]=\begin{cases}\prod\limits_{i=1}^{M}NF_i(\theta_i)=\prod\limits_{i=1}^{M}e^{-\lambda_i\theta_i}, & k=0\\ (1-NF_k(\theta_k))\prod\limits_{\substack{j=1,\\ j\neq k}}^{M}NF_j(\theta_j)=(1-e^{-\lambda_k\theta_k})\prod\limits_{\substack{j=1,\\ j\neq k}}^{M}e^{-\lambda_j\theta_j}, & k=i\ (1\le i\le M)\end{cases} \tag{4}$$

这里，$\theta_i=\sum\limits_{P(\gamma_j)\in\sigma_i}\left(C[j,i]\times\left(\dfrac{H}{T_j}\right)\right)$ 此时 $\sigma_i=Primary(P_i)\cup active(P_i)$，表示在一个超周期内，处理机 P_i上所有主版本和主动副版本任务的总执行时间.

但是由于我们这里是单处理机失效假设，即没有大于或者等于两个处理机同时失效. 设在单处理机假设条件下随即变量 K 的概率为 $Pr[K=k]$，显然有$\sum\limits_{k=0}^{M}Pr[K=k]=1$，根据全概率公式可计算有

$$Pr[K=k]=probability[K=k]\Big/\sum_{i=0}^{M}probability[K=i],\quad 0\le k\le M \tag{5}$$

设 $RC(\Omega,\Gamma,B\Gamma)$表示主版本集合 Γ 和副版本任务集合$B\Gamma$ 在处理机集合Ω 上的可靠性代价. RC_i表示系统在状态 $K=i$ 下的可靠性代价. 那么 $RC(\Omega,\Gamma,B\Gamma)$的期望值为

$$E(RC(\Omega,\Gamma,B\Gamma))=\sum_{i=0}^{M}(RC_i\times Pr[K=i]) \tag{6}$$

这里

$$RC_0=\sum_{i=1}^{M}\sum_{P(\gamma_j)\in\sigma_i}(C[j,i]\times H/T_j)\times\lambda_i$$

此时 $\sigma_i=Primary(P_i)\cup active(P_i)$ (7)

$$RC_i=\sum_{k=1,k\neq i}^{M}\sum_{P(\gamma_j)\in\omega_k}(C[j,k]\times(3\times T_{\max})/T_j)\times\lambda_k$$

此时 $\omega_k=Primary(P_k)\cup Recover(P_k,P_i)$ (8)

其中，RC_0表示系统中无处理机故障时，系统中所有的主版本和主动方式的副版本的可靠性代价，而 RC_i的可靠性由处理机 P_i失效后，在剩下的处理机中执行恢复任务，即 $\bigcup\limits_{\omega=1,\omega\neq i}^{M}\omega_k$的可靠性代价.

将式(4)，(5)，(7)，(8)代入到式(6)中得到

$$E(RC(\Omega,\Gamma,B\Gamma))=\sum_{i=0}^{M}(RC_i\times Pr[K=i])=RC_0\times Pr[K=0]+\sum_{i=1}^{M}[RC_i\times Pr[K=i]] \tag{9}$$

而系统的可靠性 $RL(\Omega,\Gamma,B\Gamma)$则是可靠性代价的函数，表示如下：

$$RL(\Omega,\Gamma,B\Gamma)=e^{-RC(\Omega,\Gamma,B\Gamma)} \tag{10}$$

由此可见，为了提高系统的可靠性，就必须减少系统的可靠性代价. 为了方便描述，我们给出以下定义.

定义 5.　定义任务版本 γ_i分配到处理机 P_j上的可靠性代价为 $rc_{ij}=\lambda_i\times C[i,j]/T_i$.

式(8)中等号右侧的第一项表示在系统无故障时的可靠性代价,为了减少 RC_0,从两个角度入手,首先可以把任务分配到可靠性代价较小的处理机上;其次,可以采取一定的任务分配算法,使任务的副版本尽量以被动方式执行.这一方面可以增加系统的可调度性,另一方面,可以减少系统的可靠性代价.而式(8)中等号右侧的第二项表示在系统出现一个处理机故障时的可靠性代价,显然对于 RC_i 而言,为了减少可靠性代价,同样需要把处理机失效后的任务分配到可靠性代价较小的处理机上.需要注意的是,对于 $\omega_k = Primary(P_k) \cup Recovery(P_k, P_i) = Primary(P_k) \cup passiveRecover(P_k, P_i) \cup activeRecover(P_k, P_i)$,其中的 $Primary(P_k)$ 和 $activeRecover(P_k, P_i)$ 都在分配 RC_0 时,保证其值尽量小.而对于 $passiveRecover(P_k, P_i)$ 则没有考虑.因此为了尽量减少系统故障时的 RC_i,在分配任务的被动副版本时,也需尽量把它分配到可靠性代价较小的处理机上.

4 基于异构分布式系统的实时容错调度算法

与实时调度算法一样,多处理机的实时容错调度算法也是 NP 完全问题[12].本文提出一个基于异构分布式系统的容错调度算法 IRDFTAHS.IRDFTAHS 在任务分配时,以减少系统的可靠性代价为目标,并使得更多的任务的副版本以被动方式执行,从而提高系统的可靠性.

与 RDFTAHS 相比,IRDFTAHS 算法不仅考虑系统无故障时的可靠性,还考虑了任一处理机故障后系统的可靠性.根据新的可靠性代价的模型,IRDFTAHS 在分配任务的副版本时,也尽量减少被动副版本的可靠性代价.IRDFTAHS 算法的启发原则如下:

(1)对于任务主版本的分配,尽量把任务的主版本分配到可靠性代价较小的处理机上.

(2)根据可靠性代价的分析,应使更多任务的副版本以被动方式执行,这样一方面可以减少可靠性代价,另一方面可以增加系统的可调度性.

(3)如果副版本以主动方式执行,则应当尽量把任务分配到可靠性代价较低的处理机上.

(4)即使副版本能以被动方式执行,也应当尽量把它分配到可靠性代价较低的处理机上,以减少在对应主版本失效后,副版本继续出错的概率.

为了方便算法的描述,下面给出可靠性成本向量的定义.

定义 6. 每一个任务的主/副版本定义一个可靠性成本向量 $\boldsymbol{V}_i = (v_{i1}, v_{i1}, \cdots, v_{iM})$,其中 $v_{ij} = (rc_{ij}, P_{ij})$,$rc_{ij} = (\lambda_{P_{ij}} \times c(i, P_{ij})/T_i)$ 表示任务 v_i 分配到处理机 P_{ij} 后的可靠性代价,$\boldsymbol{V}_i$ 中的元素按照可靠性代价的非递减的顺序进行排序,形式化描述排序结果为 $\forall i \in [1, N]: \forall j, k \in [1, M](j < k) \rightarrow (rc_{ij} \leq rc_{ik})$.

IRDFTAHS 算法对主版本和副版本进行分配时,都尝试把主副版本分配到可靠性代价较小的处理机上,同时保证任务的主/副版本分配到不同的处理机上.如果找到一对处理机可调度任务主/副版本,且副版本能以被动方式执行,则分配任务的主/副版本到该处理机对上,否则副版本只能以主动方式执行,以调度副版本到可靠性代价最低的处理机上.如果所有的处理机都无法调度任务的主/副版本,则返回调度失败.

算法 1. IRDFTAHS 算法.

输入:任务的主副版本集合 Γ,$B\Gamma$,处理机集合 Ω

输出:ret_value,任务分配结果的集合

1. 所有的周期任务的主/副版本按照周期的升序排列:$\tau_1, \beta_1, \tau_2, \beta_2, \cdots, \tau_N, \beta_N$.任务版本的优先级为周期的倒数,并假设主版本的优先级大于对应副版本的优先级.
2. for $i \leftarrow 1, 2, \cdots, N$ do
   /* 为 N 个任务的主/副版本分配处理机 */
   $found\_active \leftarrow$ FALSE,$tmp\_rc = 0$,生成任务 i 的可靠性成本向量 $\boldsymbol{V}_i$;
3. for $k \leftarrow 1, 2, \cdots, M$ do
4. for $s \leftarrow 1, 2, \cdots, M$, $s \neq k$ do
   /* 主/副版本分配到不同的处理机 */
5. 对任务主副版本都按照向量 $\boldsymbol{V}_i$ 的指定处理机顺序分配,即分配任务 τ_i 到处理机 $V_i.P_{ik}$ 上,分配 β_i 到处理机 $V_i.P_{is}$ 上,并由定义 4 判断 β_i 的方式.如果主/副版本都可调度且 $status(\beta_i) = passive$,则分配 τ_i, β_i 到处理机 $V_i.P_{ik}, V_i.P_{is}$ 上,转步 2;
6. 如果主/副版本都可调度且 $status(\beta_i) = active$ 且 $found\_active =$ FALSE,则设置 $found\_active \leftarrow$ TRUE,记下处理机 $V_i.P_{ik}$ 和 $V_i.P_{is}$ 计算副版本分配到 P_s 后的可靠性代价 tmp_rc;
7. 如果主/副版本都可调度且 $status(\beta_i) = active$ 且 $found\_active =$ TRUE 且副版本分配到 P_s 后的可靠性代价 $< tmp\_rc$;记下处理机 $V_i.P_{ik}$ 和 $V_i.P_{is}$,$tmp\_rc \leftarrow$ 副版本分配到 P_s 后的可靠性代价;

8. end for
9. end for
10. if (*found_active* =TRUE)
11. 分别设置 $P(\tau_i)$和 $P(\beta_i)$为步 6、步 7 记录下的处理机号，设 $status(\beta_i)\leftarrow active$；
12. else
13. $ret_value\leftarrow$FAIL；return *ret_value*；
14. end for
15. $ret_value\leftarrow$SUCCESS；return *ret_value*.

5　任务集的可调度性分析

5.1　任务分配的三种情况

为了判断任务版本在处理机上的可调度性，必须保证实时任务的主/副版本指定到不同的处理机上运行. 同时，对每一个任务的主/副版本既要保证在没有处理机故障的时候的可调度性，也要保证即使任何一个处理机故障时的可调度性.

对于任务的分配，存在 3 种情况：

(1) 当分配一个任务的主版本 τ_i给某一个处理机 P_j时，需要保证两种情况下的 τ_i的可调度性：

①当无处理机故障时，τ_i将要和已经分配给 P_j的主版本和方式为 *active* 的副版本一起调度，即判断任务集 $\sigma=Primary(P_j)\cup active\ (P_j)\cup\{\tau_i\}$的调度性；

②当某一处理机 $P_f(P_j\neq P_f)$出现故障时，对于所有的 $f=1,2,\cdots,M(f\neq j)$，τ_i和已经分配给 P_j的主版本和分配给 P_j的所有的副版本其对应的主版本在 P_f的任务集一起调度，即判断任务集$\sigma=Primary(P_j)\cup Recover(P_j,P_f)\cup\{\tau_i\}$的可调度性；

(2) 方式为 *active* 的副版本的分配. 当分配一个方式为 *active* 的副版本 β_i到处理机 P_j时，假设其对应的主版本 τ_i已经指定到 P_f，同样需要检测以下两种情况：

① 无处理机故障时，β_i将要和已经分配给 P_j的主版本和 *active* 的副版本一起调度，即判断任务集 $\sigma=Primary(P_j)\cup active\ (P_j)\cup\{\beta_i\}$的可调度性；

② 当处理机 $P_f(P_j\neq P_f)$故障时，β_i和已经分配给 P_j的主版本和分配给 P_j的所有副版本其对应的主版本在 P_f的任务集一起调度，即判断任务集 $\sigma=Primary(P_j)\cup Recover(P_j,P_f)\cup\{\beta_i\}$的可调度性；

(3) 方式为 *passive* 的副版本的分配. 当分配一个方式为 *passive* 的副版本 β_i到处理机 P_j时，假设其对于主版本 τ_i已经指定到 P_f，仅需要判断一种情况：

①因为 *passive* 副版本任务仅在其对应主版本任务所在的处理机失效时才启动执行，仅需检测 $\sigma=Primary(P_j)\cup Recover(P_j,P_f)\cup\{\beta_i\}$的可调度性.

5.2　容错调度性分析

文献[15]提出并证明了完成时间测试定理(Completion Time Test，CTT). 该定理表明，对于按照固定优先级调度的实时周期任务集，如果任务集中的每一个任务的最坏响应时间都小于它们各自对应的截止期. 那么该实时周期任务集是可调度的.

由于标准的 CTT 定理仅仅适用于标准的固定优先级调度算法，不符合本文中的容错调度模型. 因此我们提出并证明符合本文模型的容错 CTT 定理(Fault-Tolerant CTT，FTCTT).

为方便描述，首先做出以下一些定义：一般性的任务集合 σ_k，σ_k可能包含任务的主版本，也可能包含主动或被动副版本，σ_k中的任务被调度到处理机 P_k上. σ_k中优先级最低的任务版本为 $\gamma_{\max c}$.

定理 1.　如果 σ_k是在没有处理机故障的情况下处理机 P_k上的任务集，$\gamma_{\max c}$的最大响应时间 $R_{FR}(\gamma_{\max c})$由式(11)计算.

$$R_{FR}(\gamma_{\max c})=\sum_{\tau_i\in\sigma_k}\lceil R_{FR}(\gamma_{\max c})/T_i\rceil\times C[i,k]+\sum_{\beta_i\in\sigma_k}\lceil R_{FR}(\gamma_{\max c})/T_i\rceil\times D[i,k]\quad(11)$$

如果 $R_{FR}(\gamma_{\max c})$满足 $R_{FR}(\gamma_{\max c})\leq T_{\max c}$，则任务集 σ_k可调度；否则 σ_k不可调度. 本定理称为 FR-FTCTT(Fault-Free Fault-Tolerant Completion Time Test).

证明.　由于任务的分配是按照优先级由高到低进行的，所以 $\gamma_{\max c}$是当前分配的任务，且 $\sigma_k-\{\gamma_{\max c}\}$已经是可调度的，只需保证 $\gamma_{\max c}$的 WCRT 小于其截止期即可. 根据 CTT 定理，$\gamma_{\max c}$而是 σ_{FR}中响应时间最长的任务且其 WCRT 应该等于在时间 $R_{FR}(\gamma_{\max c})$内 σ_k所需要的计算量. 在无故障时，σ_k上的任务仅有主版本任务和主动副版本任务，即 $\sigma_k=Primary(P_k)\cup active\ (P_k)$：

(1) 主版本任务在每个周期 T_i内都执行一次，故在时间 t 内共需执行$\lceil t/T_i\rceil$次，当被分配到处理机 P_k上后，所需的计算时间量为$\lceil t/T_i\rceil\times C[i,k]$，所以主版本任务对 WCRT 的贡献由式(11)等号后的第一部分组成；

(2) *active* 副版本的情况与主版本任务相同，在每个周期 T_i内都执行一次，每次执行时间同样为 $C[i,k]$，所以这部分时间为式(11)等号后的第二

部分；

综合(1)，(2)，可由式(11)计算得到 $\gamma_{\max c}$ 的 WCRT. 对于 $\gamma_{\max c}$ 的截止期，无论 $\gamma_{\max c}$ 是主版本或者 *active* 副版本，$\gamma_{\max c}$ 的截止期应该为 $\gamma_{\max c}$ 的周期，即 $T_{\max c}$，显然如果 $R_{FR}(\gamma_{\max c}) \leq T_{\max c}$，则任务集 σ_{FR} 可调度. 证毕.

定理 2. 如果 σ_k 是在某一个处理机故障的情况下处理机 P_k 上的任务集. $\gamma_{\max c}$ 的最大响应时间 $R_{RO}(\gamma_{\max c})$ 由式(12)计算，$\gamma_{\max c}$ 的容错截止期 $FTDeadLine(\gamma_{\max c})$ 由式(13)计算.

$$R_{FO}(\gamma_{\max c}) = \sum_{\tau_i \in \sigma_k} \lceil R_{FO}(\gamma_{\max c})/T_i \rceil \times C[i,k] + \sum_{\beta_i \in \sigma_k} \Phi(i, R_{FO}(\gamma_{\max c})) \times D[i,k] \quad (12)$$

此时，

$$\Phi(i,t) = \begin{cases} \lceil t/T_i \rceil, & status(\beta_i) = active \\ 1, & status(\beta_i) = passive \text{ 且 } t \leq B_{(i-1)} \\ \lceil (t - B_{(i-1)})/T_i \rceil + 1, & status(\beta_i) = passive \text{ 且 } t > B_{(i-1)} \end{cases}$$

$$FTDeadLine(\gamma_{\max c}) = \begin{cases} T_{\max c}, & \gamma_{\max c} = Priamry \text{ 或 } \gamma_{\max c} = active \\ B_{(\max c-1)}, & \gamma_{\max c} = passive \end{cases} \quad (13)$$

如果 $R_{FO}(\gamma_{\max c})$ 满足 $R_{FO}(\gamma_{\max c}) \leq FTDeadLine(\gamma_{\max c})$，则任务集 σ_k 可调度；否则 σ_k 不可调度. 本定理称为 FO-FTCTT(Fault-Occurance Fault-tolerant Completion Time Test).

证明. 定理 2 的证明同定理 1，$\gamma_{\max c}$ 是 σ_{FO} 中响应时间最长的任务且其 WCRT 应该等于在时间 $R_{FR}(\gamma_{\max c})$ 内 σ_k 所需要的计算量. 在出现处理机故障时，σ_k 上的任务仅有主版本任务和 3 种方式的副版本任务：

(1) 主版本任务在每个周期 T_i 内都执行一次，故在时间 t 内共需执行 $\lceil t/T_i \rceil$ 次，当主版本任务在处理机 P_k 上，所需的计算时间量为 $\lceil t/T_i \rceil \times C[i,k]$，这部分时间为式(12)等号后的第一部分；

(2) 对于 *active* 副版本，与主版本任务相同，在每个周期 T_i 内都执行一次，故此时 $\Phi(i,t) = \lceil t/T_i \rceil$，即在时间段 $[0,t]$ 内，所需的计算量为 $\lceil t/T_i \rceil \times C[i,k]$；

(3) 对于 *passive* 副版本. *passive* 副版本的第一个请求必须在 β_i 的恢复时间 B_{i-1} 内完成，而 β_i 后续请求的周期都为 T_i，因此如果 $t \leq B_{i-1}$，则仅有 β_i 的第一个请求在时间 $[0,t]$ 被执行，即 $\Phi(i,t) = 1$，否则在时间 $[0, B_{i-1}]$ 内 β_i 请求执行一次，在 $[B_{i-1}, t]$ 内请求执行 $\lceil (t - B_{i-1})/T_i \rceil$，此时 $\Phi(i,t) = \lceil (t - B_{i-1})/T_i \rceil + 1$.

综合(1)～(3)，可由式(12)计算得到 $\gamma_{\max c}$ 的 WCRT. 对于 $\gamma_{\max c}$ 的容错截止期，$\gamma_{\max c}$ 为主版本或 *active* 副版本时，$FTDeadLine(\gamma_{\max c})$ 为 $\gamma_{\max c}$ 的周期 $T_{\max c}$；当 $\gamma_{\max c}$ 为 *passive* 副版本时，由于 $T_{\max c} > B_{\max c-1}$，只需保证 $\gamma_{\max c}$ 的第一个任务的 WCRT 小于 $B_{\max c}$，则 $\gamma_{\max c}$ 在以后的周期中均可调度，故此时 $FTDeadLine(\gamma_{\max c})$ 为 $B_{\max c-1}$. 显然如果 $R_{FO}(\gamma_{\max c}) \leq FTDeadLine(\gamma_{\max c})$，则任务集 σ_{FO} 可调度. 证毕.

6 实验与结果分析

本节将 IRDFTAHS 算法和文献[9]中提出的 RDFTAHS 算法进行性能比较，首先从可调度性(schedulability)和可靠性(reliability)两个方面进行比较；进一步，第三个实验对两个算法的综合性能进行比较，采用一个综合量度标准——Performability，该量度综合考虑算法的可靠性和可调度性. 模拟程序用 VC++6.0 完成，硬件平台为带有 512MB RAM 的 Pendtium4 1.7GB.

6.1 算法的可靠性比较

模拟实验的步骤如下：

1. 每一次任务分配测试中，随机生成的实时任务集数目为 L，L 分别取 200，400，600，800，1000；

2. 为了方便计算可靠性代价，在生成的实时任务集中，周期任务 τ_i 和 β_i 的周期 T_i 在集合{1，2，4，5，10，20，25，50，100，125，250，500}中取值，这样保证了生成的任务集的超周期为 500；定义任务集中任务的最大负载为 $\alpha = \max_{i=1,2,\cdots,N;\ j=1,2,\cdots,M} C[i,j]/T_i$. 每一个任务的执行时间服从均匀分布 $0 \leq C[i,j] \leq \alpha T_i$. 为方便起见，设定 τ_i 执行时间与 β_i 相同，即 $D[i,j] = C[i,j]$，α 分别设置为 0.2，0.5，0.8；

3. 确定处理机的个数 *Processor_Num*，每次实验中处理机的个数对应于任务的个数，每 200 个任务有 15 个处理机，即 $Processor\_Num = (L \times 15)/200$；

4. 性能量度标准为由式(9)计算的可靠性代价 *ReliabilityCost*，对于每一个 L 和 α，对每一次生成的任务集分别用 IRDFTAHS 算法和 RDFTAHS 算法进行调度分配，并计算任务分配后的可靠性代价，实验重复进行 30 次，并取 30 次实验的平均值为最终结果，分别设为 M 和 N. 表 1 总结了仿真实验的各个参数. 图 1 给出了最终的实验结果.

表 1　仿真实验的参数说明

参数	说明	取值范围
FR	处理机的失效率	$-(0.95,0.96,\cdots,1.05)\times10^{-6}$
T	任务的周期	{1,2,4,5,10,20,25,50,100,125,250,500}
α	任务的最大负载	[0.2,0.5,0.8]
$Processor\_Num$	处理机的个数	[15,30,45,60,75]
L	任务的个数	[200,400,600,800,1000]

从图 1 中可以看出，对于相同的 α 值，随着任务个数的增加，所需的可靠性代价也逐渐增加，这是因为随着任务数目增加，需要更多的处理机时间来调度任务，所以可靠性代价也随之增加. 更重要的是，对于相同的 α 和相同的任务个数，IRDFTAHS 算法的可靠性代价比 RDFTAHS 要小，说明 IRDFTAHS 算法比 RDFTAHS 更加可靠. 当任务个数 L 比较小时(200,400)，IRDFTAHS 与 RDFTAHS 的差别并不大，随着任务个数 L 的增加，IRDFTAHS 的可靠性代价相对 RDFTAHS 越来越小. 这是因为，当任务个数还不大的时，较多任务的副版本以被动方式执行，这样，可靠性代价的差别主要体现在主版本的分配上，而当任务个数逐渐增大的情况下，较多的任务以主动方式执行，这样可靠性除了体现在主版本的分配上，还体现在主动副版本的分配上. 同理，对于相同的任务个数，随着任务负载 α 的增加，IRDFTAHS 显得更加可靠.

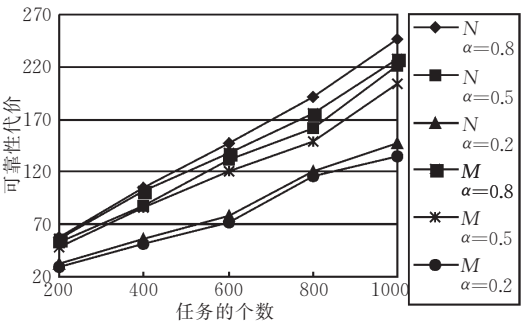


图 1　算法 IRDFTAHS 与 RDFTAHS 可靠性代价的比较

6.2　算法的可调度性比较

IRDFTAHS 和 RDFTAHS 算法都是基于异构分布式系统模型下的实时容错调度算法，两个算法都以任务集和处理机集合为输入参数来判断调度是否成功，由此判断处理机集合是否满足调度的需要. 因此对一个实时任务的集合，所需要最小处理机 MNP(Minimal Number of Processors)的个数可以作为判断调度算法的可调度性的标准. 这里采用文献[10]提出的求解最小处理机数目的算法 FMNP (Find Minimal Number of Processors)来比较两个算法的可调度性. 算法描述中用 $Func\_RTFT$ 参数代表 FMNP 算法中使用的两种不同的实时容错调度算法.

算法 2.　FMNP 算法.

输入：任务的主副版本集合 Γ，$B\Gamma$，调度算法 $Func\_RTFT$

输出：算法所需要的最小处理机个数 $MinP$

1. $Lower\leftarrow1$，$Upper\leftarrow N$，$MinP\leftarrow\lfloor(Lower+Upper)/2\rfloor$
2. while $Lower<>MinP$ do
3. 　$ret\_val\leftarrow Func\_RTFT(\Gamma,B\Gamma,\Omega)$;
4. 　if $ret_val=$SUCCESS then $Upper\leftarrow MinP$;
5. 　else $Lower\leftarrow MinP$;
6. 　$MinP\leftarrow\lfloor(Lower+Upper)/2\rfloor$;
7. end while
8. return $(MinP+1)$;

图 2 表示了使用 IRDFTAHS 和 RDFTAHS 算法进行调度所需的 MNP 的示意图，从图中可以看出随着任务个数的增加，调度所需的 MNP 也随着增加，这是因为随着任务个数的增加，需要更多的处理机来调度这些任务集. 另一方面可以看出，针对相同的负载 α 和任务个数 L，IRDFTAHS 和 RDFTAHS 算法所需的 MNP 基本相同，或者多一到两个处理机，因此可以看出，算法 RDFTAHS 在可调度性方面略优于 IRDFTAHS. 并且两者间的差别并不大，且与任务负载 α 和任务个数的关系也不大.

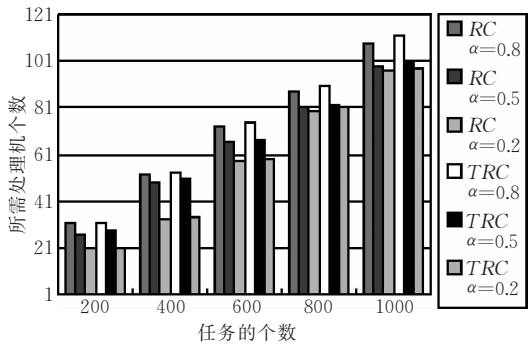


图 2　算法 IRDFTAHS 与 RDFTAHS 所需最小处理机个数的比较

6.3　算法的综合性能比较

为了对两个算法的综合性能进行比较，本文采用一种综合量度标准——综合性能(Performabili-

ty),定义为

$$Performability = MinP \times ReliabilityCost \tag{14}$$

其中 $MinP$ 为对给定任务集进行调度所需的最小的处理机个数,$ReliabilityCost$ 为用最小处理机个数进行调度时的可靠性代价.显而易见,对于给定任务集合进行调度,算法的值越小,综合性能越好.

实验在 6.2 节的基础上进行,对于任务集中的任务个数 L 和 α 用 6.2 节中得出的 $MinP$ 个处理机进行调度,每一个 L 和 α 值进行 30 次调度,计算每次调度的 Performability,取 30 次的平均值,图 3 描述了调度的最后结果.

从图 3 中可以看出,对于不同的 L 和 α 值,算法 IRDFTAHS 比 RDFTAHS 有更佳的综合性能.并且,α 和 L 值越大,IRDFTAHS 算法的优越性越明显.

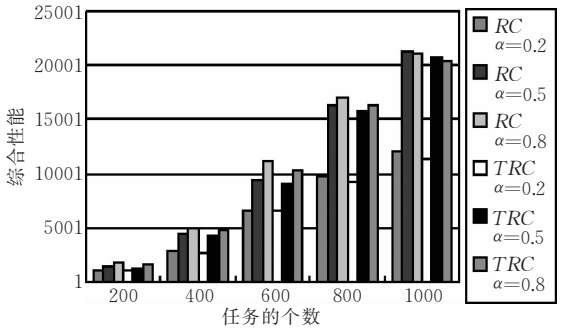


图 3 算法 IRDFTAHS 与 RDFTAHS 的综合性能比较

7 结束语

本文的主要贡献有:(1)提出一种改进的基于抢占性周期任务集的可靠性的调度模型,引入可靠性代价的概念;(2)基于该模型,提出一个基于异构分布式系统的实时容错调度算法 IRDFTAHS;(3)将 IRDFTAHS 和已有的调度算法在可调度性、可靠性和综合性能 Performability 三个方面进行实验比较,实验结果表明 IRDFTAHS 比现有算法具有更好的总体性能.

下一步的工作还包括:(1)提出更加高效的任务分配算法,新的高效算法比本文提出的算法具有更高的性能;(2)研究一种混合的可靠性调度模型,该模型能适用于周期任务和非周期任务;(3)考虑周期任务之间的通信和优先级约束,并考虑任务间通信的可靠性.

参 考 文 献

[1] Ranaweera S, Agrawal D P. Scheduling of periodic time critical applications for pipelined execution on heterogeneous systems//Proceedings of the 2001 International Conference on Parallel Processing Valencia. Spain, 2001: 131-138

[2] Srinivasan S, Jha N K. Safety and reliability driven task allocation in distributed systems. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(3): 238-251

[3] Dogan A, Ozguner F. Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing//Proceedings of the 29th International Conference on Parallel Processing. Toronto, Canada, 2000: 307-314

[4] Yang C H, Deconinck G, Gui W H. Fault-tolerant scheduling for real-time embedded control systems. Journal of Computer Science and Technology, 2004, 19(2): 191-202

[5] Yang F M, Luo W, Pang L P. An efficient real-time fault-tolerant scheduling algorithm based on multiprocessor systems. Wuhan University Journal of Natural Sciences, 2007, 6(11): 219-223

[6] Al-Omari R, Somani A K, Manimaran G. An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems. Journal of Parallel and Distributed Computing, 2005, 65(5): 595-608

[7] Bertossi A A, Mancini L V, Rossini F. Fault-tolerant rate-monotonic first-fit scheduling in hard-real-time systems. IEEE Transactions on Parallel and Distributed Systems, 1999, 10(9): 934-945

[8] Qin Xiao, Han Zong-Feng, Pang Li-Ping. Towards real-time scheduling with fault-tolerance in heterogeneous distributed systems. Chinese Journal of Computers, 2002, 25(1): 49-56 (in Chinese)
(秦啸,韩宗芬,庞丽萍.分布式实时系统的容错调度算法.计算机学报, 2002, 25(1): 49-56)

[9] Luo W, Yang F M, Pang L P, Qin X. Fault-tolerant scheduling based on periodic tasks for heterogeneous systems//Proceedings of the 3rd International Conference on Autonomic and Trusted Computing. Wuhan, China, 2006: 571-580

[10] Qin X, Han Z F, Jin H, Pang L P, Li S L. Real-time fault-tolerant scheduling in heterogeneous distributed systems//Proceedings of the International Workshop on Cluster Computing-Technologies, Environments, and Applications (CC-TEA). Las Vegas, USA, 2000: 421-427

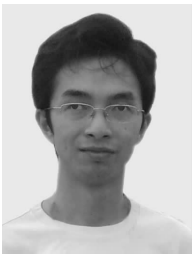
[11] Qin X, Jiang H, Swanson D R. An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems//Proceedings of the International Conference on Parallel Processing. British Columbia, Canada, 2002: 360-368

[12] Luo Wei, Yang Fu-Min, Pang Li-Ping, Li Jun. A real-time fault-tolerant scheduling algorithm for distributed systems based on deferred active backup-cop. Journal of Computer Research and Development, 2007, 44(3): 521-528(in Chinese)
(罗威,阳富民,庞丽萍,李俊. 基于延迟主动副版本的分布式实时容错调度算法. 计算机研究与发展, 2007, 44(3): 521-528)

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

[14] Johnson B W. Design and Analysis of Fault Tolerant Digital Systems. New York: Addison Wesley, 1989

[15] Klein M H, Lehoczky J P, Rajkumar R. Rate-monotonic analysis for real-time industrial computing. Computer, 1994, 27(1): 24-33



LUO Wei, born in 1980, Ph. D. candidate. His current research interests include real-time systems, fault-tolerant computing and embedded system.

YANG Fu-Min, born in 1966, M. S., professor. His current research interests include real-time and embedded operation system.

PANG Li-Ping, born in 1944, professor and Ph. D. supervisor. Her current research interests include parallel and distributed systems, real-time systems

TU Gang, born in 1976, Ph. D., associate professor. His current research interests include real-time systems and embedded system and real-time scheduling algorithms.

Background

The work reported in this paper was supported in part by the National Natural Science Foundation of China under grant No. 60603032.

Fault-tolerance is an inherent requirement in real-time distributed systems. In this area, Primary Backup scheme plays an important role. Different from homogeneous distributed systems, the most essential characteristic of heterogeneous distributed systems is their great varieties of the computing power and reliability of different processors. Therefore, when designing a real-time scheduling algorithm, we should not only consider the real-time property and schedulability of tasks but also take the reliability of the scheduling results into accounts.

Reliability has been a main concern of computer systems research community for many years. Conventionally, the reliability of a system is defined as the probability that the system functions are properly and continuously without any interruption. With the emergence of critical business applications and safety critical systems, the traditional definition of reliability needs to be extended to incorporate fault tolerance. While achieving high reliability, it is critical to ensure that real-time tasks can be completed before their deadlines even in presence of failures. Thus, it is imperative for a reliable system to take fault-tolerant actions after failures are detected so that the reliable system can continue its execution without any interrupt due to the failures. However, to the best of our knowledge, most existing reliability models constructed for real-time systems has not addressed the issue of fault tolerance.

To bridge this gap, this paper proposes a novel reliability model based on preemptive periodic tasks. Compared with existing algorithms in literature, the proposed reliability model can be one-processor-failed fault-tolerant. Moreover, a real-time fault-tolerant scheduling algorithm based heterogeneous distributed systems, namely IRDFTAHS, is presented. IRDFTAHS try to assign tasks copies in a way to improve reliability of system. IRDFTAHS considers both active and passive backup copy, which make the proposed algorithm more flexible than existing algorithms. Future studies in this arena are three-fold. First, the authors are going to develop more efficient scheduling algorithms aiming at boosting system's reliability. Second, they will further extend the reliability model to deal with hybrid real-time applications containing the three different types of real-time tasks. Third, they intend to investigate a more complex version of scheduling algorithm, in which the precedence constrains among tasks and aperiodic tasks scheduling are incorporated.