Journal of System Simulation

Volume 29 | Issue 5 Article 27

6-3-2020

Real-Time Scheduling Algorithm of Dynamic with Fault-Tolerant in Heterogeneous Distributed Systems

Chongjie Dong
1. Dongguan Polytechnic, Dongguan 523808, China; ;

Yuqiang Chen
1. Dongguan Polytechnic, Dongguan 523808, China; ;2. Guangdong University of Technology, Guangzhou 510006, China;

Follow this and additional works at: https://dc-china-simulation.researchcommons.org/journal

 Part of the Artificial Intelligence and Robotics Commons, Computer Engineering Commons, Numerical Analysis and Scientific Computing Commons, Operations Research, Systems Engineering and Industrial Engineering Commons, and the Systems Science Commons

This Paper is brought to you for free and open access by Journal of System Simulation. It has been accepted for inclusion in Journal of System Simulation by an authorized editor of Journal of System Simulation.

Real-Time Scheduling Algorithm of Dynamic with Fault-Tolerant in Heterogeneous Distributed Systems

Abstract

Abstract: The traditional heterogeneous distributed real-time scheduling algorithm didn't give consideration to the dynamic behavior of tasks. An aperiodic non-preemptible and heterogeneous distributed dynamic fault-tolerant model and presenting two fault-tolerant scheduling algorithms was proposed based on this model: DRFSA algorithm and DSFSA algorithm. *DRFSA(Dynamic and Reliability-driven of hybrid with Fault-tolerant Scheduling Algorithm)algorithm is to improve the system reliability as the scheduling objective through the reasonable scheduling. DSFSA (Dynamic and Schedulability-driven of hybrid with Fault-tolerant Scheduling Algorithm) algorithm is to improve schedulability as the scheduling objective and tries to enhance the schedulability through less task execution time. Both algorithms can dispatch the dynamic real-time tasks in the heterogeneous system andsimultaneously can respond to the task demands as far as possible.*

Keywords

primary copy/backup copy, dynamic real-time task, heterogeneously distributed, reliability, schedulability

Recommended Citation

Dong Chongjie, Chen Yuqiang. Real-Time Scheduling Algorithm of Dynamic with Fault-Tolerant in Heterogeneous Distributed Systems[J]. Journal of System Simulation, 2017, 29(5): 1132-1140.

This paper is available in Journal of System Simulation: https://dc-china-simulation.researchcommons.org/journal/vol29/iss5/27

第 29 卷第 5 期 系统仿真学报© Vol. 29 No. 5
2017 年 5 月 Journal of System Simulation May, 2017

异构分布式系统动态实时容错调度启发式算法

董崇杰[1]，陈俞强[1,2]
(1. 东莞职业技术学院，广东 东莞 523808；2. 广东工业大学，广东 广州 510006)

摘要：传统的异构分布式实时调度算法基本没有考虑任务的动态特性。提出一种非周期不可抢占式异构分布式的动态容错模型，在该模型上基于不同调度需求给出两种不同容错调度算法：DRFSA(Dynamic and Reliability-driven of hybrid with Fault-tolerant Scheduling Algorithm)算法与DSFSA(Dynamic and Schedulability-driven of hybrid with Fault-tolerant Scheduling Algorithm)算法。*DRFSA 算法以提高可靠性代价为调度目标，通过合理调度提高系统可靠性。DSFSA 算法以可调度性为调度目标，通过减少任务执行时间来增加系统可调度性。算法能够在异构系统中调度动态的实时任务，且能够尽可能响应任务需求。*

关键词：基/副版本；动态实时任务；异构分布式；可靠性；可调度性
中图分类号：TP391.9 文献标识码：A 文章编号：1004-731X (2017) 05-1132-09
DOI: 10.16182/j.issn1004731x.joss.201705027

Real-Time Scheduling Algorithm of Dynamic with Fault-Tolerant in Heterogeneous Distributed Systems

Dong Chongjie[1], *Chen Yuqiang*[1,2]
(1. Dongguan Polytechnic, Dongguan 523808, China; 2. Guangdong University of Technology, Guangzhou 510006, China)

Abstract: The traditional heterogeneous distributed real-time scheduling algorithm didn't give consideration to the dynamic behavior of tasks. An aperiodic non-preemptible and heterogeneous distributed dynamic fault-tolerant model and presenting two fault-tolerant scheduling algorithms was proposed based on this model: DRFSA algorithm and DSFSA algorithm. *DRFSA(Dynamic and Reliability-driven of hybrid with Fault-tolerant Scheduling Algorithm)algorithm is to improve the system reliability as the scheduling objective through the reasonable scheduling. DSFSA (Dynamic and Schedulability-driven of hybrid with Fault-tolerant Scheduling Algorithm) algorithm is to improve schedulability as the scheduling objective and tries to enhance the schedulability through less task execution time. Both algorithms can dispatch the dynamic real-time tasks in the heterogeneous system andsimultaneously can respond to the task demands as far as possible.*

Keywords: primary copy/backup copy; dynamic real-time task; heterogeneously distributed; reliability; schedulability

引言

实时系统的一个重要特性是实时任务须在指定时间内完成，无论系统出现任何错误，实时任务仍可以得出可接受的结果[1]。为了确保任务在系统出故障后仍然能够在期限内完成，容错能力成为系统的必要特性。基于基/副版本调度方法是实现实时分布式系统容错的主要手段之一。针对实时分布式容错系统的特点，根据不同性能指标，学者们分别基于同构和异构分布式系统中的实时容错调度

收稿日期：2016-04-29 修回日期：2016-08-23；
基金项目：国家自然科学基金(61106019)，广东省高等学校优秀青年教师培养计划(YQ2015232)，广东省科技计划项目(2014A010103002)，东莞市社会科技发展项目(2013108101045), 2013108101046)；
作者简介：董崇杰(1982-)，男，山东菏泽，硕士，副教授，研究方向为数据库技术。

Published by Journal of System Simulation, 2017

Journal of System Simulation, Vol. 29 [2017], Iss. 5, Art. 27

算法进行研究和设计，其目的是旨在通过高质量的调度算法来提高实时分布式系统可靠性和可调度性，以及任务接收率等不同的性能。目前，许多学者对容错技术进行了相关研究[2-6]。文献[2]提出一种分布式投票方法，文献[3]给出一种带反转恢复技术的容错方法，文献[4-6]是目前比较流行并广泛应用于分布式系统的基于基/副版本备份技术(Primary/Backup copy，P/B)的容错调度算法，备份任务通过备份处理机执行达到系统容错的目标。文献[7]研究了动态实时调度算法与速率单调算法。文献[8]讨论带固定优先级实时调度算法，这些算法均没有考虑系统的容错问题。文献[9-10]只讨论了同构环境下的周期和非周期任务周期模型。

文献[11-12]针对非周期任务的异构分布式模型及算法在可靠性代价方面进行分析。文献[13]针对基/副版本模型提出了负载均衡容错调度算法，但是没有分析周期性调度任务。文献[14]研究一种基于周期性实时调度模型及算法在异构分布式系统的应用，但是没有研究分布式系统中非常重要的非周期任务。文献[15]提出一种基于静态模型混合式异构实时容错调度模型。目前，具有良好的容错性、可靠性等重要指标的实时系统广泛应用于各个行业，但是，由于实时系统在实际应用中存在大量动态实时任务，因此还需要考虑任务的动态特性。

本文在结合上述文献部分思想的基础上，给出了一个动态异构分布式实时系统模型。并在此基础上从不同角度给出了不同调度算法。由于容错任务调度问题是一个 NP(Non-Deterministic Polynomial)难题[16-19]，所以，本文提出的算法均为启发式调度算法。最后，通过模拟实验分析对比了不同调度算法的性能，得出了研究结论。

1 模型设计

异构分布式系统的动态容错调度模型分为系统模型和调度模型[12]。系统模型给出异构分布式系统的实时任务模型和处理机模型，而调度模型则给出描述任务的动态调度方式。

1.1 系统模型

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，由以下定义进行描述。

定义 1. 异构分布式系统描述为一个处理机集合 $\Phi=\{P_1, P_2,\ldots, P_k\}$，处理机失效向量 $\boldsymbol{R}=\{\lambda_1, \lambda_2,\ldots, \lambda_k\}$及处理机状态集 $\Omega=\{\omega_1, \omega_2,\ldots,\omega_k\}$且 ω=FR|AP，其中，FR 表示处理机处于空闲状态，而 AP 表示处理机正在处理实时任务。假设各处理机之间出错过程相互独立且符合泊松过程。其中，λ_i 为处理机 P_i 的失效率，ω_i 为处理机 P_i 当前所处的状态，k 为系统中处理机的个数[9,15]。

定义 2. 实时任务集合 $T=\{t_1, t_2, \ldots, t_n\}$。在容错调度过程中保证实时任务集合在异构系统中某一处理机失效时，仍能够在规定的截止期内完成。假设实时任务集合只有一个副版本,且只有在基版本所在的节点机出现故障时副版本才投入运行。$t_i=\{d, t_p, t_b\}$，其中，d 表示任务截止时间；t_p 与 t_b 表示调度任务的基版本和副版本，其中，$t_p=t_b=\{a, s, \boldsymbol{C}, \text{pcs}\}$，a, s 分别表示实时任务的基或副版本到达系统时间和在处理机上开始执行时间且知 $s\geqslant a$；$\boldsymbol{C}$ 是一个向量集合表示调度到各处理机上的执行时间；pcs 表示基或副版本所调度到的处理机。

定义 3. 在异构分布式系统中，每个任务 ti 在不同的处理机上有不同的执行时间，所以为每个任务 t_i 定义一个时间向量 $\boldsymbol{C}$。$\boldsymbol{C}_i=[c(\text{i},1), c(\text{i},2),\ldots, c(\text{i}, \text{m})]$，$C_{ij}$ 表示任务 t_i 在处理机 j 上执行完所需时间。为了使问题简化我们考虑的截止时间 d 都是相对于任务的到达系统时间，所以任务绝对截止时间 $D=d+a$；同时假设考虑的任务基/副版本完全相同，即任务的基/副版本在同一处理机上的执行时间相同。

定义 4. 在异构分布式系统中为了描述方便，定义 RQ(Refuse Queue)表示被拒绝的任务集合。被拒绝的实时任务形式化描述为：

$$\text{RQ}=\{t_i\in\text{SQ}\cap\forall P_j\cdot\omega=\text{FR}\cap j\in[1,k]\cap \text{Now}+\text{Min}\{c(i,j)\}>t_i\times a+t_i\times d\}$$

式中：SQ 表示调度队列；Now 表示当前时间，即上述公式说明任务 t_i 在时间 Now 时还没有调度，

https://dc-china-simulation.researchcommons.org/journal/vol29/iss5/27
DOI: 10.16182/j.issn1004731x.joss.201705027

Dong and Chen: Real-Time Scheduling Algorithm of Dynamic with Fault-Tolerant in

且在所有空闲的执行时间最少的处理机执行结束时间大于任务截止时间，任务 t_i 被拒绝。

为了集中讨论所关心的问题,还做出以下一些假设[16]：

(1) 仅考虑单处理机失效，即在一个处理机失效并恢复之前，不存在第二个处理机失效；

(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) 采用某种故障检测机制，如 fail-signal 机制来检测处理机故障,使得处理故障能在一个任务结束时能被检测出来；

(4) 任务基版本在处理机失效时通知对应的副版本运行，需要一个通信时间，本文设通信时间为 0。

1.2 调度模型

图 1 针对异构分布式系统调度模型进行了描述[13]。在模型中，通过调度队列 SQ (Schedule Queue)，所有的实时任务都到达全局调度机 GS(Global Scheduler)，SQ 中所有任务的调度分配与资源管理都由 GS 负责执行。GS 通过一定调度机制把调度队列中任务分配给每一个处理机 $P_i(i\in[1,k])$，每一个处理机 P_i 都有 3 种不同的状态，并且系统中所有处理机的状态均能够被 GS 监测。需要被处理机 P_i 处理的任务由一个局部队列 LQ(Local Queue)负责接收。

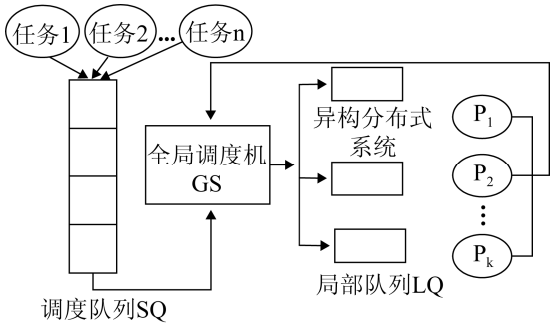

图 1 异构分布式系统的动态调度模型
Fig. 1 Dynamic Scheduling Model of Heterogeneous Distributed Systems

从调度模型知处理机所处的状态与实时任务在处理机上的执行时间相关，为了简化问题和便于说明，设有一组实时任务$\{t_1,\ldots,t_i\}$已经被分配到处理机 P_j 执行，且有一组实时任务$\{t_{i+1},\ldots, t_n\}$在 SQ 中等待处理，调度在 P_i 上的模型图如图 2 所示。

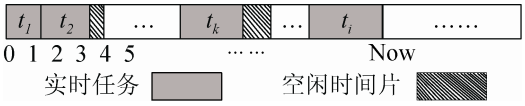

图 2 处理机任务运行模型
Fig. 2 Runtime Model of Processor Tasks

设现在时间为 *Now*。那么对于任务 t_k，其中 $t_k\in$SQ 被调度到处理机 P_i，使 P_i 在接下来状态的形式化描述如下：

如果 if((∃Status(P_j)=FR→$c(k,j)<c(k,i)$)∪ (SQ=∅) ∪(Now+$c(k,i)>t_k.d+t_k.a$))则 Status′ (P_i)=*FR*。

如果 if(∃ *Status*(P_j)=*FR*∩$c(k, i)<\forall c(k, j)$∩ Now+$c(k, i)\leqslant t_k.d+t_k.a$))∪(∀ Status($P_j$)=*AP*|$j\neq i$)则 Status′ ($P_i$)=*AP*。

以上表明，在当前时刻的下一时刻处理机的状态由等待队列与处理机状态相关。如果 Status′(P_i)为空闲状态则存在空闲处理机使当前调度任务在该处理上的执行时间比在 P_i 上执行时间少，或者待调度任务为空。如果 Status′(P_i)为执行状态则当前调度任务在 P_i 上执行时间比任何其它空闲处理执行时间少，或者没有其它的空闲处理机。

1.3 分析模型

算法分析模型主要依据算法特性来定义，本文基于可靠性与可调度性给出 2 种分析模型：

(1) 可靠性分析模型

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，因此实时任务调度算法的设计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[16]定义系统的可靠性为系统中的任务集无故障运行的概率，为了简化问题，并给出了可靠性代价的定义，还做出以下 2 点可靠性模型的假设：

Published by Journal of System Simulation, 2017

1) 如果处理机在 $P_i.\omega$=FR 出现故障，将由备份处理机立即进行替换，所以这种情况不影响系统的可靠性。

2) 如果处理机在 $P_i.\omega \neq FR$ 出现故障后，在第 2 个故障出现前的时间，同样由备份处理机替换故障的处理机来保证系统在一段时间后能恢复到无故障时的状态，并保证系统可正常工作的处理机数目不变。

基于单处理机故障和上述 2 点假设，所以可以仅考虑故障前的情况，而且只需要考虑系统 $P_i.\omega \neq FR$ 时出现故障的情况。因此所有任务集在异构分布式系统上的可靠性代价为:

$$\text{reliabilityCost} = \sum_{i=1}^{k}\Big(\sum_{tj\cdot pcs=i}\lambda i \times c(j,i)\Big), j \in [1,n] \quad (1)$$

减少系统的可靠性代价是提高系统可靠性唯一方法。从式(1)知可靠性代价由任务在处理机的执行时间和处理机可靠性参数决定，为了描述方便我们引入可靠性代价向量 RcV_i=[rcv_{i1}, rcv_{i2},…, rcv_{im}]，其中 rcv_{ij}=(rc_{ij}, j)。rcv_{ij} 表示任务 t_i 被调度到处理机 j 上的可靠性代价，即 rcv_{ij}=$c(i, j)\times\lambda_j$。所以如果我们在每次调度任务时尽可能保证调度任务的 rcv_{ij} 值取最优，那么 reliabilityCost 的值最优。

(2) 可调度分析模型

可调度性模型是算法调度性能指标。主要有任务的平均响应时间，即在系统环境下通过调度一定数量的任务 n，并求出任务的平均响应时间[10] ART(Average Response Time)，平均响应时间越短，表示可调度性越强.，其平均响应时间可形式化描述为：

$$\text{ART} = \frac{\sum(t_i \cdot s - t_i \cdot a)}{n}\left\{i \in [1,n] \bigcap t_i \in T_p\right\} \quad (2)$$

从式(2)知如果要 ART 值越小，那么必须在任务数 n 特定情况下，ART 值与任务到达时间和任务到达率 β 相关。但每两个任务到达的间隔 A 是一个随机变量，它满足指数分布函数：$f_A(x)$= $P\{A \leqslant x\}=1-e^{-\beta x}$，其中 β 表示任务到达率，单位为个/秒，即任务是一个泊松分布的到达过程。但该方法要尽可能保证所有任务都要被成功调度。但从图 2 不难看出某些任务 t 在所有空闲处理机上执行的完成时间都大于任务截止时间，则该任务将永远无法执行。即表示为对于 t_i 满足 $t_i.d+t_i.a> \forall P_j=AP\cap(P_{j\cdot\text{Now}}+c(i, j))$其中 $j\in[1, k]$，那么任务 t_i 是无法得到响应(被拒绝)。针对这一问题并结合文献[14]部分思想给出一种任务接收率测试方法 GR(Guaranteed Ratio)。即一定时间内被执行的任务总数 $T_{\text{Num-Dispose}}$ 与到达系统的任务总数 $T_{\text{Num-Arrive}}$ 比可表示为:

$$\text{GR} = \frac{T_{\text{Num-Dispose}}}{T_{\text{Num-Arrive}}} \times 100\% \quad (3)$$

由式(3)知 GR 越大算法的可调度性越强，反之亦然。

2 启发式调度算法

根据以上可靠性代价及可调度性的定义，本文提出一个可靠性驱动实时容错调度算法 DRFSA (Dynamic and Reliability-driven of Hybrid with Fault-tolerant Scheduling Algorithm)算法和一个可调度性驱动实时容错调度算法 DSFSA(Dynamic and Schedulability-driven of Hybrid with Fault-tolerant Scheduling Algorithm)。算法都是采用准入机制(Admission Control)来调度动态到达的任务，不同的是 DRFSA 算法以增加系统的可靠性为目标，而 DSFSA 算法以增加系统的可调度性为目标。从而在不增加硬件代价的前提下提高系统性能。调度算法启发原则如下：

对于任务基版本分配本文采用被动式副版本技术，尽可能把任务基版本分配到可靠性代价较小或执行时间较小的处理机上，并采用贪婪式调度策略 ASAPE(as soon as possible early)。如果处理机在 $P_i.\omega$=AP 时出现故障，那么，该任务的副版本重新进入 SQ 队列进行调度。

基/副版本方式来实现容错调度，是对任务进行分派调度时，对每个任务生成基版本任务 $t_i.t_p$，复制一个副版本任务 $t_i.t_b$。同时对于任务均先调度

https://dc-china-simulation.researchcommons.org/journal/vol29/iss5/27
DOI: 10.16182/j.issn1004731x.joss.201705027

第 29 卷第 5 期 系统仿真学报 Vol. 29 No. 5
2017 年 5 月 Journal of System Simulation May, 2017

基版本，只有在执行基版本任务的处理机出现故障时才调度副版本。并在任务调度过程中尽可能早的调度任务以减少任务的等待时间。设 $s_t(t_{i,j})$表示任务 t_i 在处理机 P_j 上的执行时间，它必须满足以下两个条件：

(1) $s_t(t_{i,j})$必须晚于任务到达时间 $t_i.a$；

(2) t_i 的完成时间 $s_t(t_{i,j})+c(i, j)$必需早于任务的截止时间，即满足 $s_t(t_{i,j})+c(i, j)\leqslant t_i.d+ t_i.a$；

2.1 DSFSA 算法

输入：任务总数 n，任务到达率 β，处理机个数 k

输出：任务接受率，调度任务集合

1. 初始化任务集合及处理机

2. 根据任务到达率，从调度队列 SQ 中取出任务，并生成到达时间，执行时间集，截止时间

3. 从已到达任务中取 t_i 且满足 $t_i.a$=Min{$t_j.a$}, $j\in[1, k]$并复制生成其副版本 $t_i.t_b$；/*按先来先服务原则*/

4. for j=1 to k do /*选择一个合适的处理机*/

if(Now$\leqslant t_i.d+t_i.a$){/*任务截止时间未到*/

if($c(i, j)$=Min{$c(i, h), h\in[1, k]\}\cap P_j$. ω=FR∩Now + $t_i.c(i, j)\leqslant t_i.d+t_i.a$){

/*把 t_i 调度到处理机 P_j 上执行，并更新相关信息*/

goto 3;

}

else{/*选择可调度性次优处理机*/}

}else(Now$\leqslant t_i.d+t_i.a$){

SQ=SQ－{t_i};RQ=RQ+{t_i};/*加入到拒绝队列*/

goto 3;

}

5. if($\exists P_j.\omega=AP\cap P_j$ 出现故障){

把处理机 P_j 正在执行的任务对应的副版本放入到已到达任务队列;

goto 3;

}

6. if(SQ=$\varnothing$ ∪总任务调度完成){return;}

else{goto 1;}

DSHFA 算法时间复杂度分析如下：调度一个任务选择合适处理机所需时间为 O(k)，k 为处理机个数。由于一个处理机故障，该处理正在执行的任务的副版本会进入到已到达任务队列，所以实现执行的任务数大于 n 但小于 2n，时间为 O(2n)。从上述描述知算法的时间复杂度为 O(2n×k)。

2.2 DRFSA 算法

该算法主要目标是通过减少任务在系统总的可靠性代价来提高算法可靠性。该算法与 DSFSA 算法的基本区别在于算法在选择处理机时以可靠性代价为目标进行选择。算法的主要步骤如下：

DRFSA 算法：

Step 1 根据任务到达率，生成已到达任务列表及任务到达时间

Step 2 按先来先服务原则在已到达任务调度队列中取一个任务 t_i，生成任务副版本并生成可靠任务的可靠性代价向量[$rcv_{i1}, rcv_{i2},\ldots, rcv_{ik}$]

Step 3 选择一个可靠性代价最小且满足 rcv_{ij}=Min{$rcv_{ih}, h\in[1, k]\}\cap P_j.\omega=FR\cap$Now+$t_i.c(i, j)\leqslant t_i.d+t_i.a$ 处理机 P_j。如果此时 P_j 发生故障，该任务对应副版本 t_ib 加入到已到达队列中重新调度。

Step 4 如果任务在执行时间最短处理上执行的截止时间满足 $t_i.d\leqslant$Now+Min{$t_i.a$}，则从 SQ 删除放入拒绝队列 RQ。

Step 5 如果队列 SQ 为空及总任务调度完成说明所有调度成功。

算法的时间复杂度如下：其中生成一个调度任务的可靠性代价向量与调度一个任务选择合适处理机所需时间分别为 O(k)，k 为处理机个数。调度 n 个任务时间复杂度为 O(n×(k+ k))，处理机故障使实际调度任务数大于 n 小于 2n，所以时间复杂度为 O(2n×(k+k))。从上面知两种算法时间复杂度相同。

3 实验与性能分析

本文使用带有酷睿四核 Intel i5 CPU，内存 8G 的 PC 机进行模拟实验，模拟程序语言使用 Java，编译工具使用 Eclipse4.3。实验中的实时任务及异构分布式系统按以下参数生成：

(1) 分布式系统处理机的数量为 k，在范围[5,15]取值；

(2) 处理机的失效概率 λ 符合 Poison 分布在 0.80×10^{-6}/h (10^{-4})到 1.20×10^{-6}/h(10^{-4})间均匀分布；

(3) 实时任务在各个处理机上的执行时间区域为 C_E，且 C_E=[Min_E, Max_E], Min_E 表示任务执行时间最小值，Max_E 表示任务执行时间最大值；

(4) 实时任务的相对截止期限为任务在各处理机上平均实际执行时间的 δ 倍($\delta\geq2.0$)且服从均匀分布；

(5) 任务到达率 Rate，取值在 1 ~10 之间并属于均匀分布。

各个参数的意义及取值范围如表 1 所示。

表 1 模拟实验参数

Tab. 1 Simulation of experimental parameters

参数	说明	取值范围
k	处理机个数	{5, 6,…, 14, 15}
λ	处理机失效率/h	{0.60,0.70, …,1.30}×10^{-6}
Min_E	任务执行时间最小值/s	{3, 4, 5, 6}
Max_E	任务执行时间最大值/s	{7, 8, 9, 10}
δ	任务相对截止期参数/s	[2.0, 4.0]
R	任务到达率	[1, 5]

3.1 算法可靠性代价分析

本节比较两种算法可靠性代价，模拟实验(1)参数为：处理机数 K=15，R 在[1, 5]之间随机取值，任务粒度 C_E=[4, 8]。从图 3 中知可靠性代价随着任务增加成线性增长。因为在其它环境不变情况下调度任务数增加，总执行时间增加，从而由式(1)知，可靠性代价线性增长。

DRFSA 算法可靠性优于 DSFSA 算法，这主要由于 DRFSA 在调度任务时以可靠性为优先调度条件。性能越好处理机执行相同任务时间越短，由式(1)知可靠性代价也越低。

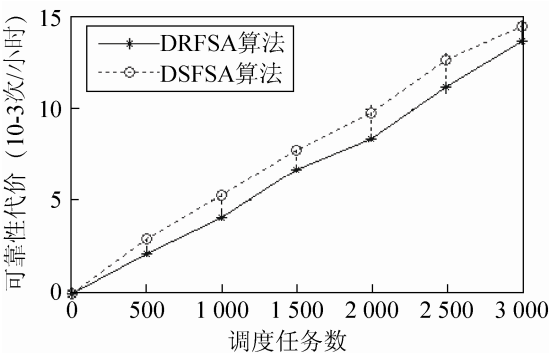

图 3 算法可靠性代价

Fig. 3 Algorithm reliability cost

3.2 算法可调度性分析

本节对比算法可调度性，其参数与模拟实验(1)中的参数类似。实验显示 DSFSA 算法任务接受率优于 DRFSA 算法，这是由于性能优秀的处理器可靠性代码低，任务执行时间短。

由图 4 可知，随着任务的增加，接受率有所下降。但是，当任务达到一定数量后将趋于稳定，如实验环境下 2500~3500 算法 DSFSA 算法接收率为 0.8856，0.8833，0.8823。出现这种情况的主要原因是由于在任务粒度和到达率都确定的情况下，调度任务达到一定数量后达到系统处理能力极限。

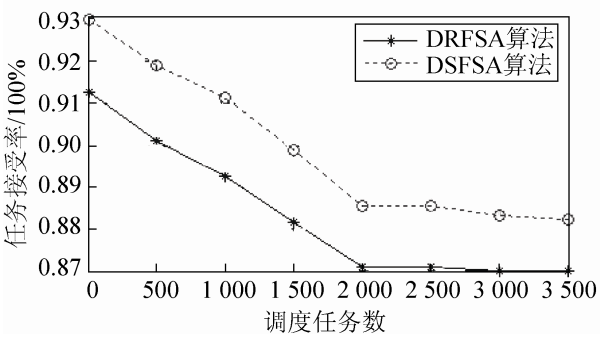

图 4 算法可调度性

Fig. 4 Algorithm schedulability

3.3 可调度性与算法 R 关系

模拟实验(3)主要讨论 DSFSA 算法可调度性与 R 值关系，基本参数为：处理机数 K=10，任务粒

https://dc-china-simulation.researchcommons.org/journal/vol29/iss5/27
DOI: 10.16182/j.issn1004731x.joss.201705027

第 29 卷第 5 期 系统仿真学报 Vol. 29 No. 5
2017 年 5 月 Journal of System Simulation May, 2017

度 C_E=[5, 9]，任务到达率分别取 R=1, R=2, R=3, R=5。图 5 模拟结果表明在其它条件一定的情况下，R 值越大，即单位时间内到达系统的任务越多，任务的接受率越低。这是由于系统在某一时刻处理能力有限，众多任务在一定时间内到达数量越多，被拒绝的任务也就越多。实验还表明在处理机不变情况下，任务的接受率与调度任务数量无关，而与任务到达率相关。这是由于系统在一定时间内处理能力有限，假如任务集中到达，系统将无法处理更多的任务，这就是系统瓶颈，只能通过增加系统处理能力或延迟对任务响应时间加以解决。综上述，任务到达率 R 越大，DSFSA 算法可调度性越差，而与调度任务数量无关。DRFSA 算法也具有相同特征，不再详述。

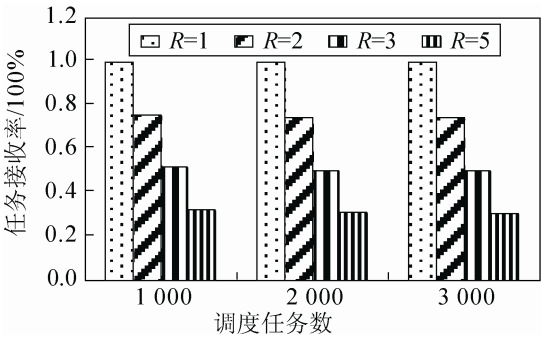

图 5 算法 DSFSA 可调度性与 R 关系
Fig. 5 Relationship between Dispatchability of DSFSA Algorithm and R

3.4 可调度性与算法任务粒度关系

模拟实验(4)主要从任务粒度 C_E 角度分析算法可调度性关系。参数为：处理机数 K=15，任务到达率 R 在[1, 5]之间随机取值，任务截止时间 δ 取[2.0, 4.0]之间随机取值，调度任务数为 3000 时，任务粒度分别取 C_E∈{[3 ,7], [4, 8], [5, 9], [6, 10]}。图 6 模拟结果表明任务粒度越大算法接受率越低，主要是由于在系统处理能力不变的情况下，任务粒度越大单个任务消耗的资源越多，导致系统能够处理的任务数越少。综上所述，任务粒度 C_E 越大，算法可调度性越差。从图 6 中还知 DSFSA 算法接受率略优于 DRFSA 算法，原因同 3.2 节分析。

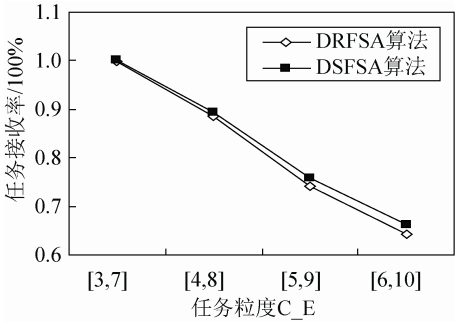

图 6 算法 DSFSA 可调度性与任务粒度 C_E 关系
Fig. 6 Relationship between Schedulability of DSFSA Algorithm and Task Granularity C_E

3.5 可调度性与算法 δ 关系

模拟实验(5)主要讨论 δ 值对算法的可调度性影响，基本参数为：处理机数 K=15，任务到达率在[1, 5]之间随机取值，任务粒度 C_E ∈{[3, 7], [4, 8], [5, 9], [6, 10]}，调度任务数为 3 000 时，δ 取 2.0，3.0，4.0 三组数据分析算法可调度性与 δ 关系。DSFSA 算法与 δ 关系模拟结果如图 7 所示，结果表明在当前环境下任务接收率没有随着 δ 变化而变化，如 C_E=[4, 8]，在 δ=2 时 GR 为 0.896，δ=3 时 GR 为 0.897，δ=4 时 GR 为 0.903。原因是异构系统在繁忙状态下，无论 δ 是否变大对整个系统而言均没有空闲资源处理。除非没有截止时间，一直等到系统有空闲资源，但这不符合实时系统特性。实验表明系统处于繁忙状态，部分任务无论等待多久都有可能无法得到响应，除非提高任务优先级或系统支持可抢占式服务，尽可能响应优先级高的任务，该问题后续研究讨论。综上所述，在系统繁忙情况下，任务相对截止期 δ 值的大小对算法 DSFSA 可调度性影响不大。DRFSA 算法也具有相同特性，不再详述。

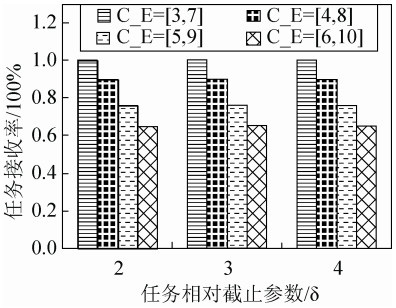

图 7 算法 DSFSA 可调度性与 δ 关系
Fig. 7 Relationship between Dispatchability of DSFSA Algorithm and δ

Published by Journal of System Simulation, 2017

3.6 算法平均响应时间

模拟实验(6)通过平均响应时间来讨论两种算法的可调度性与处理机关系。为保证任务能够全部得到响应，本实验处理机数 K=20，任务到达率 R 在[1, 5]之间随机取值，任务截止时间 δ 取[2.0, 4.0]之间随机取值，任务粒度分别取 C_E=[4, 9]。从下图知，在系统不饱和状态下(即所有任务都有响应)，DSFSA 算法响应时间优于 DRFSA 算法，平均约快 0.026 s 左右。主要原因是 DRFSA 算法在任务调度中选择可靠性代价最少的处理机时，不一定是执行时间最短的处理机，这就使平均响应时间大于 DSFSA 算法。

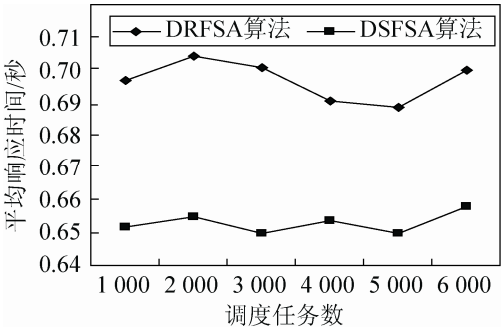

图 8 算法平均响应时间
Fig. 8 Average Response Time of Algorithm

4 算法比较

本文在异构分布式实时系统中，充分考虑现实系统中动态实时任务情况下，基于不同的性能标准，提出并分析两种算法。同时，本文还讨论了在不同任务环境下对算法不同影响。在实际应用中可以根据下面 2 种不同环境选择调度算法：

(1) 当异构分布式系统处于繁忙状态的时，即同一时间任务到达率比较高，此时选择使用 DSFSA 算法，因为所有处理机均在处理任务，此时可靠性代价不变，即 relibalitycost 中 $\sum_{tj\cdot pcs=i} c(j,i)$ 不变，而 λi 是由处理机性决定，所以总可靠性代价不变。因为处理机一直繁忙的状态，所以每个处理机的执行任务时间总和不变，可靠性代价也不变，所以总的可靠性代价不变，此时应该选择调度更多任务。

(2) 当异构分布式系统处于不繁忙状态的时候，应该选择使用 DRFSA 算法，即在系统有空闲处理机的情况下，调度任务时应该选择可靠性代价低的处理机执行，从而提高整个系统的可靠性。综合分析 DSFSA 算法优于 DRFSA 算法。

5 结论

本文的主要研究成果包括 3 点：

(1) 提出了一种基于基/副版本的异构动态容错模型，能够动态的实时调度非周期实时任务。

(2) 在该模型的基础上，根据不同的性能指标通过模拟实验对 DRFSA 算法和 DSFSA 算法做了分析比较，实验结果表明 DRFSA 算法在可靠性方面优于 DSFSA 算法，但是，DSFSA 算法在平均响应时间方面占优。

(3) 对两种算法在不同参数条件下表现出各种不同的特性进行了分析，并给出了研究结论。

接下来的工作有 2 点：(1)本文研究的是一种不可抢占的动态调度算法，因此，下一步将在此基础上结合其它学者的研究成果在可抢占式任务调度方面进行努力。(2)本文还没有考虑动态实时周期任务，因此，研究一种更接近实际应用的具有多种实时任务异构分布式动态调度模型及相关算法将成为下一步的工作重点。

参考文献:

[1] Kieckhafer R M, Walter C J, Finn A M, et al. The MAFT architecture for distributed fault tolerance [J]. IEEE Transactions on Computers (S0018-9340), 2012, 37(4): 398-405.

[2] L H Xu, J Bruck. Deterministic voting in distributed systems using error-correcting codes [J]. IEEE Trans on Parallel and Distributed Systems (S1045-9219), 1998, 9(8): 813-824.

[3] T H Lin, K G Shin. Damage assessment for optimal rollback recovery [J]. IEEE Trans on Computers (S0018-9340), 2013, 47(5): 603-613.

[4] Oh Y, Son SH. Multiprocessor support for real-time fault-tolerant scheduling [J]. Journal of Systems

https://dc-china-simulation.researchcommons.org/journal/vol29/iss5/27
DOI: 10.16182/j.issn1004731x.joss.201705027

第 29 卷第 5 期 系统仿真学报 Vol. 29 No. 5
2017 年 5 月 Journal of System Simulation May, 2017

Architecture (S1383-7621), 2011, 57(5): 485-486.

[5] Al-Omari R, Somani A K, Manimaram G. An adaptive scheme for fault-toleran scheduling of soft real-time tasks in mulitprocessor systems [J]. Journal of Parallel and Distributed Computing (S1097-2803), 2015, 65(5): 595-608.

[6] Pathan R M. Fault-tolerant and real-time scheduling for mixed-criticality systems [J]. Real-Time Systems (S0893-3405), 2014, 50(4): 509-547.

[7] C M Krishna. Fault-tolerant scheduling in homogeneous real-time systems [J]. ACM Computing Surveys (S0360-0300), 2014, 46(4): 1-34.

[8] Abhaya Kumar Samal, Rajib Mall, Chittaranjan Tripathy. Fault tolerant scheduling of hard real-time tasks on multiprocessor system using a hybrid genetic algorithm [J]. Swarm and Evolutionary Computation (S2210-6502), 2014, 14(5): 92-105.

[9] 秦啸, 韩宗芬, 庞丽萍, 等. 混合型实时容错调度算法的设计和性能分析 [J]. 软件学报, 2000, 11(5): 686-693. (QIN Xiao, HAN Zong-fen, PANG Li-ping, et al. Design and Performance Analysis of a Hybrid Real-Time Scheduling Algorithm with Fault-Tolerance [J]. Journal of Software, 2000, 11(5): 686-693.)

[10] 曹洁, 曾国荪. 云环境下周期和非周期混合实时任务双容错调度算法 [J]. 计算机应用, 2015, 35(3): 648-653. (CAO Jie, ZENG Guosun. Dual fault-tolerant scheduling algorithm of periodic and aperiodic hybrid real-time tasks in cloud environment [J]. Journal of Computer Applications, 2015, 35(3): 648-653.)

[11] 庞丽萍, 秦啸, 李胜利, 等. NFRL: 一种分布系统的实时容错调度算法 [J]. 小型微型计算机系统, 2000, 21(3): 232-234. (PANG Li-ping, QIN Xiao, LI Sheng-li, et al. NFRL: An Algorithm For Fault-Tolerant Real-Time Scheduling Based on Distributed Systems [J]. Journal of Chinese Mini-Micro Computer Systems, 2000, 21(3): 232-234.)

[12] 张坤龙, 秦啸, 韩宗芬, 等. 异构分布式实时系统中容错调度模型的研究 [J]. 华中理工大学学报, 2000, 28(8): 17-18. (Zhang Kunlong, Qin Xiao, Han Zongfen, et al. Study of the Model for Fault-Tolerant Scheduling in Heterogeneous Distributed Real-Time Systems [J]. Huazhong Univ. of Sci .& Tech, 2000, 28(8): 17-18.)

[13] Guo Hui, Wang Zhi-guang, Zhou Jian-li. Load Balancing Based Process Scheduling with Fault-Tolerance in Heterogeneous Distributed Systems [J]. Chinese Journal of Computers (S0254-4164), 2013, 28(11): 1807-1816.

[14] 罗威, 阳富民, 庞丽萍, 等. 异构分布式系统中实时周期任务的容错调度算法 [J]. 计算机学报, 2007, 30(10): 1740-1749. (LUO Wei, YANG Fu-Min, PANG Li-Ping, et al. A Real-Time Fault-Tolerant Schedul ing Algorithm of Periodic Tasks in Heterogeneous Distributed Systems [J]. Chinese Journal of Computers, 2007, 30(10): 1740-1749.)

[15] 邓建波, 张立臣, 邓惠敏. 异构分布式系统混合型实时容错调度算法 [J]. 计算机科学, 2011, 38(3): 87-92. (DENG Jian-bo, ZHANG Li-chen, DENG Hui-mi. Real-time Scheduling Algorithm of Hybrid with Fault-Tolerant in Heterogeneous Distributed Systems [J]. Computer Science, 2011, 38(3): 87-92.)

[16] Garey R, Johnson D S. Computers and Intractability: a Guide to the Theory of NP-Completeness [M]. New York, USA: W.H. Freeman Company, 2015.

[17] 陈晗鸣, 罗威, 李明辉. 分布式系统中基于主/副版本的实时容错调度综述 [J]. 计算机应用研究, 2012, 29(11): 4017-4022. (CHEN Han-ming, LUO Wei, LI Ming-hui. Survey of primary /backup copy based real-time and fault-tolerant scheduling in distributed systems [J]. Application Research of Computers, 2012, 29(11): 4017-4022.)

[18] 张童, 刘云生, 查亚兵. 面向服务仿真系统的最大可靠性容错调度算法 [J]. 系统仿真学报, 2009, 21(4): 1816-1818. (ZHANG Tong, LIU Yun-sheng, ZHA Ya-bing. Fault-tolerant Scheduling Algorithm for Maximizing Reliability of Service-Oriented Simulation System [J]. Journal of System Simulation, 2009, 21(4): 1816-1818.)

[19] 景维鹏, 吴智博, 刘宏伟, 等. 支持优先级约束任务的容错调度算法 [J]. 清华大学学报(自然科学版), 2011, 51(增 1): 1440-1444. (JING Wei-peng, WU Zhi-bo, LIU Hong-wei, et al. Fault-tolerant scheduling algorithm for precedence constrained tasks [J]. Journal of Tsinghua University (Science and Technology), 2011, 51(S1): 1440-1444.)