

基于动态异构模型的非周期性任务容错调度

叶盛钊, 赵 博, 魏 帅, 殷从月

(国家数字交换系统工程技术研究中心, 郑州 450002)

摘 要: 拟态技术可有效解决实时系统中的安全问题, 但其异构冗余的特性会增加系统时延。为此, 以拟态工控处理机架构为基础, 在动态异构的多模冗余场景下, 结合具体表决策略并执行清洗切换任务, 提出一种硬实时非周期性任务容错调度算法。仿真结果表明, 相比静态异构模型下的 DRFTS 算法, 该算法在保证硬实时的条件下, 能够提高调度成功率。

关键词: 动态异构模型; 非周期性任务; 容错调度; 2-out-of- n 表决; 清洗切换

中文引用格式: 叶盛钊, 赵 博, 魏 帅, 等. 基于动态异构模型的非周期性任务容错调度[J]. 计算机工程, 2018, 44(10): 6-13.

英文引用格式: YE Shengzhao, ZHAO Bo, WEI Shuai, et al. Fault tolerant scheduling for aperiodic tasks based on dynamic heterogeneous model[J]. Computer Engineering, 2018, 44(10): 6-13.

Fault Tolerant Scheduling for Aperiodic Tasks Based on Dynamic Heterogeneous Model

YE Shengzhao, ZHAO Bo, WEI Shuai, YIN Congyue

(National Digital Switching System Engineering and Technological R&D Center, Zhengzhou 450002, China)

[Abstract] Mimic defense techniques can effectively solve the security problems in real-time systems, but its heterogeneous redundancy will increase the system delay. To solve this problem, based on the architecture of the mimic processor, a hard time and aperiodic tasks fault tolerant scheduling algorithm is proposed in the dynamic heterogeneous multi-mode redundancy scenario, combining the specific voting strategy and implementing the cleaning handover tasks. Simulation results show that compared with the static-heterogeneous-model-based DRFTS algorithm, this algorithm can improve the guarantee ratio under the condition of hard real-time.

[Key words] dynamic heterogeneous model; aperiodic tasks; fault tolerant scheduling; 2-out-of- n voting; cleaning handover

DOI: 10.19678/j.issn.1000-3428.0051389

0 概述

近年来, 伊朗核电站“震网”病毒事件、乌克兰电力系统瘫痪等针对工控系统的攻击事件频发, 导致安全因素正逐渐成为工控系统设计中不可忽视的一部分。系统在面对日益严峻的安全威胁且自身资源条件有限时, 需要兼顾可靠性与可信性, 能够抵抗故障以及外部的攻击行为。在此背景下, 融合了动态异构冗余 (Dynamic Heterogeneous Redundant, DHR) 的拟态防御技术为解决上述需求提供了一种思路^[1]。相对传统防御技术的静态、确定性等特点, 拟态防御技术不依赖单一构件的安全以及先验知识,

其利用多维重构、多模表决、负反馈控制等技术保证了系统可以得到高可靠、高可信的输出。目前, 基于拟态防御的工控处理机^[2]、Web 服务器^[3]等产品已经得到越来越多的关注。

拟态防御技术引入 DHR 因素后, 在提升可靠性与可信性的同时, 必然会增加清洗切换、输出同步等开销, 从而导致系统时延的增加。而安全关键领域广泛使用的系统通常对任务执行有较高的实时性要求, 尤其是硬实时性任务, 要求其必须在固定时间限制, 即截止期前完成, 否则会造成结果输出无意义。

针对上述问题, 本文参考拟态工控处理机架

基金项目: 国家科技重大专项“高安全等级网络基础设施关键装备核心芯片及软件研发”(2017ZX01030301); 国家自然科学基金面上项目“网络空间拟态安全异构冗余机制研究”(61572520); 国家自然科学基金创新研究群体项目“网络空间拟态防御基础理论研究”(61521003)。

作者简介: 叶盛钊 (1994—), 男, 硕士研究生, 主研方向为拟态防御、容错调度、安全性调度; 赵 博, 助理研究员; 魏 帅, 讲师; 殷从月, 硕士研究生。

收稿日期: 2018-04-28

修回日期: 2018-06-11

E-mail: itisanickname@outlook.com

构^[2],提出一种基于动态异构模型的非周期性任务实时容错调度算法 DHFTS,并通过仿真与对比来分析该算法在调度成功率和平均执行时间方面的性能。

1 研究现状

常用的容错调度算法多数基于主/副版本备份 (Primary-Backup, PB) 和三模冗余 (Triple Modular Redundancy, TMR)。其中, PB 是实时容错调度中的主流容错方案,其在多处理机的环境下通过分配到达任务的主版本和副版本至不同的处理机,以保证在任务主版本出现故障或者输出异常的情况下,通过执行副版本使任务能在最终的截止期前完成。在主/副版本的调度过程中,需要根据时间裕度来考虑主动副版本与被动副版本的策略问题,同时还需要考虑各种资源约束,以达到降低使用节点数目、提高任务接受率的目的。文献[4]发现 PB 中主版本执行成功时不用执行被动副版本且主/副版本的调度策略可以不同,对非周期非抢占任务的副版本提出了重叠和释放的技术,其可以降低容错所需的节点数量并提高任务接受率。文献[5]提出一种自适应的重叠方案,其通过错误率和任务裕度计算重叠部分的大小。文献[6]提出一种延迟主动副版本以达到最小化处理数的启发式算法,能够在保证实时性与可靠性的同时减少冗余。文献[7]在异构集群系统的场景下探讨具有 QoS 需求的容错调度,其能够根据系统负载情况自适应地调整任务执行的 QoS 级别以及副版本的运行方式(主动或被动),以此提高资源的整体利用率。文献[8-9]分别在卫星观测与云环境下拓展 PB 容错的模型,并在具体的容错模型下提出调度算法。

然而,运行基于 PB 技术的算法通常需要一些前提条件^[10]:1) 主版本出错时副版本能得到正确结果;2) 如接受性测试的硬件检测机制能发现处理机错误。条件1)认为不同处理机上的任务执行同时出错为小概率事件,然而在同构系统中很容易出现共模错误的情况;条件2)通过增加单点成本来解决可靠性问题,即依据接受性测试对任务某一版本的检测来保证可靠性,但是,相比单纯增加单点成本,使用多版本表决时更易得到可靠的结果,也更节约成本^[11]。

TMR 技术通过多处理机冗余并行执行任务的方式,由表决得到最终结果。文献[12]提出一种同时融合 TMR、PB 以及非精确计算 (Imprecise Computation, IC) 的容错调度算法,可满足多场景下的可靠性、实时性需求,且能够提高可调度性。文献[13]在不可靠的副版本场景下引入 TMR 技术。文献[10]根据截止期决定三冗余的实现方式(时间冗余、空间冗余、混合冗

余),同时结合副版本覆盖与释放技术,以提高可调度性。文献[14]为解决卫星通信中引入 N 版本技术后开销较大的问题,对有依赖关系的任务采用多处理器的任务调度算法。

以上容错调度算法所用的 TMR 模型均是静态架构下的同构冗余,而相同的设计面对共模或者同态故障时很容易失去整体的容错功能。拟态防御技术除了要应对随机故障外,还要防御利用漏洞后门信息的入侵行为,因而需要避免由同构导致的相同错误输出,以及由静态导致的利用已知漏洞和已验证方法且易于实现的“单向透明、内外配合”的攻击方式^[1]。本文参照拟态工控处理机的架构,探讨动态异构因素对容错调度的实时性与可靠性的影响。

2 容错调度模型

2.1 调度器模型

拟态防御的 DHR 模型结合多维重构、多模表决、负反馈控制等技术,同时融合 DHR 的架构设计优势,能够保护拟态界中的执行体组成系统,将复杂多变的安全性问题归一化为概率可控的可靠性问题。本文参照文献[1-2]设计一种动态异构的调度器模型,如图1所示。

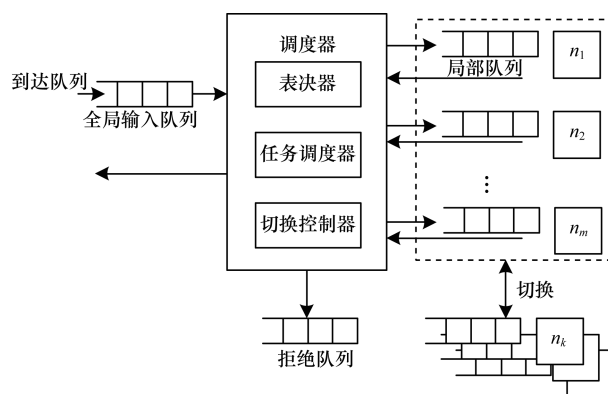


图1 调度器模型

在该动态异构模型中,使用集中式的调度器来连接异构的处理机以及外围电路,其中,调度器的任务包括:1) 为用户提供统一的接口,将系统外部的输入复制分发至各异构处理机(该处理机即为拟态 DHR 模型中提到的执行体);2) 对处理机输出进行表决并得到一致的结果;3) 当发现异常状态处理机时进行清洗切换。

调度器与处理机节点并行运行,其中,调度器包含表决器(可根据表决策略输出同时判别需要清洗的异常状态处理机)、任务调度器(调度非周期性任务到特定的处理机上)及切换控制器(切换异常处理机),需要维持外部输入的等待调度的全局输入队列以及存储待表决结果的表决队列(表决队列没有在图1中展示);而异构体也有局部队列,保存需要进

行处理的混合实时任务,不能调度的节点将被丢弃到拒绝队列中。该模型采用清洗恢复技术^[15]处理异常处理机,在确认处理机受到攻击(如表决结果较差),或者是按照前摄式切换策略在需要切换的情况下,将已经调度的处理机进行切换。本文选用异常状态触发的反馈式切换策略搭配固定周期的前摄式切换策略,关于切换策略的细节不是本文研究的重点。

2.2 任务模型

设动态异构模型中的一组异构处理机节点集合为 $N = \{n_1, n_2, \dots, n_m, \dots, n_M\}$ 。其中, N 中的前 m 个处理机为并行的异构处理机, M 表示系统中处理机总数量。 $APER = (\omega_{ij})_{m \times n}$ 为系统中非周期性任务集矩阵, 其中, ω_{ij} 表示可能要在系统中运行的第 i ($1 \leq i \leq m$) 个处理机上的第 j ($1 \leq j \leq n$) 个到达的非周期性任务。 ω_{ij} 由一个三元组 (a_{ij}, C_{ij}, D_{ij}) 来描述, 其中, a_{ij} 表示第 j 个任务的到达时间, C_{ij} 表示第 j 个任务的最坏情况执行时间, 此处不讨论任务提前结束的资源回收问题, 基本可以认为不被打断并在正常状态下执行完成的任务就是用了 C_{ij} 的时间, D_{ij} 表示第 j 个任务的相对截止期。对于 n 个非周期性任务, 按照非递减顺序定义到达时间数组为 $\{a_1, a_2, \dots, a_n\}$ 。

通过动态异构模型下的实时调度, 调度器将任务分发到各异构处理机, 安排好每个处理机上任务执行的顺序, 同时通过异构执行、表决输出以及动态切换的技术手段保证到达任务的可靠运行。为简化研究, 本文讨论的任务模型基于以下假设:

- 1) 出于对实时性影响的考虑, 假定异构处理机对同一任务的执行时间不同。
- 2) 不考虑存储结果的队列限制。
- 3) 非周期性任务的执行时间在调度前已知。
- 4) 各处理机节点间相互独立, 不存在通信与同步等信息的交流。
- 5) 任务相互独立且可抢占, 忽略上下文切换以及调度开销的影响, 同时任务只有在被抢占的情况下才会被挂起。
- 6) 在进行清洗切换时, 假定用于切换的备用处理机状态已经与正常执行的处理机状态同步, 且忽略同步开销。

3 容错调度算法 DHFTS

3.1 非周期性任务调度

本文讨论的实时系统中的非周期性任务具有偶发性与强实时性的特点。一方面, 由于非周期性任务是针对外部事件的响应, 只有当达到处理机时, 才能够触发调度器对其进行可调度性判断; 另一方面,

在类似工控系统的实时系统中, 强实时性的非周期性任务需要保证任务在截止期内可靠可信地进行执行处理。综合非周期性任务的偶发性、强实时性的需求, 以及前文中可抢占的假设, 本文分配局部队列的任务时采用 EDF (Earliest Deadline First) 调度算法^[16]。EDF 算法是一种在单处理机上针对动态优先级的可抢占调度算法, 采用 EDF 算法的处理机在理论上最多能以 100% 的利用率处理任务。EDF 算法根据当前时刻距离任务的截止期长短排定优先级, 距离越短优先级越高。就绪任务中的最高优先级任务会抢占低优先级任务的执行时间, 此时低优先级任务被挂起。

可调度性要求任务需要在截止期内完成, 即处理机 i 上任务 j 的完成时间 $f_{ij}(t)$ 小于任务 j 的绝对截止期 d_{ij} ($d_{ij} = a_{ij} + D_{ij}$)。为方便进行形式化描述, 按照优先级非递增排序组织系统中当前运行和已经就绪的 L 个任务, 得到处理机 i 上的任务集 $\Omega = \{\omega_{i1}, \omega_{i2}, \dots, \omega_{iL}\}$ 。由于 EDF 算法的可抢占性, 新到达的任务 $\omega_{i(\text{new})}$ 将抢占低于其优先级的任务的执行顺序, 同时影响低优先级任务的可调度性。假定插入新任务后的任务集为 $\Omega' = \{\omega_{i1}, \omega_{i2}, \dots, \omega_{iL}, \omega_{i(\text{new})}, \omega_{i(L+1)}, \dots, \omega_{iL}\}$, 设 $C_{ij}(t)$ 为处理机 i 上任务 j 在 t 时刻的剩余执行时间, 文献[1]给出了加入任务 $\omega_{i(\text{new})}$ 后可调度性判别的充要条件:

对于 Ω' 中的任务 $\{\omega_{i(\text{new})}, \omega_{i(L+1)}, \dots, \omega_{iL}\}$, 均满足:

$$f_{ij}(t) = a_{ij} + \sum_{k=1}^j C_{ik}(t) \leq d_{ij} \quad (1)$$

本文针对非周期性任务的输出, 在图 1 的调度器模型中选用 2-out-of- n 算法。对于在 m 个处理机中执行的任务 i , 只要有 2 个处理机给出的结果在表决误差之内, 该算法即产生输出。文献[17]验证了对于表决空间足够大且故障独立的冗余系统, 2 个相同结果得到的表决输出能够满足实际的可靠性需求。对于冗余版本不能在所有处理机上满足式(1)的任务, 若能保证在 2 个以上的处理机上可调度, 则继续将该任务分发至合适的处理机。此时满足的可调度性条件为:

$$\begin{aligned} & \exists \text{ 节点 } i' \text{ 和 } i'' (i' \neq i'') \text{ 且 } i', i'' \in [n_1, n_2, \dots, n_m], \\ & \text{使得 } f_{i'j} \text{ 和 } f_{i''j} \text{ 同时满足式(1)} \end{aligned} \quad (2)$$

使用 2-out-of- n 算法的另一个目的是实现实时性。任务的最终输出需要考虑输入分发、异构体执行、表决输出等, 虽然本文不探讨表决输出开销对可调度性的影响, 但是实时系统的最终响应速度与表决策略有关。按照先后顺序给出任务 j 的完成时间序列 $F_j = \{f_{1j}, f_{2j}, \dots, f_{mj}\}$ 以及其相应的输出结果序列 $O_j = \{o_{1j}, o_{2j}, \dots, o_{mj}\}$, 则表决开始时刻 $t_{\text{vote}} = f_{kj} (k \in [2, m])$ 的条件为:

有且仅有 $o_{ij} (i \in [1, k-1])$ 使得 $o_{kj} = o_{ij}$,

且 $o_{pj} \neq o_{qj} (p, q \in [1, k-1], p \neq i, q \neq i)$ (3)

在得到任务 i 的表决输出结果后, 未完成任务执行的冗余处理机将继续执行。一方面, 因为在处理机并行执行 2-out-of- n 表决的条件下, 输出实时性得到了保证, 而频繁调整处理机中的任务调度时开销较大, 反而会降低处理机的响应速度; 另一方面, 从安全性和可靠性角度出发, 冗余处理机继续执行, 便于通过比对筛选出的故障或者出错的处理机来及时地进行切换操作。本文 DHFTS 算法的任务分配伪代码如下:

while true do

/* assume the j th task ω_{*j} arrives at the queue */

AllocateNodeNum $_j \leftarrow 0$

TaskFinishNum $_j \leftarrow 0$

CurrentTime $\leftarrow a_j$

add ω_{*j} to all the m nodes

initialize ShiftCircle

for each node n_i in the $\{n_1, n_2, \dots, n_m\}$ do

if ω_{ij} match condition(1) then

AllocateNodeNum $_j++$

end if

end for

if AllocateNodeNum $_j < 2$ then

delete ω_{*j} in all the m nodes

/* allocation fails */

end if

end while

3.2 处理机动态切换调度策略

非周期性任务在进行切换时, 采取前摄式切换策略或反馈式切换策略, 都需要将调度器已经复制并下达到局部队列中的任务迁移到其他无故障处理机中。在同构系统中, 由于实现方式相同, 迁移时无需考虑任务能否在新处理机中满足可调度性的问题。但是在动态异构模型中, 新加入运行的异构处理机可能由于处理某些任务时的性能差异导致迁移过程中出现某些任务不可调度的问题。

本文采用基于任务执行情况的丢弃策略来解决由异构处理机间迁移导致的任务拒绝问题。该策略的设计思想是在冗余并行的执行方式下, 对于可能提高系统内任务拒绝率的较高优先级的任务, 新处理机选择性地丢弃某些已经在其他处理机上完成的任务, 或者能在其他处理机上以较高可靠性完成的任务, 以此保证在动态迁移条件下系统内较低的任务拒绝率。

此处沿用第 3.1 节的形式化描述。设 t 时刻需要迁移到新处理机 j' 上的任务集 $\Omega = \{\omega_{i1}, \omega_{i2}, \dots, \omega_{iL}\}$, 且这 L 个任务按照优先级非递增排序进行组织。设使

用式(1)检查迁移到新处理机上的任务的可调度性时, 第一个不满足条件的任务为 $\omega_{i(l+1)}$, 则对于 $\{\omega_{i1}, \omega_{i2}, \dots, \omega_{il}\}$, 从最高优先级的任务开始删除已经达到系统输出条件的任务(即已满足式(3)); 若不存在这种任务, 则检查 $\{\omega_{i1}, \omega_{i2}, \dots, \omega_{il}\}$ 中是否存在已经分配到 2 个以上的其他正常处理机中运行的任务; 如果 $\omega_{i(l+1)}$ 仍然不可调度, 则从最高优先级的任务开始丢弃, 直至 $\omega_{i(l+1)}$ 满足调度条件。DHFTS 算法的处理机切换伪代码如下:

if ShiftCircle $_i < 0$ or n_i needs reactive shift then

$n_i \leftrightarrow n_k$

/* $n_k \in \{n_{m+1}, n_{m+2}, \dots, n_M\}$ */

if $\exists \omega_{i(l+1)}$ in the local input queue of the new n_i doesn't match condition(1) then

for each task $\omega_{ij'}$ in the $\{\omega_{i1}, \omega_{i2}, \dots, \omega_{il}\}$ do

if TaskFinishNum $_j \geq 2$ then

delete $\omega_{ij'}$

/* reject task $\omega_{ij'}$ */

else if AllocateNodeNum $_j \geq 3$ then

AllocateNodeNum $_j--$

delete $\omega_{ij'}$

end if

if $\forall \omega_{iq}$ in the local input queue of the new n_i match condition(1) then

break

end if

end for

end if

if $\exists \omega_{i(l+1)}$ in the local input queue of the new n_i doesn't match condition(1) then

for each task $\omega_{ij'}$ in the $\{\omega_{i1}, \omega_{i2}, \dots, \omega_{il}\}$ do

if $\forall \omega_{iq}$ in the local input queue of the new n_i match condition(1) then

break

else

delete $\omega_{ij'}$

end if

end for

end if

end if

4 实验结果与分析

本文通过仿真实验来测试 DHFTS 算法的调度性能。将不使用故障切换的基于非相似冗余(Dissimilar Redundancy, DR)模型的容错调度算法命名为 DRFTS, DRFTS 算法由于无法切换故障主机, 在出现错误时, 其不能达到表决输出条件而无法正常工作。本实验通过 DHFTS 与 DRFTS 的对比来验证在容错调度中引入动态切换的作用。除 DRFTS 算法外, 还将本文算法与文献[13]算法进行对比, 该算法结合了 PB 与

TMR 技术,其算法约束如下:

1) 使用被动副版本时,副版本为主版本的简化,其最差执行时间将有所减少。

2) 主/副版本在不同处理机上运行。

3) 如果 2 个任务的主版本在相同处理机上,同时任务副版本也在相同处理机上,则副版本任务之间不可以重叠。

4) 各节点均为同构节点,相同任务的运行时间相同。

5) 当出现故障时,将同时运行三冗余的副版本,处理结果在表决后输出。

本文选取该算法进行对比的目的是讨论动态异构算法和结合主备与冗余方式的算法在性能上的差别。

仿真实验参考文献[7]中的部分方法及参数,使用 Python 实现以上 3 种算法。实验环境为 Windows 10,硬件平台为 Intel-core-i5,8 GB 内存。实验中的表决均使用 2-out-of- n 算法,同时对于文献[13]算法,改变其约束 4) 为异构处理机节点,并在单处理机上也采用 EDF 算法来调度到达的任务。本实验着重比较不同算法的调度成功率以及平均完成时间。

4.1 仿真方法与参数

文献[7]根据其提出的异构集群系统中有服务质量(QoS)需求的调度算法设计仿真实验,通过引入节点异构性与任务异构性,分别描述节点间的差异与任务间的差异,在此基础上,考虑不同 QoS 的影响,分析节点、任务难度差异和 QoS 对执行时间 C_{ij} 的影响。用参数 $NodePowerAverage$ 和 $NodePowerSpan$ 分别表示节点平均处理能力与浮动范围,节点异构因子 p_i 取 $[(NodePowerAverage - NodePowerSpan), (NodePowerAverage + NodePowerSpan)]$ 之间的随机数;用参数 $TaskHardnessAverage$ 和 $TaskHardnessSpan$ 分别表示任务平均处理难度和浮动范围,任务异构因子 h_j 取 $[(TaskHardnessAverage - TaskHardnessSpan), (TaskHardnessAverage + TaskHardnessSpan)]$ 之间的随机数。执行时间 C_{ij} 的计算公式为:

$$C_{ij} = q \times BaseTime \times 10 \times \frac{h_j}{p_i}$$

其中, q 为文献[7]中探讨的任务 j 的 QoS 级别, $BaseTime$ 为一个影响执行时间的随机数。本文不讨论 QoS 级别对执行时间的影响,但是为了便于实现,本文将 q 值设置为固定值 0.5。

除对 q 值进行简化处理,本文针对非周期性任务的场景,改动文献[7]中关于固定时间间隔的设置,生成长度为 $TaskNum$ 的到达时间数组 $\{a_1, a_2, \dots, a_i, a_{i+1}, \dots\}$,令相邻到达时间 a_i 和 a_{i+1} ($0 \leq i \leq$

$TaskNum - 1$) 之间的间隔 $TimeInterval$ 服从参数为 $\frac{1}{\lambda}$ 的指数分布。任务的截止期 d_j 参照文献[7],仍然根据 $d_j = a_j + \max(C_{ij}) + TimeBaseDeadline$ 生成,其中, $TimeBaseDeadline$ 控制截止期 d_j 的宽松程度。同时,本文还限定非周期性任务的种类 $TaskCategory$,通过随机产生任务序列号来确定特定时间内到达的是何任务,这一处理方式是考虑到工业控制等实时系统中运行的任务均是经过测试保证了可行性的特定任务。设无故障情况下的前摄式清洗切换周期 $ShiftCircle$ 为固定值,其也随特定的清洗任务调度策略影响系统的性能。表 1 所示为实验参数设置。其中,参数取值为(固定值)或(最小值,最大值,步长)。

表 1 实验参数设置

参数名称	参数取值
$NodeNum$	(4)
$TaskNum$	(10 000)
$TaskCategory$	(100)
$NodePowerAverage$	(700)
$NodePowerSpan$	(360)
$TaskHardnessAverage$	(300)
$TaskHardnessSpan$	(120)
$TimeBaseDeadline$	(360, 1 440, —) 取随机的固定值
$BaseTime$	(55, 65, —) 取随机的固定值
$\frac{1}{\lambda}$	(20, 300, 20)
$ShiftCircle$	(1 000, 10 000, 1 000)
$ErrorBaseInterval$	100, 500, 1 000, 2 000
$ErrorIntervalSpan$	在相应 $ErrorBaseInterval$ 内进行等比例取值
$FaultTolerantTimes$	(2, 10, 2)

4.2 任务到达时间间隔对算法性能的影响

任务到达时间间隔服从参数为 $\frac{1}{\lambda}$ 的指数分布。

本节实验观察无故障条件下 $\frac{1}{\lambda}$ 对调度成功率及平均执行时间(此处平均执行时间只统计调度成功任务的执行时间)的影响。实验中的 $ShiftCircle$ 为固定值 10 000。在图 2(a)中, $\frac{1}{\lambda}$ 的取值为 20 ~ 300, 步长为 20, 为便于观察变化趋势,还选取 $\frac{1}{\lambda}$ 为 500、1 000 和 10 000 的点(为方便展示,图中横坐标只取 0 ~ 400)。在图 2(b)中, $\frac{1}{\lambda}$ 的取值为 10 ~ 3 200(横坐标只取 0 ~ 200)。由表 1 可知,仿真中使用了大量的随机数,为得到较平均的结果,每个 $\frac{1}{\lambda}$ 的取值进行 50 次仿真并取均值。

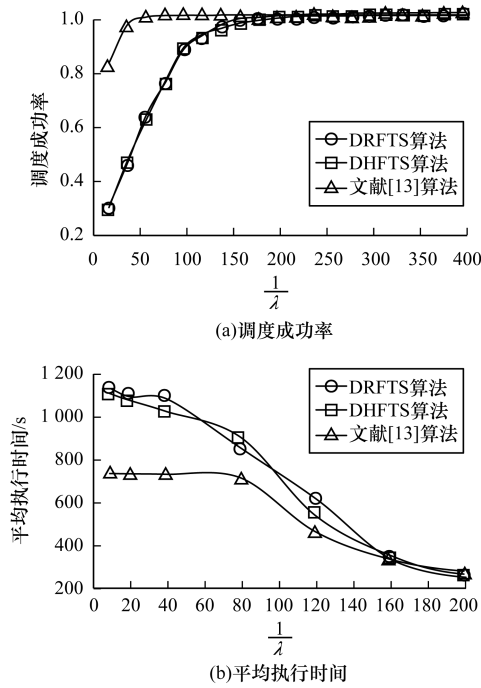


图2 $1/\lambda$ 对3种算法性能的影响

由图2(a)可以看出,随着 $1/\lambda$ 的增加,任务到达时间间隔增加,3种算法的调度成功率均呈上升趋势。其中,DRFTS与DHFTS的上升曲线基本相同,均在 $1/\lambda = 220$ 附近趋于平稳;而文献[13]算法在 $1/\lambda = 60$ 附近趋于平稳,且在 $60 \sim 220$ 的区间内, $1/\lambda$ 越小,其调度成功率与DRFTS和DHFTS差距越大,这说明冗余并行导致系统负载高时可调度性下降明显,而文献[13]算法在无故障条件下使用重叠方式调度任务的副版本,并在主版本执行成功后取消副版本,等同于每次完成一个任务只需一个处理机。

由图2(b)可以看出,随着 $1/\lambda$ 的增加,3种算法平均执行时间相应减少,且均在 $1/\lambda = 40$ 时开始下降。该结果表明,虽然采用了2-out-of-n的表决方式,但是在无故障条件下,系统负载增加到一定程度时,相对于使用PB方式的文献[13]算法,DHFTS和DRFTS在实时性上有明显不足。本实验中,在 $1/\lambda \geq 160$ (此时调度成功率在97%以上)区间内进行比较时,DHFTS的平均执行时间在文献[13]算法的6.8%以内浮动(考虑到仿真中随机值带来的误差);而 $1/\lambda \leq 40$ 时,该数值会增加40.3%~50.8%(此时DHFTS的调度成功率已经不足47%,参考价值不大)。

4.3 前摄式切换周期对算法性能的影响

本节实验基于第4.2节的结果,取DHFTS算法调度成功率即将平稳的点,即 $1/\lambda = 220$ 来观察前摄式切换周期对DHFTS算法性能的影响。*ShiftCircle*取值为100~1000,步长为100。同时,为方便观察变化趋势较快的区域,还选取*ShiftCircle*为100~200的点(步长为20)进行仿真,结果如图3所示。由图3(a)可以看出,当*ShiftCircle*增加时,调度成功率呈现出线性的增加趋势。在本节实验条件下,当*ShiftCircle*达到2000时,调度成功率趋于稳定,且为99%以上。这说明在无故障条件下,前摄式切换的频率越高,任务越需要频繁地判断可调度性以及重复执行,导致调度成功率降低。由图3(b)可以看出,当*ShiftCircle*增加时,平均执行时间下降,在切换周期为500左右时趋于稳定。这说明在无故障条件下,前摄式切换的频率越高,任务越频繁地被打断并重复执行,从而增加了执行时间。

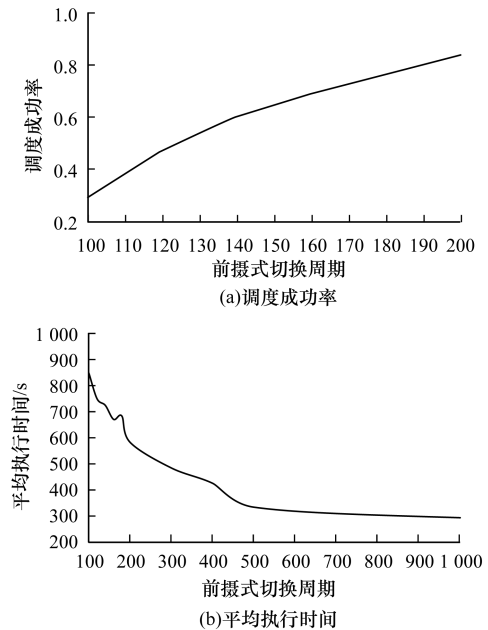


图3 前摄式切换周期对DHFTS算法性能的影响

4.4 故障时间间隔异构度对算法性能的影响

第4.2节、第4.3节均是在无故障条件下的讨论,第4.4节~第4.6节均是在有故障情况下的实验与分析。根据前文的结果,设置参数取值为 $1/\lambda = 220$ 、*ShiftCircle* = 2000。

拟态架构中的异构体现在功能等价体的实现方式不同,而对于性能的差异强调并不明显,因此,本次实验讨论故障时间间隔的异构度*ErrorIntervalSpan*对算法性能的影响,且本次讨论的故障类型参照文献[18-19]中的瞬时故障类

型,故障持续时间短且只影响任务的一次执行(实验中任务失败即为无输出记录)。瞬时故障的产生符合参数为 λ_{fault} 的 Poisson 分布。故障时间间隔服从指数分布,参数 $\frac{1}{\lambda_{\text{fault}}}$ 取 $[(\text{ErrorBaseInterval} - \text{ErrorIntervalSpan}), (\text{ErrorBaseInterval} + \text{ErrorIntervalSpan})]$ 之间的随机数。 $\frac{1}{\lambda_{\text{fault}}}$ 取正整数,因此,有 $\text{ErrorIntervalSpan} \leq \text{ErrorBaseInterval}$ 。实验对 ErrorBaseInterval 取 2 000、1 000、500、100 的 4 种情况进行探讨,每种情况的 ErrorIntervalSpan 在相应的 ErrorBaseInterval 上等比例取 5 个值(如 $\text{ErrorBaseInterval} = 2\,000$ 时, ErrorIntervalSpan 取 0.500、1 000、1 500、2 000)。实验结果如图 4 所示,其中,每个 ErrorBaseInterval 中的 5 个直方图分别对应 ErrorIntervalSpan 的 5 个取值。由图 4(a)和图 4(b)可以看出,与故障时间间隔异构度相关的 ErrorIntervalSpan 在相同 ErrorBaseInterval 下无明确变化趋势,说明在故障环境下,异构冗余的表决输出方式能够屏蔽故障时间之间的差异,影响调度成功率与平均执行时间的主要因素为 ErrorBaseInterval 。这是因为当 ErrorBaseInterval 相同时,故障时间间隔异构度的增加虽然容易导致出现故障时间间隔特别小的不可靠的处理机,但是由于反馈式触发的影响,能够保证可靠的处理机尽可能长的处于运行状态,因而降低了异构的影响。

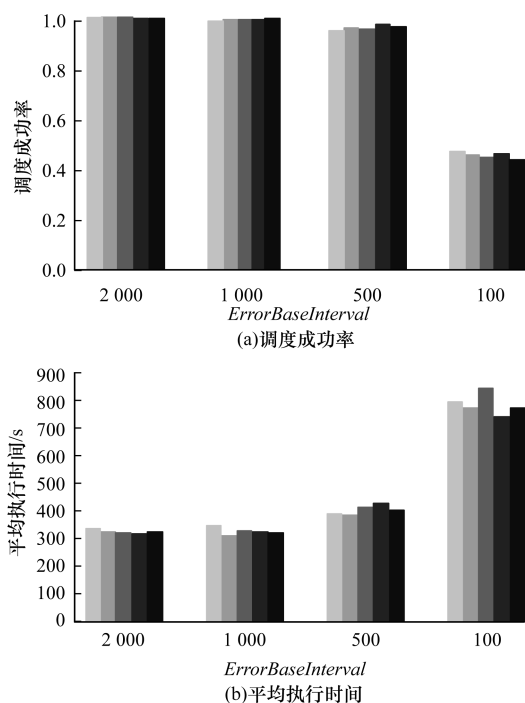


图 4 不同 ErrorBaseInterval 条件下 ErrorIntervalSpan 对 DHFTS 算法性能的影响

4.5 错误时间间隔对算法性能的影响

由第 4.4 节的结果可知,对于 DHFTS 算法,错误时间间隔的变化主要与 ErrorBaseInterval 有关。本节实验参数 ErrorBaseInterval 取值为 20 ~ 200,步长为 20,同时为了观察变化趋势,还选取 200 ~ 500(步长为 100)以及 1 000 ~ 5 000(步长为 1 000)的点。其余参数均参照第 4.4 节实验,本节实验结果如图 5 所示。

由图 5(a)可以看出,3 种算法的调度成功率均随 ErrorBaseInterval 增大而增大,且从 $\text{ErrorBaseInterval} = 60$ 开始,DHFTS 算法表现最好,其次是文献[13]算法,DRFTS 算法最差,尤其在差别较大的 [80, 1 000] 区间中,相比文献[13]算法和 DRFTS 算法,DHFTS 算法平均分别高出 5.3% 和 26.0%。这说明在超过故障阈值时,DHFTS 算法通过切换到其他处理机重新处理任务来保证任务的调度成功率。

由图 5(b)可以看出,DRFTS 算法和文献[13]算法的平均执行时间受 ErrorBaseInterval 变化影响不大,在 $\text{ErrorBaseInterval} = 1\,000$ 前,DRFTS 算法的平均执行时间略高于文献[13]算法。而 DHFTS 算法受 ErrorBaseInterval 变化影响较大,尤其是在 $\text{ErrorBaseInterval} \leq 1\,000$ 时,其平均执行时间明显高于另外 2 种算法,最差情况下 ($\text{ErrorBaseInterval} \leq 100$),比另外 2 种算法的结果高出 2.1 倍左右。但是,虽然 DHFTS 算法的平均执行时间较长,但是该结果依然符合硬实时要求。

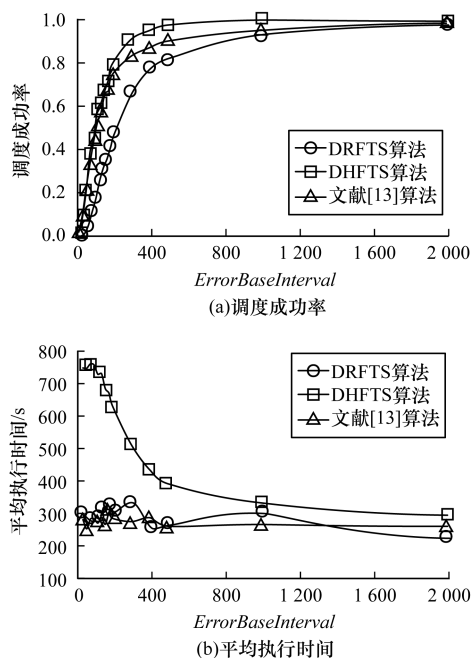


图 5 ErrorBaseInterval 对 3 种算法性能的影响

4.6 单处理机上容错次数对算法性能的影响

单处理机上的容错次数 $\text{FaultTolerantTimes}$ 是指触发反馈式切换的错误数量最大值^[20]。本节实验

的其他参数同第4.4节,实验结果如图6所示。由图6(a)和图6(b)可以看出,DHFTS算法性能指标均随 *FaultTolerantTimes* 增加而减小。造成该结果的原因是随着 *FaultTolerantTimes* 的增加,DHFTS算法会渐渐退化为DRFTS算法。且从图6(a)和图6(b)中还可以看到,在4以后的区间中,调度成功率和平均执行时间变化较小,在[4,10]区间内的变化幅度只占[1,4]区间内变化幅度的35.6%和15.4%,这说明在固定故障时间间隔条件下,较大的单处理器容错次数对于算法调度性能的影响较小。

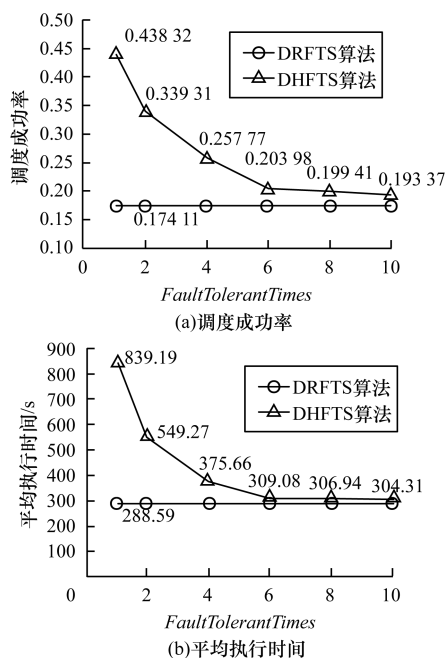


图6 *FaultTolerantTimes* 对2种算法性能的影响

5 结束语

本文借鉴拟态工控处理机架构,在动态异构模型中同时考虑表决、切换等因素的影响,针对硬实时的非周期性任务提出一种容错调度算法DHFTS。实验结果表明,与静态异构模型下的DRFTS算法以及文献[13]算法相比,DHFTS算法在考虑瞬时故障的硬实时条件下,能够获得更高的调度成功率,尤其在故障产生时间变化较明显的区间内,相比于2种同类算法,其调度成功率分别高出5.3%和26.0%左右。下一步将研究安全评估模型及其相应的开销,并分析动态异构模型下受安全资源限制的启发式调度算法。

参考文献

[1] 邬江兴. 网络空间拟态安全防御导论[M]. 北京: 科学出版社, 2017.
 [2] 魏帅, 于洪, 顾泽宇, 等. 面向工控领域的拟态安全处理机架构[J]. 信息安全学报, 2017, 2(1): 54-74.

[3] 张铮, 马博林, 邬江兴. Web服务器拟态防御原理验证系统测试与分析[J]. 信息安全学报, 2017, 2(1): 13-28.
 [4] GHOSH S, MELHEM R, MOSSÉ D. Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems[J]. IEEE Transactions on Parallel and Distributed Systems, 1997, 8(3): 272-284.
 [5] AL-OMARI R, SOMANI A K, MANIMARAN G. An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems[J]. Journal of Parallel and Distributed Computing, 2001, 65(5): 595-608.
 [6] 罗威, 阳富民, 庞丽萍, 等. 基于延迟主动副本的分布式实时容错调度算法[J]. 计算机研究与发展, 2007, 44(3): 521-528.
 [7] 朱晓敏, 祝江汉, 马满好. 异构集群系统中具有QoS需求的实时任务容错调度[J]. 软件学报, 2011, 22(7): 1440-1456.
 [8] 朱晓敏, 王建江, 王吉, 等. 面向容错的对地观测卫星实时任务调度研究[J]. 电子学报, 2015, 43(8): 1471-1480.
 [9] ZHU X, WANG J, GUO H, et al. Fault-tolerant scheduling for real-time scientific workflows with elastic resource provisioning in virtualized clouds[J]. IEEE Transactions on Parallel and Distributed Systems, 2016, 27(12): 3501-3517.
 [10] 马民, 金士尧. 三模容错多处理器动态实时调度算法[J]. 计算机工程与科学, 2007, 29(2): 76-78.
 [11] HATTON L. N-version design versus one good version[J]. IEEE Software, 1997, 14(6): 71-76.
 [12] MANIMARAN G, MURTHY C S R. A new scheduling approach supporting different fault-tolerant techniques for real-time multiprocessor systems[J]. Microprocessors and Microsystems, 1997, 21(3): 163-173.
 [13] 雷宇, 谷建华. 多处理机系统的一种实时容错调度算法[J]. 微处理机, 2009, 30(1): 110-112.
 [14] CHERNIGOVSKIY A S, TSAREV R Y, KNYAZKOV A N, et al. Hu's algorithm application for task scheduling in N-version software for satellite communications control systems[C]//Proceedings of International Siberian Conference on Control and Communications. Washington D. C., USA: IEEE Press, 2015: 1-4.
 [15] 全青, 张铮, 邬江兴. 基于软硬件多样性的主动防御技术[J]. 信息安全学报, 2017, 2(1): 1-12.
 [16] BUTTAZZO G C. Hard real-time computing systems: predictable scheduling algorithms and applications[M]. Berlin, Germany: Springer, 2008: 58-63.
 [17] SCOTT R K, GAULT J W, MCALLISTER D F. Fault-tolerant software reliability modeling[J]. IEEE Transactions on Software Engineering, 2006, 32(5): 582-592.
 [18] 丁万夫, 郭锐锋, 秦承刚, 等. 硬实时系统中基于软件容错模型的容错调度算法[J]. 计算机研究与发展, 2011, 48(4): 691-698.
 [19] 廖福蓉, 王成良, 陈蜀宇. 基于任务备份的云计算容错调度算法[J]. 计算机工程, 2012, 38(24): 17-20.
 [20] 曹凯, 文捷. 基于 $(k+2, k)$ MSR的多容错低修复带宽编码[J]. 计算机工程, 2018, 44(2): 84-87, 91.