分布式系统中基于主/副版本的实时容错调度综述

陈晗鸣，罗　威，李明辉
（中国舰船研究设计中心，武汉 430064）

摘　要：对分布式系统中基于主/副版本技术的实时容错调度算法进行了归纳和总结，从主/副版本执行的关系、任务的调度方式以及调度环境等各个方面深入分析和比较了近年来基于主/副版本的实时容错调度算法，并指出它们各自的优缺点和适应环境。最后指出了本研究领域的未来研究发展方向。

关键词：实时任务；容错；主版本/副版本；分布式系统
中图分类号：TP306　　**文献标志码**：A　　**文章编号**：1001-3695(2012)11-4017-06
doi:10.3969/j.issn.1001-3695.2012.11.004

Survey of primary/backup copy based real-time and fault-tolerant scheduling in distributed systems

CHEN Han-ming, LUO Wei, LI Ming-hui
(*China Ship Development & Design Center*, *Wuhan* 430064, *China*)

Abstract: This paper surveyed the update-to-date research and practice of primary-backup copy based real-time and fault-tolerant scheduling in distributed systems from several typical aspects and pointed out their strong and weak points. Finally, it discussed the potential research and development directions of this research area in the future.

Key words: real-time tasks; fault-tolerance; primary/backup copy; distributed systems

随着高速网络和高性能 PC/工作站的发展，实时系统越来越多地应用于各种分布式系统中，如飞行控制系统、核电站控制系统等。实时系统可分为软实时系统和硬实时系统。对于软实时系统来说，如果任务不满足响应期限时，只是系统性能退化，并不会导致灾难性的后果；而在硬实时系统中，系统的正确性不仅依赖于任务计算的逻辑结果，而且依赖于结果产生的时间。如果任务没有在期限前完成，往往会导致灾难性的后果[1]。

由于在现实中设计一种完全无错的系统是不可能的，因此必须采用一定的手段来提高系统的可靠性，这些手段可分为错误预防（fault prevention）、错误验证（validation）、错误预报（fault forecasting）和容错（fault tolerant）四种[2]。相对而言，容错是提高系统可靠性的现实而有效的方法。

传统的容错方法包括重试（re-execution）、N 版本程序设计（N-version programming）和恢复块（recovery block）技术等容错方法[3~5]。这些方法对于提高系统的可靠性，延长系统的使用寿命有着重要的应用价值。但是，这些传统容错方法存在两个问题：a）传统的容错方法没有考虑系统的实时性问题，因此不适用于现代的实时系统；b）这些方法在空间和时间上的开销都很大，并且代价昂贵。因此传统的容错方法尽管能在一定程度上提高系统的可靠性，但这些方法都不适用于现代的实时分布式系统。

为了给实时系统提供容错能力，Liestman 等人[6]首先提出了一种单处理机上基于主/副版本（primary/backup，PB）备份技术的容错调度算法。PB 备份技术是指在单处理机系统中，任务分为主（primary）任务和替代（alternate）任务。一个主任务拥有多个替代任务，而每个替代任务只能对应一个特定的容错处理，它们的计算时间各不相同，替代任务仅在主任务执行失败的时候启动执行。该算法能有效地保证任务（主任务或替代任务）在截止期限内完成。但是该算法仅处理周期任务，并且任务间的周期必须是倍数关系。另外，该算法对替代任务的执行时间也作了限制：替代任务的执行时间必须小于对应的主任务的执行时间。在 Liestman 的研究基础上，很多学者陆续提出了单处理机上基于主/副版本的实时容错调度算法[7~9]。

由于具有简单、易于实现的特点，主/副版本技术已经被广泛应用于各种实时分布式系统中。在这种技术中，同一个任务有主/副两个或多个版本。与单处理机不相同的是，它们分别运行在不同的处理机上，以保证在其中一个处理机失效的情况下，另一个处理机上的任务仍然能在截止期前产生正确的结果，从而达到实时容错的目的。

本文从副版本的执行方式、运行环境、任务之间的逻辑关系等方面对基于主副版本的实时容错调度算法进行分类分析和比较。

1　实时容错调度算法的分类

1.1　副版本的调度方式

在主/副版本技术中，同一个任务有主/副两个版本，它们分别运行在不同的处理机上，以保证在其中一个处理机失效的

收稿日期：2012-04-01；**修回日期**：2012-05-22

作者简介：陈晗鸣（1972-），男，湖北武汉人，高级工程师，博士，主要研究方向为实时容错调度算法、实时嵌入式操作系统等；罗威（1980-），男，湖北黄石人，博士，主要研究方向为实时容错调度算法、实时嵌入式操作系统等（free_xingezi@163.com）；李明辉（1989-），男，安徽宿州人，硕士研究生，主要研究方向为舰船信息系统、实时嵌入式操作系统等.

情况下,另一个处理机上的任务仍然能在截止期前产生正确的结果。从同一个任务的主/副版本在执行时间上的关系进行分类,副版本有三种执行方式。

1)主动方式副版本

主动方式的副版本的执行特点是:与主版本并行执行,两者在执行时间上相互重叠且不存在同步关系。因此主动方式的副版本的优点主要在于实现简单、对任务的执行时间没有约束要求,但是其缺点在于系统无故障时需要两倍于无容错需求时的处理机数。图 1 描述了主动副版本的执行方式,其中绿(浅)色为任务的主版本,红(深)色为任务的副版本,它们分别被调度到处理机 P_1 和 P_2 上,横坐标为调度的执行时间。文献[10,11]中提到了调度算法都采用了主动方式的副版本。

2)被动方式副版本

与主动方式副版本相对应的是被动方式副版本。如图 2 所示,绿(浅)色为任务的主版本,红(深)色为任务的副版本,它们分别被调度到处理机 P_1 和 P_2 上,任务的主/副版本在执行的时间上不相重叠,即副版本仅在主版本结束执行时间之后才开始执行。因此,被动方式的副版本在主版本任务执行成功的情况下无须执行(即 backup copy de-allocation 技术)。同时,在单处理机失效的假设条件下,对应主版本在不同处理机上的副版本可在执行时间上相互重叠(即 backup copy overlapping 技术),从而减少为实现容错而需要的冗余。相对应地,被动方式副版本的缺点在于主/副版本之间需要一定的同步开销。总体来说,从提高系统可调度性和资源利用率的角度出发,希望更多任务的副版本以被动方式运行。文献[2,12,13]中提到的调度算法都采用了被动方式的副版本。

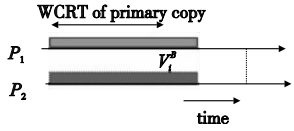

图1 主动副版本示意图

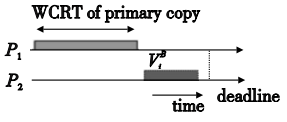

图2 被动副版本示意图

3)主/副版本间的重叠方式

如图 3 所示,这种方式的副版本执行方式介于主动方式和被动方式之间,主/副版本在执行时间上存在重叠,即这种方式的副版本既能够利用主动方式副版本无执行时间约束的优点,也具有被动方式副版本的高效性。可以看出,主/副版本间的重叠方式是主动副版本和被动副版本执行方式之间的折中[13~15]。文献[14]提出的一种适应性的实时容错调度算法,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中。文献[13]提出一种延迟调度副版本策略的技术,对主动副版本尽量延迟调度,并通过在主版本成功执行后终止副版本执行来减少系统的冗余,从而提高系统的可调度性。文献[13]提出的技术是基于不可抢占性的非周期任务。受到文献[13]的启发,文献[15]提出一种基于抢占的周期性任务的实时容错调度算法 DABCBF,它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。同时,提出一种基于该技术的以最小化处理机数目为优化目标的启发式任务分配算法,即基于延迟主动副版本的最佳适应算法。

1.2 版本间重叠技术

主/副版本间重叠方式是考虑同一任务主副版本间的关系,而本节则考虑不同任务主副版本间的重叠关系。从这个角度进行分析,可以把版本间的重叠技术分为两大类。

1)副版本间的重叠技术

副版本间的重叠技术(B/B-overlapping)定义为在单处理机失效的假设条件下,对应主版本在不同处理机上的副版本可在执行时间上相互重叠。如图 4 所示,分别将主版本任务 Pr1 和 Pr2 调度到处理机 processor1 和 processor3 上,Pr1 和 Pr2 对应的副版本任务 Bk1 和 Bk2 都被调度到处理机 processor2 上,那么 Bk1 和 Bk2 在执行时间上可以相互重叠。

显然,副版本间的重叠技术可以更加有效地利用处理机的效率,从而提高系统的可调度性。

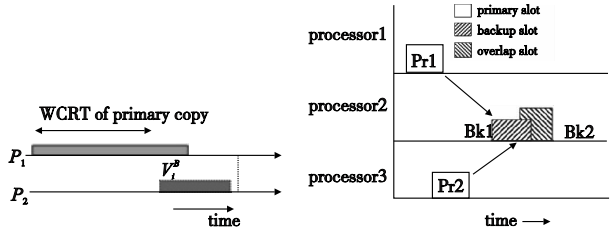

图3 P/B重叠方式的副版本示意图 图4 副版本间重叠方式示意图

实现副版本间重叠技术的关键在于单处理机假设。其原理是如果某一个处理机失效,那么其他处理机肯定不会失效,因此对应的副版本都可以被撤销执行。这样,失效处理机上主版本对应的副版本就能够与其他副版本无冲突地执行。但是在现实应用中,特别是对于大规模的分布式系统,单处理机失效的假设太严格和理想化了,为了克服这个问题,很多学者提出基于聚合模式的副版本间的重叠技术。具体可以分为以下几种类型:

a)无聚合模式。这种模式中,对于有 N 个处理机的分布式系统,副版本有($N-1$)种调度选择[13]。因此,在一个时间片内,最多可以有 $N-1$ 个副版本进行相互重叠。尽管这种调度模式能够获得很高的可调度性,但是它仅能容忍一个处理机故障,这样的情况对于系统较大的分布式系统来说,显然这种假设要求太严格和理想化了。

b)静态聚合模式。在这种模式中,系统中的处理机被静态地分配成不同的逻辑组(logical group),主/副版本都在同一个逻辑组中的副版本可以相互重叠调度执行[16]。假设一个分布式系统被分为 g 个逻辑组而每一个组内都有 K 个处理机,那么在同一个逻辑组内,尽管同一个时间片内最多只能有($K-1$)个副版本相互重叠执行,但是整个分布式系统最多可同时容忍 g 个处理机故障,因此提高了系统的可靠性。换句话说,静态聚合模式可以在系统的可调度性和容错能力之间进行折中。

c)动态聚合模式。为了同时充分利用无聚合模式和静态聚合模式的优点并且提高聚合模式的使用效率,文献[17]提出了一种动态聚合模式,该聚合模式动态地把处理机分为不同的逻辑组。当任务到来时,动态地创建或扩展逻辑组;当任务执行完毕时,动态地删除或减少逻辑组的大小,并且在静态聚合模式中,一个处理机只能属于一个逻辑组,而在动态聚合模式中,一个处理机能属于多个逻辑组。文献[17]中的仿真实验说明,动态聚合模式能够提供比静态聚合模式更优的可调度性,但是在可靠性上不亚于静态聚合模式。

2)主副版本间的重叠技术

与前面提出的主副版本间重叠方式不同的是,这里的主副版本间的重叠技术指的是一个任务的主版本与另外任务的副

版本相互重叠。

文献[18]提出了一种主副版本间的重叠技术。如图 5 所示, 主版本任务 Pr1 被分配到处理机 processor1 上, 它的副版本 Bk1 被分配到处理机 processor2 上。另一个任务的主版本 Pr2 与 Bk1 在同一个处理机上,并且调度时间重叠,它的副版本 Bk2 被分配到 processor3 上。在这种模式下,如果无处理机故障, Pr1 和 Pr2 都正常执行, 如果处理机 processor1 故障, Bk1 和 Bk2 都会被调度执行。因此,与 B/B-overloading 技术相比 P/B-overloading 能够提前调度任务的主版本,从而提高可调度性。

1.3　静态调度与动态调度

与实时调度算法一样,实时容错调度算法根据实时任务的分派和可调度性测试的时机可分为三类,即静态任务调度、动态任务调度和静/动态混合任务的调度。这里任务的分派是指选择某一个任务到某一处理机上调度的过程;而任务的可调度性测试指的是确定任务是否可以在某一个系统上调度的过程。

表 1 描述了实时任务的调度分类。其中,(a)表示所有任务的调度和分派都是离线的,在设计阶段进行的,任务的分派实际就是执行严格的时间表,这种调度的优点在于可预测性好。但是由于这种调度是基于简单的调度模型和严格的时间约束,在实际的使用中是脆弱和缺乏灵活性的。为了克服这种不灵活性,可以在任务进行实际分派的时候再检测任务的可调度性,(c)描述了这种调度方法。(d)描述的调度方法其可调度性分析和任务分派都是在线进行的。事实上,这样的调度分析成了一种接受测试,测试新到来的任务的截止期是否会被错过,如果会被错过,则被拒绝调度;否则就进入系统进行调度。显然,这种调度方法是最具有灵活性的,但是在系统负载很重的情况下性能会很差,因此可预测性也很差。最合理的调度方式是(b)中描述的调度方法,目前来说基于优先级的调度方式是最能被接受的。

表 1　实时调度的分类

可调度性分析 (schedulability analysis)	任务的分派	
	离线(off-line)	在线(on-line)
离线(off-line)	(a)可调度性能保证;有严格的时间约束假设;不灵活	(b)可调度性能保证;松散的时间约束假设;合理性的灵活
在线(on-line)	(c)可调度性能保证;不是很灵活	(d)可调度性无法保证;很灵活

基于以上对实时任务调度的分类,本文对实时容错的调度方法分类如下:

a)静态调度。静态调度方法中,任务的分配都是离线进行的,即在实时任务正式在处理机上调度执行前,先把任务在处理机上的分配和调度时间安排好,在任务正式开始执行后按照预先的调度方案执行。这种调度方法主要用于周期任务的调度,它的优点在于能够预先安排好调动,减少任务调度过程中的开销;而缺点在于缺乏灵活性,在实际的调度中不能够及时地根据系统资源和任务的执行情况进行及时的调整。文献[2,9~11,15,19~24]提出的算法都是基于静态调度算法的。

b)动态调度。与静态调度方法相对应的是动态调度。在实时系统中,很多任务并非都以周期方式在处理机上进行调度,更多任务——特别是非周期任务——都是随机到达系统并动态调度执行的。在动态调度方法中,任务的分派和可调度性测试都是在系统运行时在线进行的。这种情况下,可调度性测试实际上变成了一种接受测试(acceptance test),测试动态到达任务的截止期是否会被保证,如果无法保证任务的截止期,任务将被拒绝调度。可以看出,动态调度与静态调度相比有更好的灵活性,然后由于可调度性测试需要在线进行,它的调度算法的复杂度不能太高,并且由于无法保证是否可以被调度,算法的可预测性(predictability)很差。文献[12~14,16~18]提出的算法都是基于动态调度算法的。

c)混合任务的调度。在实时系统中,周期任务和非周期任务往往共同存在。因此,响应的实时容错调度算法必须同时保证周期任务和非周期任务的容错特性,但是目前关于这方面的研究著述还不多。文献[13]提出一种基于混合实时任务的实时容错调度算法,该算法在保证周期任务的容错能力的同时,尽量减少非周期任务的响应时间,然而该算法没有考虑非周期任务的容错特性。文献[19]提出一种基于资源预留策略的分布式容错调度算法,该算法利用周期任务调度后的剩余时间片来调度非周期任务。由于非周期任务的调度时间和完成时间无法计算,因此对非周期任务都采用主动副版本来实现容错。

1.4　同构系统与异构系统

从分布式系统的调度环境来看,分布式系统可分为同构系统和异构系统。相应地,实时容错调度算法也可分为基于同构分布式系统的调度算法和基于异构分布式系统的调度算法。

a)基于同构分布式系统的调度算法。在同构分布式系统中,每个处理机完全相同,都有着相同的处理能力。目前大量基于分布式系统的实时容错调度算法都是基于同构平台的,如文献[2,10~18]提出的调度算法都是基于同构分布式系统。

b)基于异构分布式系统的调度算法。与同构分布式系统不同的是,异构系统中的每个处理机的运算能力、可靠性等都有很大差别,因此异构环境中的调动算法必须考虑这些综合因素。秦啸等人[20~27]深入研究了基于分布式系统的实时容错调度算法。

1.5　任务之间的约束和同步关系

在实时容错系统中,被调度的实时任务之间可能存在约束关系,总体而言,存在两类约束关系:

a)任务之间通过消息进行同步。对于非周期任务,这样的约束一般表现为一个 DAG 图。如文献[23,24]提出的实时容错调度算法的任务模型都是基于 DAG 图。图 6 描述了一个实时任务集在一组多处理机上执行关系的 DAG 图。方框中的第一项表示该任务在三个处理机上的执行时间;方框中的第二项表示任务的截止期(deadline);方框中的箭头表示了任务之间的执行逻辑关系;箭头上的数字表示了处理机之间的通信延迟。

b)任务之间由于资源共享而存在约束关系。例如文献[11]提出了一种基于资源共享的实时容错调度算法,证明了在分布式嵌入式系统中,基于多处理机的临界资源不可抢占协议 MNCSP 的阻塞时间少于资源封顶协议 PCP,并在此基础上提出了基于 MNCSP 的实时容错调度算法。

1.6　被动副版本的进程调度模型

被动副版本的进程调度模型大致可分为主动进程复制模型(active process replica)[2,12,13,28]和被动进程复制模型(passive process replica)[29~32]。

a) 主动进程复制模型。在这种模型中,主动和被动被假设有相同的负载或相同的执行时间,目前大部分的副版本模型都是基于主动进程复制模型的[2,12,13,28]。

b) 被动进程复制模型。事实上,在主/副版本进程模式中,同一时间内只有主版本的进程处于激活状态,即运行状态;而备份进程处于非激活状态,即休眠状态。为了保证节点发生故障时备份进程有足够的信息用以接管主进程的工作,主进程定期地产生进程检查点(check point)并发送给备份进程。备份进程此时处于两种可能的状态,即等待接收主进程的检查点,或者正在保存已经收到的主进程检查点。因此,备份进程的负载应该远小于主进程的负载,这种容错特性的进程调度模型称为被动进程复制模型。被动进程模型更符合实际的系统。例如,在 Paralex 系统中通过延迟绑定技术实现了被动进程复制模型下进程动态分配算法[32]。文献[29~31]提出的副版本进程模式都是被动进程复制模型。

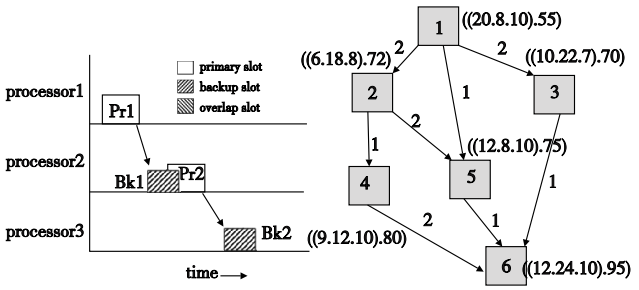

图5 主/副版本间重叠方式示意图 图6 基于DAG的任务模型

1.7 基于价值的实时容错调度模式

在一些动态调度系统中,当一个实时任务集到达系统中时,调度算法确定在不损害现有已调度的实时任务的可调度性的前体条件下,确定新的调度任务是否可被调度。在系统负载不过重(not overloaded)的前提下,完全可以保证所有任务的实时性。但是在系统过载(overloaded)的情况下,不可能所有实时任务的截止期都能得到保证,这就意味着不少任务会因为被拒绝调度而错过截止期。在这样的过载系统中,希望通过基于价值的实时调度模式(value-based scheduling)来获得最大的价值(即系统的总体利用率)[33]。

在这种调度模式中,如果实时任务的截止期能够被保证,每个实时任务会获得一定的价值,否则将遭到一定的惩罚。因此,基于价值的实时调度模式的本质就是一个决策问题,即选择一定的调度和决策方案,让调度最终获得的价值最大。选择合适的目标价值函数(value function)是实现基于价值的实时调度的关键所在。典型的价值函数有文献[10]提出的性能指标(performance index)和文献[34]提出的价值密度(value-density),这两个价值函数从不同的角度衡量了最终的性能值,两者的区别是前者强调容错能力的重要性,而后者强调在一段时间内获得的总价值。

1.8 调度目标

主副版本技术是实现实时分布式系统容错能力的重要机制。然而在不同的调度环境中,各种调度算法要实现的调度目标是不同的。如在分布式嵌入式系统中,由于空间和重量的约束,减少调度所需的处理机的个数是主要需要考虑的因素;而在大规模的动态分布式系统中,处理机和系统资源充足,实现分布式系统中各个处理机的负载均衡则是首要目标。本节从实时容错调度目标的角度来对现有的实时容错调度算法的调度目标进行分类综述。

1) 以减少调度所需要处理机的个数为调度目标 如前所述,实现容错的主要手段就是冗余,主/副版本的技术在本质上也是一种冗余。然而对于很多系统,如航天控制系统和导弹导航系统,过多的冗余将增加调度所需的处理机的个数,会导致能耗和体积的增加,而这点恰恰是这类系统不可接受的。因此,大量的实时容错调度算法都以减少调度所需处理机的个数为调度目标。为了实现这个目标,不同的算法采用了以下不同手段:

a) 使得更多任务的副版本以被动方式执行。文献[35]对传统的 RMFF(rate-monotonic first-fit)算法进行容错扩充为 FTRMF(fault-tolerant rate-monotonic first-fit),使每一个实时任务都有对应的副版本,并根据主版本的最坏响应时间 WCRT(worst-case response time)来确定副版本的执行类型(主动副版本或被动副版本),最后通过仿真实验证明,与传统的双备份调度算法相比,FTRMF 能在牺牲系统容错能力的前题条件下,有效地降低调度所需要的处理机间的个数。在 FTRMFF 的基础上,Yang 等人[29]提出一种基于最佳适应策略的实时容错调度算法 FTRMBF(fault-tolerant rate-monotonic best-fit)。与 FTRMF 不同的是,FTRMBF 在进行任务搜索时搜索所有处理机,使得尽量多的任务副版本以被动方式执行以提高系统的可调度性。相对于 FTRMFF,尽管 FTRMBF 在一定程度上提高了可调动性,然而 FTRMFF 和 FTRMBF 算法在进行任务分配时仅设置一个处理机,这样限制了任务的搜索空间,特别是对于初始任务。为了解决这个问题,文献[27]提出一个基于多处理机系统的高效实时容错调度算法(ERMBF)以提高系统的可调度性。现有调度算法(FTRMBF 和 FTRMFF)在初始调度时,仅分配单个处理机,而 ERMBF 在进行任务分配之前,先分配一定数量的处理机;这样增加了任务的搜索空间,特别是对于初始分配的任务。另一方面,在分配一个新的处理机时,对现有的任务进行重新分配以寻找一个更好的调度配置。这两种策略都以更多的任务以被动方式执行为调动目标,因此,ERMBF 能在不损失系统的实时性和容错能力的前提下节约处理机的个数。仿真实验表明,相对于现有文献中可比较的算法,ERMBF 能显著地提高系统的可调度性。

b) 减少主版本的冗余。针对 FTRMFF 算法,为了进一步减少冗余、提高系统的可调度性,文献[15]提出一种基于抢占的周期性任务的实时容错调度算法 DABCBF。DABCBF 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;同时,提出一种基于该技术的以最小化处理机数目为优化目标的启发式任务分配算法,即基于延迟主动副版本的最佳适应算法。

2) 以处理机的负载均衡为调度目标 Bannister 等人[31]首先研究了将进程的负载均匀地分布到各个节点上的负载均衡容错调度算法。然而该算法是基于主动进程复制的假设的,不符合实际需要。Kim 等人[36]所提出的两阶段算法是一个经典的被动进程复制模型下基于负载均衡的静态分配算法,该算法解决了在故障发生前后同时保证处理机的负载均衡问题。Lee 等人[28]在此基础上将算法扩展到允许多个备份进程失效的情况。然而以上所研究的多机系统或分布式系统均属于同构模型。最近,郭辉等人[30]提出了基于主/从版本的具有容错功能的进程调度算法 HDALF 和 HDLDF,分别给出了两种算法的时

间复杂度,并对算法的负载均衡性和节点资源利用率进行了讨论。与以往容错调度算法不同的是,该算法是在被动进程复制模式下、适合异构分布式系统的容错调度算法,而以往的研究都是建立在主从版本进程有相等的负载或执行时间相同的模型基础上,或者仅适合同构分布式系统。实验结果表明,HDALF 和 HDLDF 算法的性能比基于同构分布式模型下的两阶段算法更加优越,并且得出了这样的结果:当系统发生故障前后的负载均衡性权值相等时,在负载均衡和处理机资源利用率方面,HDLDF 算法都要优于 HDALF 算法。

3)以系统的可靠性为调度目标　以系统可靠性为调度目标的算法主要适用于异构分布式系统。与同构分布式系统不同的是,异构系统中的每个处理机的运算能力、可靠性等都有很大差别,因此异构环境中的调动算法必须考虑可靠性的因素。Srinivasan 等人[32]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。在此基础上,庞丽萍、秦啸等人[22~25]深入研究了基于异构分布式系统的实时容错调度算法,这些算法采用了可靠性代价的概念,并且这些调度算法都以尽量减少可靠性代价从而增强系统的可靠性为调度目标来调度实时任务。然而,这些可靠性模型都基于不可抢占的非周期任务集,不适用于可抢占性的实时、周期任务。为了克服上述问题,文献[26,37]对异构分布式系统中的可抢占性的实时、周期任务进行建模,并提出了响应的调度算法。文献[26,37]中提出的可靠性模型的主要不同在于:文献[26]中提出的可靠性模型没有考虑处理机失效后的情况;而文献[37]中的可靠性模型考虑了在单处理机失效的情况下,系统可靠性的建模和分析。

4)以节能为调度目标　在很多分布式嵌入式操作系统中,特别是以电池为主要供电手段的系统中,节能是首先需要考虑的问题。Unsal 等人[33]提出了一种基于主副版本模式的调度算法,该算法通过推迟副版本间重叠长度来减少主副版本间的重叠,从而减少电源的消耗。目前关于这方面的研究和著述还不多,因此这方面是下一步值得研究的领域之一[43]。

5)以提高系统的任务接收率为调度目标　在动态实时任务的调度模型中,任务的接收率 *GR*(guaranteed ratio)是衡量系统可调度性的重要指标,由以下公式计算的到:

$$GR = \frac{\text{被调度算法接受的总任务个数}}{\text{到达系统任务的总数}} \times 100\% \tag{1}$$

文献[12,14,16~18]提出的算法都是以提高任务的接收率为调度目标的。

6)以提高综合性能指标为调度目标　在很多环境中,很多调度目标都是相互冲突的,因此调度算法实质就是在这些调度目标间进行折中选择。为了恰当地衡量调度的综合性能,需要采用一些综合性能指标。如文献[23]中为了综合衡量调度算法的可靠性代价(reliability costs)和可调度性(schedulability),提出了综合指标 performability,定义为

$$\text{performability} = \text{reliability costs} \times \text{schedulability} \tag{2}$$

这里的可靠性代价定义同文献[22~25]提到的可靠性代价,而 schedulability 定义为任务的接收率 *GR*。文献[14]提出一种在多处理机系统中针对软实时任务的适应性容错调度模式。该文献也提出了一种综合性能指标 *SR*,也是在系统的可调度性和输出性能之间进行折中。这里的可靠性与文献[28]中定义的不同,其实质上定义为任务成功完成时对处理机的利用率 processor utility;而输出性能 value 则与任务实际完成时的最坏响应时间相关,如果任务完成时的最坏响应时间越长,输出性能越差,否则越好。显然 value 和 processor utility 是相互冲突的性能指标。*SR* 的提出就是为了综合这两种指标。SR_i 定义为一个单个任务 i 的 *SR*,即

$$SR_i = \text{utility} \times \text{valuve} \tag{3}$$

而 n 个实时任务的综合 *SR* 定义为 n 个任务 SR_i 的平均值:

$$SR = \sum_{i=1}^{n} SR_i / n \tag{4}$$

2　结束语

随着高速网络和高性能计算机的发展,分布式系统已经广泛地应用于各种实时系统,这些系统中的任务都有严格的时间约束(期限),如果任务没有在期限前完成,往往会导致灾难性的后果。为了保证系统中的实时任务即使在系统出现故障后仍能在期限前完成,必须为系统提供一定的容错能力,而主/副版本备份技术是分布式系统常用的实时容错调度方法。本文对分布式系统的基于主/副版本备份技术的实时容错调度进行了归纳和总结,从主/副版本执行的关系、任务的调度方式以及调度环境等各个方面深入分析并比较了近年来基于主/副版本备份技术的实时容错调度的技术并指出他们各自的优缺点。

笔者认为,未来分布式系统基于主/副版本的实时容错调度研究发展方向主要有:

a)主/副版本调度方法的根本思想是通过冗余来实现容错,但是引入冗余后不可避免地会对系统的可调度性和资源利用率造成影响。因此从副版本的机制本身和任务的调度算法两个方面来进一步减少冗余和提高资源利用率是下一步研究的重点。

b)现有的研究成果都仅仅考虑单一的主/副版本容错机制,这样单一的方法在实践应用中存在很大的局限性。研究一种容错调度模型,该模型能同时考虑多种容错机制,如主/副版本调度和 check point 方法等,能从理论和实际应用中大幅提高实时分布式系统的可靠性。

c)现有分布式系统的主副版本间的容错机制大多仅考虑实时任务的容错能力,没有考虑任务间通信的容错能力和可靠性,因此有必要对实时任务和实时任务间的通信机制进行整合考虑,从而从根本上提高分布式系统的可靠性。

d)节能的问题。随着分布式嵌入式系统的发展,在获得系统高性能的同时尽量减少电源消耗是重要的调度目标之一。因此在设计实时容错调度算法的同时充分考虑电源消耗是未来最有价值的研究方向之一。

参考文献:

[1] LIU C L, LAYLANG J W. Scheduling algorithm for multiprogramming in a hard-real-time environment[J]. Journal of the ACM, 1973, 20(1):40-61.

[2] LAPRIE J C. Dependable computing and fault-tolerant systems: basic concepts and terminology[M]. London: Springer-Verlag, 1992.

[3] IYER R K, ROSSETTI D J. A measurement-based model for workload dependence of CPU errors[J]. IEEE Trans on Computers, 1986, C-35(6):511-519.

[4] CHEN Li-ming, AVIZIENIS A. N-version programming: a fault tolerance approach to reliability of software operation[C]//Proc of the 8th Annual International Symposium on Fault Tolerant Computing. 1978: 3-9.

[5] HORNING J, LAUER H, MELLIAR-SMITH P, *et al*. A program structure for error detection and recovery[C]//Lecture Notes in Computer Science. 1974:177-193.

[6] LIESTMAN L, CAMPBELL R H. A fault-tolerant scheduling problem [J]. IEEE Trans on Software Engineering, 1986, 12(11): 1089-1095.

[7] 李俊,阳富民,卢炎生. 一种可行的容错实时系统可调度性分析[J]. 软件学报,2005,16(8):1513-1522.

[8] De LIMA G A, BURNS A. An optimal fixed-priority assignment algorithm for supporting fault-tolerant hard real-time systems[J]. IEEE Trans on Computers, 2003, 52(10): 1332-1346.

[9] KANDASMY N, HAYES J P, MURRAY B T. Tolerating transient faults in statically scheduled safety-critical embedded systems[C]// Proc of the 18th IEEE Symposium on Reliable Distributed Systems. Washington DC: IEEE Computer Society, 1999: 212-221.

[10] WANG Fu-xing, RAMAMRITHAM K, STANKOVIC J A. Determining redundancy levels for fault tolerant real-time systems[J]. IEEE Trans on Computers, 1995, 44(2): 292-301.

[11] YANG Chun-hua, DECONINCK G, GUI Wei-hua. Fault-tolerant scheduling for real-time embedded control systems[J]. Journal of Computer Science and Technology, 2004, 19(2): 191-202.

[12] 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.

[13] 张拥军,张怡,彭宇行,等. 一种基于多处理机的容错实时任务调度算法[J]. 计算机研究与发展,2000,37(4):425-429

[14] YANG Chun-hua, DECONINCK G. A fault-tolerant reservation-based strategy for scheduling aperiodic tasks in multiprocessor systems [C]//Proc of the 10th Euromicro Workshop on Parallel, Distributed and Network-based Processing. Washington DC: IEEE Computer Society, 2002: 319-326.

[15] 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, 2005, 65(5): 595-608.

[16] TSUCHIYU T, KAKUDA Y, KIKUNO T. A new fault-tolerant scheduling technique for real-time multiprocessors systems[C]//Proc of the 2nd International Workshop on Real-Time Computing Systems and Applications. Washington DC: IEEE Computer Society, 1995: 197-202.

[17] 罗威,阳富民,庞丽萍. 基于延迟主动副版本的分布式实时容错调度算法[J]. 计算机研究与发展,2007,37(4):425-429.

[18] GHOSH S, MELHEM R, MOSSE D. Fault-tolerance through scheduling of aperiodic tasks in hard real-time multiprocessor systems[J]. IEEE Trans on Parallel and System, 1997, 8(3): 272-284.

[19] AL-OMARI R, SOMANI A K, MANIMARAN G. A new fault-tolerant technique for improving the schedulability in multiprocessor real-time systems[C]//Proc of the 15th International Parallel and Distri-buted Processing Symposium. Washington DC: IEEE Computer Society, 2001: 32-39.

[20] 秦啸,韩宗芬,庞丽萍. 基于异构分布式系统的实时容错调度算法[J]. 计算机学报,2002,25(1):49-56.

[21] 秦啸,韩宗芬,庞丽萍,等. 混合型实时容错调度算法的设计和分析[J]. 软件学报,2000,11(5):686-693.

[22] 庞丽萍,秦啸. 一个基于分布式系统的实时容错调度算法[J]. 小型微型计算机系统,2000,21(5):686-693.

[23] QIN Xiao, HAN Zong-fen, JIN Hai, *et al*. Real-time fault-tolerant scheduling in heterogeneous distributed systems[C]//Proc of International Workshop on Parallel and Distributed Processing Techniques and Applications. 2000: 26-29.

[24] QIN Xiao, JIANG Hong, XIE Chang-sheng, *et al*. Reliability-driven scheduling for real-time tasks with precedence constraints in heterogeneous distributed systems[C]//Proc of the 12th Internationall Conference on Parallel and Distributed Computing Systems. 2000: 617-623.

[25] 韩宗芬,秦啸,庞丽萍,等. 分布式系统的实时容错任务调度算法设计[J]. 华中理工大学学报,1999,27(4):12-14.

[26] QIN Xiao, JIANG Hong, SWANSON D R. An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems[C]//Proc of the 31st International Conference on Parallel Processing. 2002: 360-368.

[27] QIN Xiao, JIANG Hong. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems [J]. Parallel Computing, 2006, 32(5-6): 331-356.

[28] LEE H, KIM J, HONG S. Fault-tolerant process allocation with load balancing[C]//Proc of Pacific Rim International Symposium on Fault-Tolerant Systems. 1995: 124-129.

[29] YANG Chun-hua, JI Li, SHENG De-yao, *et al*. Fault-tolerant scheduling based on the BEST-FIT heuristics for real-time multiprocessor systems[J]. Computer Engineer and Science, 2003, 25(5): 61-64.

[30] 郭辉,王智广,周敬利,异构分布式系统中基于负载均衡的容错调度算法[J]. 计算机学报,2005,28(11):1807-1816.

[31] BANNISTER J A, TRIVEDI K S. Task allocation in fault-tolerant distributed systems[J]. Acta Informatica, 1983, 20(3): 261-281.

[32] SRINIVASAN S, JHA N K. Safety and reliability driven tasks allocation in distributed systems[J]. IEEE Trans on Parallel and Distributed Systems, 1999, 10(3): 238-251.

[33] UNSA O S, LKOREN I, KRISHNA C M. Toward energy-aware software-based fault tolerance in real-time systems[C]//Proc of International Symposium on Low Power Electronics Design. 2002: 124-129.

[34] DAVOLI R, GIACHINI L, *et al*. Parallel computing in networks of workstations with parallex[J]. IEEE Trans on Parallel and Distributed Systems, 1996, 7(4): 371-384.

[35] LUO Wei, QIN Xiao, BELLAM K. Reliability-driven scheduling of periodic tasks in heterogeneous real-time systems[C]//Proc of the 21st IEEE International Conference on Advanced Information Networking and Applications. 2007: 778-783.

[36] KIM J, LEE S H, LEE S. Process allocation for load distribution in fault-tolerant multicomputers[C]//Proc of the 25th IEEE International Symposium on Fault-Tolerant Computing. Washington DC: IEEE Computer Society, 1995: 124-129.

[37] LUO Wei, YANG Fu-min, PANG Li-ping. Fault-tolerant scheduling for periodic tasks in heterogeneous systems[C]//Proc of the 3rd International Conference on Autonomic and Trusted Computing. 2006: 571-580.

[38] MANIMARAN G, MURTHY C S R. A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis [J]. IEEE Trans on Parallel and Systems, 1998, 9(11): 1137-1152.

(下转第 4027 页)

[10] FREEMAN T, KEAHEY K. Flying low: simple leases with workspace pilot[C]//Proc of the 14th International Euro-Par Conference on Parallel Processing. Berlin:Springer-Verlag,2008:499-509.

[11] SOTOMAYOR B, KEAHEY K, FOSTER L, *et al*. Enabling cost-effective resource leases with virtual machines[C]//Proc of IEEE International Symposium on High Performance Distributed Computing. 2007:237-240.

[12] HE Xiao-shan,SUM Xian-he, Von LASEEWSKI G. QoS guided min-min heuristic for grid task scheduling[J]. Journal of Computer Science and Technology,2003,18(4):442-451.

[13] CHANHAN S S,JOSHI R C. A heuristic for QoS based independent task scheduling in grid environment[C]//Proc of International Conference on Industrial and Information Systems. 2010:102-106.

[14] XU Meng,CUI Li-zhen,WANG Hai-yang,*et al*. A multiple QoS constrained scheduling strategy of multiple workflows for cloud computing [C]//Proc of IEEE International Symposium on Parallel and Distributed Processing with Applications. 2009:629-634.

[15] HAO Fang, LAKSHMAN T V, MUKHERJEE S, *et al*. Enhancing dynamic cloud-based services using network virtualization[J]. Compu-ter Communication Review,2010,40(1):67-74.

[16] 孙大为,常桂然,李凤云,等. 一种基于免疫克隆的偏好多维 QoS 云资源调度优化算法[J]. 电子学报,2011,39(8):1824-1830.

[17] BUYYA R, YEO C S,VENUGOPAL S. Market-oriented cloud computing: vision, hype, and reality for delivering it services as computing utilities[C]//Proc of the 10th IEEE International Conference on High Performance Computing and Communications. 2008:5-13.

[18] DEELMAN E, SINGH G, LIVNY M, *et al*. The cost of doing science on the cloud: the Montage example[C]//Proc of ACM/IEEE Conference on Supercomputing. Piscataway:IEEE Press,2008:1-12.

[19] De ASSUNCAO M D, Di COSTANZO A,BUYYA R. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters [C]//Proc of the 18th ACM International Symposium on High Performance Distributed Computing. New York: ACM Press,2009:141-150.

[20] YIGITBASI N,IOSUP A,EPEMA D. C-Meter: a framework for performance analysis of computing clouds[C]//Proc of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. Washington DC:IEEE Computer Society,2009:472-477.

[21] BRANDT J,GENTILE A,MAYO J,*et al*. Resource monitoring and management with OVIS to enable HPC in cloud computing environments[C]//Proc of the 23rd IEEE International Parallel and Distributed Processing Symposium. Washington DC:IEEE Computer Society. 2009:1-8.

[22] BUYYAA R, YEOA C S, VENUGOPAL S, *et al*. Cloud computing and emerging IT platforms:vision,hype,and reality for delivering computing as the 5th utility[J]. Future Generation Computer Systems,2009,25(8):599-616.

[23] 葛新,陈华平,杜冰,等. 基于云计算集群扩展中的调度策略研究[J]. 计算机应用研究,2011,28(3):995-997.

[24] 尹红军,李京,宋浒. 云计算中运营商效益最优的资源分配机制[J]. 华中科技大学学报:自然科学版,2010,39(1):51-55.

[25] BARROSO L A, HOLZLE U. The datacenter as a computer: an introduction to the design of war ehouse-scale machines[M]. San Rafael:Morgan & Claypool Publishers,2009.

[26] PRODAN R, WIECZOREK M, FARD H M. Double auction-based scheduling of scientific applications in distributed grid and cloud environments[J]. Journal Grid Computing,2011,9(4):531-548

[27] MEZMAZ M, MELAB N,KESSACI Y, *et al*. A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems[J]. Parallel Distribute Computing, 2011, 71 (1): 1497-1508.

[28] BOSS G,MALLADI P,QUAN D,*et al*. Cloud computing, IBM white paper[R/OL]. (2007-06-25). http://grid. lzu. edu. cn/resource/article_download. jsp? id = 33.

[29] CHEUNG W K,LIU Ji-ming,TSANG K H,*et al*. Wong. Dynamic resource selection for service composition in the grid[C]//Proc of IEEE/WIC/ACM International Conference on Web Intelligence. 2004:412-418.

[30] OpenNebula project [EB/OL]. (2008-12-23). http://www. openNebula. org/doku. php.

[31] 张建勋,古志民,郑超. 云计算研究进展综述[J]. 计算机应用研究,2010,27(2):429-433.

[32] SOTOMAYOR B, MONTERO R S, LIORENTE I M, *et al*. Virtual infrastructure management in private and hybrid clouds[J]. IEEE Internet Computing,2009,12(5):14-22.

[33] SOTOMAYOR B, MONTERO R S, LLORENTE I M, *et al*. Virtual infrastructure management in private and hybrid clouds[J]. IEEE Internet Computing,2009,13(5):14-22.

[34] Eucalyptus project [EB/OL]. (2008-02-12). http://eucalyptus. com.

[35] NURMI D, WOLSKI R, GRZEGDRCZYK C, *et al*. The Eucalyptus open source cloud computing system[C]//Proc of the 9th IEEE/ACM Internatioanl Symposium on Cluster Computing and the Grid. Washington DC:IEEE Computer Society. 2009:124-131.

[36] TIAN Wen-hong. Adaptive dimensioning of cloud data centers[C]//Proc of the 8th IEEE International Conference on Dependable, Autonomic and Secure Computing. 2009:5-10.

(上接第 4022 页)

[39] YANG Fu-min, LUO Wei, PANG Li-ping. An efficient real-time fault-tolerant scheduling algorithm based on multiprocessor systems [J]. Wuhan University Journal of Nature Science,2006,6(11): 219-223.

[40] LUO Wei, YANG Fu-min, Tu Gang. DYFARS:boosting reliability in fault-tolerant heterogeneous systems through dynamic scheduling [C]//Proc of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing. 2007:640-645.

[41] BRTOSSI A A, MANCINI L V, ROSSINI F. Fault-tolerant rate-monotonic first-fit scheduling in hard-real-time systems[J]. IEEE Trans on Parallel and Distributed Systems,1999,10(9):934-945.

[42] CHEREQUE M, POWELL D, REYNIER P,*et al*. Active replication in Delta-4[C]//Proc of the 22nd International Symposium on Fault-Tolerant Computing. 1992:28-37.

[43] ALDARMI S A, BURNS A. Dynamic value-density for scheduling real-time systems[C]//Proc of the 11th Euromicro Conference on Real-Time Systems. 1999:270-277.

[44] Ruan Xiao-jun, YIN Shu, MANZANARES A, *et al*. A message scheduling scheme for energy conservation in multimedia wireless systems[J]. IEEE Trans on Systems Man & Cybernetics,2011,41 (2):272-283.