

**毕 业 论 文（设 计）外 文 参 考 资 料 及 译 文**



**译文题目： 控制系统的弱硬实时模型调查**

**学生姓名：** 李轩 **学　　号：** 2012001059

**专　　业：** 软件工程

**所在学院：** 软件工程学院

**指导教师：** 闵建

**职　　称：** 讲师

2023年 12 月 28 日

说明：

要求学生结合毕业论文（设计）课题参阅一篇以上的外文资料，并翻译至少一万印刷符（或译出3千汉字）以上的译文。译文原则上要求打印（如手写，一律用400字方格稿纸书写），连同学校提供的统一封面及英文原文装订，于毕业论文（设计）工作开始后2周内完成，作为成绩考核的一部分。

摘要：

弱硬实时系统的概念可用于对实时系统进行建模，这些系统可能以有限和可预测的方式容忍偶尔错过最后期限。该模型适用于许多实际应用，在实时控制系统的背景下尤其有趣。实际上，应用硬性实时约束可能过于严格，因为在某些应用程序中，错过一定数量的截止日期是可以接受的。为了保持系统的稳定性，需要对违反最后期限的数量和分配施加限制。这些限制可以正式表示为薄弱的硬实时约束。当前在弱硬实时任务调度领域的研究侧重于设计调度算法，以保证满足约束条件，同时旨在最大限度地增加及时完成的任务实例总数。本文对与弱硬实时系统模型相关的工作及其与控制系统设计领域的联系提供了广泛的文献综述。描述了弱硬实时系统模型和相应的调度问题。此外，还概述了源自广义弱硬实时系统模型的系统模型，重点介绍了适用于实时控制系统的模型。描述并比较了用于调度具有弱硬实时约束的任务的最先进算法。最后，概述了依赖弱硬实时模型的控制器设计方法。

1. 导言

控制系统在现代工程中起着至关重要的作用，可以精确有效地调节复杂的物理过程。它们广泛用于各种应用，以确保稳定性、准确性和安全性。但是，设计有效的控制系统是一项艰巨的任务，需要深入了解系统动力学、建模技术和控制策略。

控制系统设计中的传统方法在时间限制方面采用保守原则。控制任务的执行受限于固定的时段，这些时隙是根据任务的最坏情况执行时间确定的。此外，不允许任务激活时间出现抖动。此类限制可以归类为硬性实时约束，严格禁止错过任务截止日期，因为这可能会对系统及其环境造成灾难性损害。此外，控制任务通常根据固定优先级策略进行安排，以保证可预测性。

实时调度理论的研究侧重于确定实时系统的规模，以最大限度地利用给定资源，而减少系统负载的方法通常依赖于放松严格的实时约束。此外，高级调度技术依赖于动态优先级分配，因为在一般情况下，在静态调度方案下不可能充分利用CPU（中央处理单元）[1]。尽管控制系统设计的既定方法实现了实时调度的一些基本原理，但这两个研究领域的单独发展导致控制问题领域与实时调度问题领域之间存在差距。因此，实时调度理论的结果和设计方法几乎不适用于控制系统设计技术 [2]。将更先进的调度技术集成到控制系统中可能非常有益，因为它可以提高系统设计的灵活性，并有可能设计出更具成本效益的控制系统。主要挑战是在控制性能和高效计算资源利用率之间取得平衡。因此，无论是在工业界还是在学术界，整合控制和实时调度研究领域都已成为一项巨大的挑战。文献中已知的大多数控制和调度协同设计方法都依赖于放松硬实时约束。放宽严格的实时限制通常意味着在某些条件下允许违反任务截止日期。对于包括实时控制系统在内的许多实际应用而言，严格的实时限制可能过于严格，因为如果错过最后期限的数量和分布遵循特定模式，系统可以容忍偶尔错过最后期限。由于实时控制系统的稳健设计，偶尔错过最后期限而受到充分限制对系统性能的影响可以忽略不计；它们可能导致服务质量下降，但不会危及系统的稳定性 [3,4]。此外，错过一定程度的截止日期的能力提供了更多的日程安排和灵活性，因此允许设计更具成本效益的系统。

另一方面，在软实时系统中，错过最后期限是可以容忍的，因为它们仍然可以为系统带来一定的效用。但是，软实时系统模型并未提供任何机制来限制违反最后期限的比率和连续性。引入弱硬实时系统模型是为了弥合硬实时系统和软实时系统之间的差距，它为描述在精确分布下能够容忍一定程度的截止日期错过的系统的特性提供了一个框架。为了定义时间约束以指定可容忍的最后期限和分布，引入了弱硬实时约束的概念 [4]。弱硬实时模型的第一个应用是在通信系统中调度消息的背景下 [5]。最初的模型进一步扩展到在单个CPU [6,7] 和多处理器系统 [8] 上调度任务的问题。在文献中，有许多应用依赖弱硬模型来定义时间约束，包括网络系统 [9,10]、控制系统 [11,12,13]、信息物理系统 [14,15,16]、通信系统 [17,18] 和多媒体系统 [19,20,21]。近年来，弱硬实时系统领域的研究方向包括可调度分析 [22,23,24,25]、能量受限调度 [26,27,28,29]、控制系统稳定性分析 [30,31,32] 以及控制和调度协同设计 [33,34]。最近有大量的调查论文涵盖了控制系统设计的新方法 [35,36,37,38]，只有少数论文回顾了基于弱硬实时系统模型的调度方法 [39,40]。在调度和控制系统协同设计的背景下，有几项现有的调查涵盖了这一主题。但是，他们主要考虑网络控制系统 [41,42]，从调度方面来看，它们处理的是超限处理策略的分析，而不是对不同调度算法的分析 [43]。在本文中，我们的目标是巩固基于弱硬实时系统模型的调度和控制协同设计领域的最新研究，并系统化弱硬实时系统的现有调度算法，这些算法迄今为止在任何调查论文中都没有涉及。这项工作的目标是为弱硬实时系统领域的潜在研究人员提供现有系统模型、调度算法和可调度分析方法的广泛背景。此外，还解释了与控制系统设计研究领域的联系，以及调度和控制系统协同设计领域中最重要的结果。提供了以下研究课题的文献综述：

文献中介绍的弱硬实时模型及其应用；

在弱硬实时约束下调度任务的现有方法；

分析弱硬实时系统可调度的现有方法；

最近在控制和调度协同设计方面的工作，重点是依赖弱硬实时模型的方法。

本文的其余部分组织如下。第 2 节描述了经典实时调度理论中的任务模型以及使用弱硬实时模型的动机。第 3 节描述了文献中已知的系统模型，这些模型可以从广义的弱硬实时模型中推导出来。第 4 节概述了用于调度具有弱硬约束的任务集的算法，而第 5 节描述了对此类任务集进行可行性分析的方法。第 6 节描述了控制和调度协同设计领域的挑战和最近的研究。第7节给出了结论，并概述了未来研究面临的开放挑战。

1. 预备知识

2.1 系统模型

在一般的实时系统模型中，实时系统由一组表示 N 任务：

，其中每项任务都通过三个时间参数来描述：周期

，最后期限

，以及最坏情况下的执行时间

。每项任务

释放无限序列的实例，即任务。这个 j任务的第-n 个任务

表示为

。对于定期任务，两个连续作业之间的间隔恰好是一个周期

。超级时期

是重复执行定期任务计划的最小时间间隔。通常，超周期对应于任务周期的最小公倍数：

= lcm （ T 1 ， T 2 ， ... ， T N ) 。如果任务的第一个任务未按时发布

，在首次发布任务之前经过的时间称为偏移量

任务的

。在一般情况下，的调用时间 j-th 个任务任务

，表示为

，等于

。最后期限

对应于一项任务的时间

必须完成其执行。与工作到达时间相关的截止日期称为相对截止日期，而与时间相关的截止日期则称为相对截止日期

绝对是最后期限。相对截止日期等于任务期限的特定案例，即

，被称为隐含的最后期限。任务截止日期是定义系统的时间约束和定义调度算法可行性条件的最广泛使用的参数。如果作业在截止日期之前完成了执行，则说该任务已在截止日期之前完成，否则将错过最后期限。如果所有工作都按时完成，则认为时间表是可行的。作业的响应时间

表示为

它的计算公式为

，在哪里

是作业的完成时间

。任务的最坏情况响应时间

是所有任务任务中的最大响应时间

:

最大

。一项任务

当且仅在以下情况下才被视为可安排

[44]。

给定集合的利用系数 U

N 个周期性任务中定义为执行任务集所花费的处理器时间的一小部分

[1]：

(1)

如果

，则该任务集不存在可行的时间表。

可调度分析中最重要的概念之一是关键时刻，即在所有任务任务中产生最大响应时间的作业的任务到达时间

。可调度性测试通常依赖于关键时刻定理 [1]，该定理指出，任务的关键时刻

每当有任务时发生

与优先级较高的作业同时发生 [45,46

]。

弱硬实时系统可以通过扩展所描述的具有弱硬时间约束的系统模型来表征。弱硬约束用于指定可容忍的最后期限错过的程度和分布。如果违反约束，则称系统处于动态故障 [5]。

定义 1

（动态故障）。

如果在连续 k 个客户的窗口中满足最后期限的客户少于 m 个，则截止日期为 (m, k) 的直播会出现动态故障。

文献中已知有几种过程模型指定了这些约束：

-公司模型

，

跳过模型，

广义的弱硬实时模型。

这个

-firm模型是在通信系统中调度消息的背景下引入的 [5]。这个

-fir m 约束条件规定，在任何 k 次连续调用中都必须满足至少 m 个截止日期。跳过模型是一种涵盖不允许连续错过最后期限的特殊情况的方法 [47]。该模型使用跳跃因子表示法

，它规定在一个窗口中只能错过一个截止日期

连续调用任务

。跳过模型的应用包括实时控制系统和多媒体系统。在实时控制系统中，连续错过最后期限将危及系统的稳定性，而在多媒体系统中，连续错过最后期限将降低服务质量。广义的弱硬实时系统模型提供了一组常见的约束条件，这些约束条件描述了两者

-公司模型和跳过模型。此外，它们还描述了对错过的最后期限的数量和连续性的限制，这些限制不在

-固定和跳过模式。这个

第 3 节详细描述了固定模型、跳过模型和广义弱硬模型

。

必须强调的是，根据应用程序的不同，系统模型在处理错过截止日期的策略方面会有所不同。最常见的方法是裁员和继续工作策略 [48]。在 job-kill 方法中，对未完成任务的计算会在截止时间即时丢弃，作业中止。这种方法适用于跳过模型。在任务继续策略中，任务将始终执行直到完成，这种方法通常用于

-公司模型

。

2.2。动机和工业应用

本节提供了一个实时控制系统的实际示例，该示例说明了通过弱硬实时模型分析此类系统的动机。事实证明，根据弱硬实时模型设计系统可以确保过载条件下的预测行为。此外，弱硬约束可以为解决过载情况提供一种便捷的机制

。 示例 1

（计算机驱动的控制系统）。以用于控制自主移动机器人 (AMR) 的实时系统为例。AMR 由电池供电，并配备了用于定位和障碍物探测的飞行时间 (ToF) 传感器。定位还依赖于测量车轮旋转速度的编码器。AMR 配备了用于视频和音频信号的发光二极管 (LED) 和蜂鸣器。控制系统的每项功能都通过定期任务来执行：定位、导航、障碍物检测、电池管理、电机控制和信号。另一项任务，即避障任务，将在检测到障碍物时激活，在计算出避开障碍物的轨迹之前，该任务将保持活动状态。此任务还会定期释放实例。任务和硬件组件（传感器、执行器等）的示意图如图 1 所示。表 1 中总结了任务的时间特征。

该系统在正常运行条件下得到充分利用，即当避障处于非活动状态时：

(2)

激活避障任务后

，总利用率变为：

(3)

1.1

并且系统暂时处于过载状态。在此示例中，通过应用弱硬实时约束来解决过载问题。

在正常运行条件下，调用信号任务的频率应与导航任务的频率相同，以便在附近环境中正确地向人类指示 AMR 的运动，例如，当 AMR 改变方向时改变 LED 灯的颜色，当 AMR 倒车时打开蜂鸣器信号等。但是，在系统设计期间，信号任务被分配了约束条件 (1、5)，这意味着至少要完成一项工作在连续发布的五个工作岗位中，必须完成——换句话说，五分之四的工作可能是必要时丢弃。这意味着，在最坏的情况下，导航任务的每五分之一输出都会发出信号。即使在最坏的情况下，此任务的功能也不会受到丢弃任务的影响，因为人眼看不出以 20 Hz 而不是 100 Hz 的频率刷新 LED。此外，在最坏的情况下，蜂鸣器将在相应的导航操作发生后50毫秒触发，并且这种延迟也是可以容忍

的。

电池管理任务负责监控电池状态、估算功耗以及评估何时安排充电。通过跳跃系数为该任务分配了弱硬约束

—丢弃一项任务后，必须连续完成四个作业。这是可能的，因为该任务的输入不经常更改，而且电池管理算法的稳健设计允许电池管理系统（BMS）即使在每五项工作都被丢弃的最坏情况下也能稳定运行

。

调度算法旨在仅在过载条件下调度强制性作业。如果我们考虑信号和电池管理任务的最大可跳过实例数量，则总利用率

变为：

(4)

在不危及稳定性和降低系统性能的情况下成功解决了过载问题。避障任务的操作完成后，系统将恢复到正常运行模式。

必须强调的是，关于电池管理任务限制的假设也可以用于其他任务（电机控制、导航、定位）；但是，为简单起见，跳过模型仅适用于一项任务。

此示例显示了弱硬约束不仅可以确保过载条件下的确定性行为，而且还为设计具有成本效益的系统提供了一种机制，该系统在正常条件下效率最高，而在过载条件下，可以以可预测的方式降低每项任务的利用率。对考虑过的具有严格实时约束的实时系统进行建模需要对系统进行尺寸测量，以便在最坏的情况下能够满足所有截止日期。与正常操作相比，最坏情况下的任务需求通常比本示例中显示的还要高，如果根据最坏情况下的需求来确定系统的尺寸，则无法在正常运行模式下充分利用该系统。因此，放松严格的时间限制可能有利于更好地利用资源。另一方面，对允许放宽时间限制的任务（信号和电池管理任务）应用软实时约束是不够的，因为无法保证错过最后期限的模式，并且连续错过大量截止日期可能会损害系统的稳定性并降低性能。此外，在软时间限制 [49,50,51] 下调度系统的常用技术基于缩短任务的响应时间，而不考虑违反截止日期的每项任务的比率。因此，对于特定应用（例如示例中显示的控制系统），为了设计具有精确约束时间行为的经济高效的系统，可以合理地考虑时间限制，这些限制不如硬实时要求严格，但比软实时要求更为严格。

3.弱硬实时系统模型概述

3.1。任务模型包含

-确定截止日期

与能够容忍偶尔错过最后期限的系统有关的最早研究是哈姆达维和拉马纳森的工作。在 [5] 中，他们引入了以下概念

-确定截止日期，作为通信系统中调度消息时限的规范。更具体地说，他们的系统模型由可调度实体流（他们称之为客户）组成，这些实体必须安排在单个服务器上。该系统模型可用于表示许多计算机和电信系统，包括在 CPU 上执行多个任务以及从共享同一介质的多个流传输消息。每个客户都有指定的截止日期

由此可望该系统提供全面服务.如果没有在截止日期之前为客户提供全面服务，则视为错过了最后期限。为客户提供服务所需的时间称为服务时间，它等同于

。所有直播的服务时间都相同，不能抢占服务。客户提出请求之间的最短间隔时间等于该期限

。因此，可以将每个直播描述为一项任务

，而与直播对应的客户请求等同于一项任务

。为简单起见，此后使用任务和任务这两个术语来表示直播和相应的客户。为每项任务分配了两个参数，即 m 和 k，这两个参数指定在任何 k 个连续请求任务的窗口中，至少有 m 个作业必须在截止日期之前完成。如果未满足所述要求，则任务出现动态故障。值得注意的是，这种需求意味着最大截止日期错过率为

。所描述的系统模型如图 2 所示

。

实时系统研究中解决的问题

-firm deadlinedy 是将每项任务的作业安排到一台服务器上，这样可以避免动态故障

。

3.2。跳过模型

跳过模型是由科伦和莎莎在 [47] 中引入的。作者解决了安排可抢占的定期任务的问题，这些任务可能偶尔会错过最后期限。任务截止日期被认为是隐含的，即等于任务周期。该模型采用了裁员策略，错过最后期限的工作将被视为跳过。系统在过载条件下进行行为分析，跳岗的限制相当保守；不允许连续两项工作错过最后期

限。

通过引入跳跃因子对经典任务模型进行了扩展

，它规定了任务的容忍度

转到跳过的工作。更具体地说，跳过系数定义了两个错过截止日期的工作之间必须在截止日期之前完成的工作数量的最低要求。如果是一项任务

至少被跳过了

后续工作必须在最后期限之前完成。值得注意的是

无穷大

对应于不允许跳过最后期限的硬实时系统。为了对作业是否可以跳过进行分类，作者提出了红色和蓝色作业的术语。红色任务必须在截止日期之前完成，而蓝色任务可能会错过最后期限。

3.3。广义弱硬实时系统模型

弱硬实时系统可以正式定义为 [4]：

定义 2

（弱硬实时系统）。

弱硬实时系统是指在时间窗w内按时完成和错过的最后期限的分布有精确界限的系统。

为了描述弱硬实时系统的需求，在 [4] 中引入了弱硬实时模型。经典的周期性任务模型通过弱硬约束进行扩展

，其中规定了对按时完成或错过最后期限模式的要求：

。时间限制

汇总在表 2 中，并在以下定义中正式说明 [52]。

定义 3

（约束

）。任务 “满足 m 中任意 n 个” 的截止日期 (

，

) 并表示为

如果在连续调用 m 次任务的任何窗口中，至少有 n 次按任意顺序调用符合截止日期的调

用。 定义 4

（约束

）。任务 “满足 m 行第 n 行” 的截止日期 (

，

) 并表示为

如果在连续调用 m 次任务的任何窗口中，至少有 n 次连续调用符合截止日期。

定义 5

（约束

）。一项任务 “错过了 m 中的任何 n” 截止日期 (

，

) 并表示为

如果在 m 次连续调用任务的任何窗口中，错过的最后期限不超过 n 个

。 定义 6

（约束

）。任务 “错过了 m 行第 n 行” 的截止日期 (

，

) 并表示为

如果在连续调用 m 次任务的任何窗口中，没有连续 n 次调用错过

截止日期。

对于

约束，不需要参数 m，因为约束条件的满足不取决于窗口的大小；因此，

。

值得注意的是，以前的研究人员引入的约束条件可以直接通过所描述的约束条件进行建模：

这个

[5] 中的-firm 约束条件可以直接映射到

约束;

通过跳跃因子定义的约束条件

从 [47] 对应于

。

为简单起见，在本文的其余部分中

-公司约束表示为

约束和与跳过模型相对应的约束条件表示为

。

每项任务按时完成和错过最后期限的历史通常通过二进制序列来描述，该序列称为

-图案。这个

-pattern 包含验证任务是否满足其弱硬约束所需的足够信息

。 定义 7

（

-图案）。微型图案

一项任务

是一个符号序列

，其中 1 表示已按时完成最后期限，0 表示错过了第 j 个任务的最后期限。的第 j 个符号

是通过以下方式计算的：

(5)

1

如果

0

否则

的长度

-pattern 对应于弱硬约束下的截止日期数量。一个例子

-满足 “必须满足五个截止日期中的四个” 限制的模式是 11011。最左边的位对应于最早调用的截止时间。如果以下调用错过了截止日期，

-pattern 变成 10110 并且相应的任务处于失败

状态。

弱硬约束用于两种不同的上下文：可行性分析和新调度算法的实现。在可行性分析的背景下，目标是确定给定算法调度的任务集是否满足弱硬约束。在调度算法的实现中，

-patterns 用于执行动态失效概率的在线计算

。

3.4。研究问题

考虑到所描述的弱硬实时系统的规格，有几个问题需要解决：

指定较弱的硬性时间约束；

分析系统在弱硬约束下的可调度；

在弱硬约束下实现调度系统的算法。

[52] 详细阐述了为弱硬实时系统指定时间限制的问题。应用现有的硬实时系统分析方法无法解决分析弱硬约束下任务的可调度性的问题，因为关键的即时定理不适用于弱硬实时系统 [48]。调度具有弱硬约束的系统的问题归结为利用弱硬约束来减少系统负载，从而能够调度原本不可行的系统。现有的日程安排方法通常被归类为有保障的或尽力而为。对于有保障的调度算法，存在可调度测试来保证不会发生动态故障，而无保证的调度器则使用尽力技术来避免动态故障。通常将尽力而为的方法与动态失效的概率进行比较。动态失效概率用于衡量违反弱硬约束的频率。用于比较调度算法性能的另一个指标是服务质量 (QoS)。QoS 表示已完成的任务占任务总数的比率

：

(6)

QoS

数字

已完成

总计

数字

有保障的方法通常采用离线优先级分配，其中构造一个预先构建的序列来决定哪些任务将被跳过。在线方法通常根据系统的动态参数在运行时做出调度决策。调度方法的复杂性通常决定了该方法是否足以满足特定的应用程序。例如，计算成本高昂的可调度性测试不足以满足在线应用程序的需求

。

4。调度方法概述

4.1。的调度方法

-公司系统模型

Hamdaoui和Ramanathan引入了基于距离的优先级分配（DBP）作为降低动态故障平均概率的方法 [5]。在这种技术中，在运行时跟踪每项任务的状态，以确定接近失败状态的任务，即截止日期少于 m 的状态。通过为最接近失败状态的任务分配最高优先级，可以降低动态失败的概率。为每项任务分配一个优先级值，该值等于将任务从当前状态变为失败状态所需的最小连续失败次数。在这种情况下，优先级值较低的客户将获得更高的优先级。

拟议的优先权分配技术可以用状态转换图来说明。例如，考虑一项任务

和

约束。图 3 显示了任务的状态转换图

。最初的

-pattern 是 111。图的边缘表示可能的状态转换（错过最后期限或已达到最后期限）。如果一个州不满足

约束，任务处于失败状态，由阴影形状表示。处于失败状态的任务的优先级值为 0，即最高优先级，而其他任务的优先级与失败状态的距离相对应。例如，具有以下内容的任务

-pattern 110 的优先级分配为 1，而任务的优先级为

-pattern 111 被分配优先级 2

。

与使用最早截止时间优先 (EDF) 算法的调度方案相比，这种方法显著降低了动态失效的概率。但是，DBP 方法可能存在的一个缺点是，它仅在考虑当前任务的情况下分配优先级，而不考虑

-共享同一服务器的其他任务的模式。此外，DBP 优先级分配仅基于错过和按期完成任务的历史记录，不考虑其他属性，例如截止日期、最短到达时间

等。

在他们的进一步研究中，哈姆达维和拉马纳森提出了一个用于计算动态失效概率的分析模型 [53]。该方法可用于为实时任务提供统计服务质量保证。

[54] 在多跳网络调度的背景下进一步研究了DBP方法，其中引入了扩展调度策略DBP-M。在这种方法中，动态调整任务的优先级，以减少连续错过最后期限的次数，从而降低动态失败的可能性。每项任务的优先级是根据状态信息计算的，状态信息保存在每个任务的单独队列中。值得注意的是，队列的数量随着任务数量的增加而增加，因此，这种方法在计算和内存要求方面会产生开销。但是，就动态失效的概率而言，这种方法的性能优于传统的 DBP

。

在 [55] 中，现有尽最大努力调度系统的可扩展性问题

限制条件已得到解决。在这种情况下，可扩展性是指调度算法的特性，它可以在不增加计算要求成本的情况下处理任务数量的增加。作者指出了考虑动态故障的性能和实现该性能所涉及的调度成本之间的权衡。DBP-M 需要大量的调度成本才能为每项任务维护单独的队列。另一方面，诸如 EDF 之类的传统调度策略要求所有任务使用单一队列，因此需要

日程安排成本。Striegel和Manimaran推出了增强型基于距离的优先级（EDBP）算法，该算法具有与DBP-M方法相似的性能，但保持了固定的调度成本 [55]。EDBP 算法通过将任务的 DBP 状态与单个作业（即截止日期前的剩余时间）相结合来调度作业。仿真表明，所提出的方法实现了与DBP-M方法相似的动态故障性能，同时使调度成本与EDF算法的调度成本接近

。

王等人研究了将DBP与EDF以外的调度策略相结合的可能性 [56]。这种方法背后的动机是，无论错过和按时完成的最后期限的分布情况如何，DBP与EDF相结合，都只考虑与故障状态的距离。例如，考虑下面的两个任务

约束条件有以下几点

-模式：00001 和 10000。两种模式都对应于故障状态，DBP 将为这两个任务分配相同的优先级 (0)。但是，为了退出失败状态，任务有

-pattern 00001 需要在最后期限之前完成一个，而任务 10000 需要两个。王等人引入了综合DBP（IDBP）作为一种补救方法，它克服了传统DBP所描述的缺点。IDBP 使用优先级分配方案，该方案不仅考虑与动态故障状态的距离，还考虑退出故障状态的距离。对该方法的评估表明，与传统的DBP相比，IDBP可以有效降低过载条件下动态失效的概率

。

作者陈等人的进一步研究通过引入类别选择算法（CSA）扩展了EDBP算法 [57]。作者研究了可扩展性和 QoS 粒度之间的权衡。QoS 粒度是 [58] 中引入的一项指标，它对应于每项任务实现和所需的 QoS 的变化。CSA 的主要思想是提供一种将任务分类为预定义数量的类（即队列）的机制。使用固定数量的类可以提高可扩展性，但代价是 QoS 粒度，因此，可以实现可扩展性和 QoS 粒度之间的平衡。分配优先级是按班级而不是按任务执行的，因此，班级中的某些任务可能会违反其个别任务

限制。CSA 的系统模型如图 4 所示。第一步是根据CSA为每项任务选择一个类别。然后根据IDBP对课程的实体进行安排

。在@@

向类别分配任务的策略方面，CSA 有两种不同的变体：静态 CSA (S-CSA) 和动态 CSA (D-CSA)。在静态 CSA 中，根据任务所需的 QoS 将任务分配给某个类别，其中每个类别对应固定的 QoS 值间隔。动态 CSA 平衡每个队列中的任务数量，并动态调整每个类别的 QoS 值边界。S-CSA和D-CSA均根据动态故障率进行了评估，并与IDBP和EDF方法进行了比较。图 5 描绘了将 S-CSA-IDBP 方法与 EDF 和 IDBP 调度器实现的故障率进行比较的结果。模拟系统包含 15 个任务。m 和 k 值根据均匀分布随机生成，任务遵循泊松分布。所有作业的服务时间等于 1。结果表明，一个类的S-CSA-IDBP的性能与EDF相同，这是预期的行为，因为它等同于具有15个任务和1个队列的EDF。另一方面，具有 15 个类别的 S-CSA-IDBP 的性能与 IDBP 相同。通过调整CSA中使用的类别数量，可以实现EDF和IDBP方法属性之间的平衡

。

经典 DBP 方法的另一项增强是 [59] 中引入的 Matrix-DBP。该算法使用其他任务的作业服务期间可能发生的最小截止时间错过次数，增强了 DBP 的优先级分配方案。这样，就采用了错过最后期限的实际数量的下限。有关在完成任务 j 时我可以错过的最小截止期限数量的信息包含在一个名为互惠矩阵的矩阵中。调度器按以下方式运行：每次需要调度作业时，都会通过减去相应的矩阵元素来更新 DBP 优先级。实验表明，Matrix-DBP在截止时间错过率方面的表现优于传统的DBP，并降低了失效状态的概率。这种方法引入了非常低的计算成本，因为它仅依赖于访问矩阵的元素。

[60] 中描述了有关矩阵-DBP方法的进一步研究。作者介绍了等效的 Matrix-DBP，它通过考虑非周期性任务来扩展矩阵DB

P。

另一种调度实时系统的方法

约束条件由 West 和 Schwan 在 [61] 中提出。作者研究了多媒体流中多帧的调度，并开发了动态窗口约束调度（DWCS）算法。该算法旨在根据延迟帧的丢失和延迟限制来限制延迟帧（作业）的数量。要传输的帧的优先级是根据两个属性分配的：截止时间和丢失容差。按照以下说明使用符号

约束，损失容差可以指定为值

。请注意，损耗容限的较低值对应于更接近动态失效的状态；因此，调度器优先考虑损失容差最低的任务的作业。同一任务的任务都具有相同的损失容差，并且它们是按到达顺序排定的。当一项任务错过最后期限时，同一任务的所有其他作业的损失容忍度就会降低；因此，该任务的优先级会提高。相反，如果在截止日期之前安排任务，则任务剩余任务的损失容忍度会增加。由于每次在要求的时间段内维修或不维修任务时都必须调整任务的损耗容限值，因此 DWCS 的每个步骤都需要

计算时间。

在 [62] 中，West 和 Poellabauer 描述了 DWCS 算法的修改版本，该版本可以保证 DWCS 算法的实现

限制。在这种方法中，流程模型仅限于隐含的最后期限

并且初始损失容差值相同。作者证明，在这些限制下，DWCS可以保证满足薄弱的硬约束，即不超过

每隔 m 份工作都错过了截止日期。此外，即使调度器过载，DWCS也可以保证实时任务的有限服务延迟 [6

3]。

在 [64] 中，伯纳特和伯恩斯解决了在网上决定哪些工作应该错过最后期限的问题，以增加执行非周期性任务的空闲时间。他们的方法基于双优先级调度（DPS）[65]。DPS 的主要思想是保持硬任务的可预测性，同时刺激非周期（软）任务的执行。DPS 需要三个不同的优先级频段：低频段、中频段和上限频段。每个硬任务可以在下限或上限区间执行，而软任务可以在中间频段执行。必须将艰巨任务从下限提升到上限的最佳时间可以通过离线分析来确定。在系统中

限制，可以利用对错过最后期限的容忍度来提高软任务的响应能力。这可以通过不提升某些任务调用来延迟优先级提升时间来实现。[64] 中介绍了几种延迟优先晋升时间的策略，以及一种用于在线决定要推广哪些任务的低成本动态算法

。

实时系统中的过载管理技术

[11] 在从控制回路的故障状态中恢复的背景下讨论了截止日期。提出的方法依赖于离线调度算法，该算法通过将任务分为强制和可选任务来降低过载条件下的等效利用率。分区的执行方式是

限制条件得到满足。强制性工作使用单调费率政策以固定的优先顺序排列，以保证强制性工作的最后期限。可选工作的优先级最低，不能保证在截止日期之前完成。一次激活的任务

在以下情况下被归类为必填项：

(7)

[11] 中表明，这个条件至少可以保证

出来了

连续实例被归类为强制性的；因此，

限制是有保证的。必须强调的是，这种实例分类不一定是最佳的。

权和胡的著作描述了与制定强制性工作政策有关的进一步研究。在 [66] 中，可以看出，找到将任务分为必选任务和可选任务的最佳策略的问题是 NP-Hard，以及确定结果任务集的可调度性的问题。作者提出了一种用于改进 [11] 中提出的任务分区策略的优化技术。这种技术的主要思想是通过修改强制和可选作业的模式来提高任务集的可调度性，以减少任务之间的执行干扰，即强制任务之间可能发生抢占的时间长度。作者根据中国一般剩余定理（在 [67] 中介绍）提出了一种确定任务间干扰的新技术。执行干扰用于计算优化过程的目标函数。作者提出了两种获取任务分区模式的优化方法：模拟退火和遗传算法。实验表明，所建议的启发式方法显著改善了可调度性，尤其是在过载条件下

。

在一篇特定的论文 [68] 中，提出了有保障的动态优先级分配（GDPA）方案，该方案旨在最大限度地提高任务的服务质量

约束，同时提供系统负载不足时违反约束条件的有限概率。活跃作业的顺序是根据与故障状态的距离和最早的截止日期来确定的。在算法的每个步骤中，都会测试构造的任务序列的可行性。事实证明，GDPA提供了调度最佳性，因为当系统负载不足时，GDPA可以满足所有任务截止日期，EDF算法也是如此，并且可以保证不会发生动态故障。由于需要进行在线可行性测试，因此GDPA算法的计算成本为

。对于无法承受高计算成本的应用程序，提供了GDPA的低成本版本——简化的GDPA（GDPA-S）。这种方法引入了

计算成本。评估拟议方法的结果如图 6 所示。用于比较的指标是截止日期满意度概率（PDS），其定义为截止日期满意度与职位发布的比率，它直接对应于服务质量。任务是随机分配的

限制。EDF、GDPA 和 GDPA-S 支持

当总利用率需求小于或等于 1 时的 PDS。在过载条件下，GDPA和GDPA-S实现了最高的PDS。

在信息物理应用的背景下，研究人员将弱硬实时约束应用于零星任务，以减少零星任务对其他任务的干扰 [16]。引入了一种新的调度算法，即作业类级调度器 (JCLS)。主要思想是根据先前完成的最后期限的数量，为零星的工作分配一个班级。将所提方法的性能与文献中的其他方法、具有速率单调周期分配的RTO和任务集可调性方面的弱硬调度分析（[25]）进行了比较。与其他考虑的方法相比，JCLS显示出明显更高的调度率

。

在 [69] 中，提出了一种调度算法，即基于紧急情况的全球调度（GEBS），旨在提高弱硬实时任务的可调度率。优先级分配不仅基于作业的运行时属性，还基于所有任务的紧急类别，这使得这种方法成为全球优先级分配方案。确定任务紧急类别的算法依赖于可行性测试，以提高给定任务集的可调度性。将GEBS方法与JCLS [16] 和 [66] 中提出的调度方法进行了比较，结果表明，从运行时间开销的角度来看，GEBS具有更高的调度比率和更好的性能

。

4.2。跳过系统模型的调度方法

跳过模型的作者已经提出了几种在线调度算法，这些算法实现了减少永久过载的可跳过方法。减少永久过载的技术在文献中被称为跳跃法 [

70]。

评估跳跃算法的结果是在假设深红色条件下得出的，如第 5.2 节所定义。

跳过模型的作者提出并分析了两种利用跳过的在线调度算法：仅限红色任务（RTO）算法和可能时为蓝色（BWP）算法。在以下文本中，描述了跳过任务的算法。“仅限红色任务” (RTO) 算法通过始终跳过蓝色作业来减少系统负载，无论它们对系统负载有何影响。红色作业根据 EDF 算法进行调度。[47] 证明，RTO算法在深红色条件下是最佳的，因为所有可行的集合都可以使用RTO进行调度。值得注意的是，在 RTO 下，连续两次跳过之间的距离正好是

（

）周期。如果系统中没有活跃的红色作业，则允许执行蓝色作业（BWP）算法。红色作业根据 EDF 算法进行调度，而蓝色作业可以根据任意调度策略进行调度，包括 EDF、最晚的截止日期优先等

。

作者 Marchand 和 Silly-Chetto [71,72,73] 进一步研究了使用可跳过的作业来安排任务集。作者引入了红色任务尽可能晚（RLP）算法，该算法通过在初步计划的空闲时间执行蓝色任务来刺激蓝色任务的执行，只考虑红色作业。空闲时间是指计划中没有处于就绪状态的红色作业的时间跨度。RLP 算法由以下特征指定

：

如果系统中没有蓝色作业，则根据EDF尽快安排红色作业；

如果系统中存在蓝色作业，则将尽可能晚地处理红色作业，蓝色作业将在红色作业的空闲时间内处理。

任何任意调度方案都可以在空闲时间内调度蓝色作业。确定红色作业空闲时间的机制依赖于尽可能晚的最早截止日期 (EDL) 算法。EDL 算法是在 [74] 中引入的，它最初的设计目的是通过在定期任务的空闲时间调度软非周期性任务来最大限度地缩短它们的响应时间。在 RLP 算法的上下文中，EDL 算法用于确定红色作业的最新开始时间，以最大限度地缩短调度蓝色作业的松弛时间。此外，一旦知道空闲间隔，就可以在线决定接受蓝色工作是否可行。当系统中没有活跃的蓝色任务发布时，以及在蓝色任务完成之后，如果系统中仍有蓝色作业，则需要在线计算可用的空闲时间 [73]

。

在 [75] 中，Marchand和Silly-Chetto推出了带有蓝色验收测试（RLP/T）的RLP算法，该算法增强了RLP算法实现的服务质量。作者指出，RLP算法的主要缺点是蓝线作业在完成之前中止的风险。因此，应进行在线测试，以确定是否可以接受蓝色任务执行。该算法的总体思路如下：红色作业在激活时进入系统，而蓝色任务在接受后进入系统。一旦蓝色工作被接受，就会尽快将其与红色工作一起安排。验收测试基于 [76] 中提出的方法，该方法最初用于对在由定期任务组成的环境中发生的零星任务进行验收测试。在这种情况下，该方法用于计算可用于执行蓝色任务的空闲时间，即在不错过最后期限的情况下无法安排任务的最大时间。执行验收测试的复杂性是

酋长国

最大

，其中 N 是定期任务的数量，

最大

是最长的截止日期，

是最短的时间段，而且

表示时间 t [72] 处于活动状态的蓝色任务的数量

。

图 7 对通过上述算法实现的 QoS 进行了比较。结果是通过模拟执行50个周期性任务集获得的，每个任务集由10个具有相同跳跃系数的任务组成，

。将结果与系统总负载进行比较。系统负载在间隔内有所不同

0.9

1.6

通过调整任务的最坏情况执行时间。RTO 算法的 QoS 等于

无论系统负载如何。RLP 和 BWP 算法的性能相似，但明显差于

RLP/T。

图 8 显示了所述算法产生的开销与集合中任务总数的比较。开销对应于执行调度操作所花费的 CPU 时间。测量在 1000 秒的仿真时间内完成。算法 BWP 和 RTO 不会带来任何开销，因为它们不需要在线计算闲置时间或执行可调度性测试。RLP 算法获得的结果是根据 EDL 计算空闲时间的结果。RLP/T 调度算法带来的开销随着任务数量的增加而显著增加。这是因为每次发布或完成蓝色作业时都必须计算 EDL 空闲时间

。

所描述的算法考虑的系统仅由可跳过的定期任务组成。Caccamo和Buttazzo的工作解决了将可跳过的定期任务与软非周期性请求相结合的问题 [77]。由于必须尽快完成非周期性任务以最大限度地缩短响应时间，因此作者研究了利用跳过来增强非周期性任务响应能力的可能性。在提出的技术中，RTO和BWP算法与总带宽服务器（TBS）算法集成在一起 [78]。TBS 算法提供了一种在线机制，用于调度非周期性请求，同时保证定期任务的执行

。

在他们的进一步研究 [79] 中，Buttazzo和Caccamo引入了一种机制，用于调整 [77] 中提出的截止日期分配算法，以平衡性能和复杂性。此外，他们还描述了如何将TBS提高到最佳状态，以实现最短的响应时间

。

在 [80] 中，通过引入超启发式方法来优化 QoS，提高了 BWP 和 RLP 算法的性能。更确切地说，通过基因编程演变而来的启发式算法被用来分配蓝色工作的优先级。与RLP和BWP相比，所提出的方法可以在过载条件下将QoS提高多达15％。此外，对优先级分配启发式方法的评估具有计算效率，并且

不会带来开销。

4.3。广义弱硬实时系统模型的调度方法

除了

约束。在 [4] 中，弱硬实时系统模型的作者讨论了一种在弱硬约束下为任务集分配最佳优先级的方法。通过应用分区方法可以找到最佳优先级分配，它基于以下特性：如果任务在弱硬约束下可调度，优先级为 p，则也可以在任何高于 p 的优先级下进行调度。算法为每项任务分配初始优先级，例如 Deadline Monotonic，并检查优先级最低的任务是否满足其限制。如果不是这样，则将为另一个任务分配最低优先级，然后重复该过程。当找到使所有任务都可调度的优先级分配方案，或者没有发现任何任务可以按给定的优先级进行调度时，寻找最佳优先级分配的过程即告完成。在后一种情况下，任务集不可行。

一种支持两者的调度算法

和

约束条件是双模态调度器 [81]。双模态（BM）调度器是一个在线调度框架，它考虑定期和非定期任务，并受隐性截止日期的限制。调度器基于两种不同的调度模式。在正常模式下，作业是根据任意调度策略安排的，该策略可能会考虑错过和按时完成的历史记录。如果可能发生动态故障，则调度程序会切换到紧急模式，为此，可以保证任务能够在最后期限之前完成。发布职位时，会根据在截止日期之前完成任务的关键程度为其分配两个州之一。如果工作可以错过最后期限，而不会影响对相应任务的薄弱硬约束的满足，则分配非临界状态。非关键任务由调度程序在正常模式下调度。如果作业必须在最后期限之前满足弱硬约束，则会将其分配为临界状态并安排在紧急模式下

。

保证所有关键任务都可以在紧急模式下调度的可调度性测试基于最坏情况的响应时间分析。作者证明，至少如果将关键任务提升为恐慌模式，系统永远不会产生动态故障

截止日期之前的时间实例 [81]。

已发布任务的状态通过关键度函数确定。临界度函数

确定在不违反弱硬约束条件的情况下可以错过的连续截止日期的数量

，基于

-图案

任务的。在 [81] 中，为四种类型的弱硬约束定义了几种临界函数。如果

，已发布的任务处于临界状态，必须将其安排在紧急模式下。这可以在关键任务发布后立即执行（即时紧急模式），如果有闲置时间（延迟紧急模式），则可以延迟一段时间。紧急模式是作为固定优先级机制实现的，其中关键任务的优先级严格高于非关键任务。作者建议根据前面文本中描述的最佳优先级分配来分配优先级。非关键任务可以根据任何调度策略进行调度。应用跟踪任务的计划策略可能很有用

-模式，从而避免进入恐慌模式

。

作者通过分析动态失效的数量并将其与其他方法（EDF和DWCS方法）进行比较来评估建议的方法 [62]。在大量综合生成的任务集上测试了调度算法的性能，平均利用率在间隔内

0.8

1.4

。EDF 调度器用于在 BM 调度器下以正常模式调度作业。评估结果如图 9 所示。在 BM 调度算法下未发生动态故障，这是预期的行为。

[82] 中描述了另一种在弱硬约束下保证任务集调度的方法。引入了两种算法，即 Meet any 算法 (MAA) 和 Meet row 算法 (MRA)，用于在

和

分别是约束条件。两种算法都旨在通过提高有可能提前进入失败状态的任务的优先级来避免动态故障。优先权的计算

对于每项任务，都基于两个表达式的组合：

(8)

哪里

是优先级分配表达式，对应于任何任意算法（例如 EDF）和

这种说法可以保证最后期限不够严格。这个表情

对于

和

限制。

和

是决定约束权重的正实数

和

。在这种方法的背景下，更大的值是

对应于更高的优先级值。作者提出了一种计算其价值的技术

基于在截止日期前完成的数量

-这样的模式

约束条件已得到满足。拟议的测定技术的计算成本

是

。在 [83] 中，作者提出了一种确定方法

对于 MRA 算法

复杂性。

图 10 显示了 MRA 和 MAA 方法在不同的值下实现的满足约束的平均速率

如果 CPU 利用率为

1.3

。将结果与 EDF 方法进行了比较，由于 CPU 利用率也是恒定的，后者获得了恒定的满足约束率。当

比率小于 1，强调保证行为；满足约束条件的平均速率接近 1。

表 3 汇总了本节中介绍的调度算法，包括的算法

-固定、跳过和广义的弱硬实时模型。这些算法的分类取决于它们是保证不会发生动态故障，还是旨在最大限度地降低动态失效的概率（保证或尽力而为）。将相应的优先级分配策略分为固定优先级和动态优先级分配分别由 FP 和 DP 表示。讨论了每种算法优先级分配的复杂性，同时考虑了任务的数量 N，最大长度

-图案

最大

，以及其他特定于算法的参数。此外，还列出了所描述算法的主要优点。

5。弱硬实时系统的可调度性分析

本节介绍确定第 3 节中描述的任务模型可调度的方法。此外，还提供了针对特定调度算法推导的可调度条件

。

5.1。的可调度分析

-公司系统模型

给定一组 N 个定期任务

约束条件，如果可以找到可以满足条件的调度算法，则称该任务集是可调度的

所有任务的限制。调度算法通常会产生一种模式，用于定义哪些工作是强制性的，即必须在截止日期之前完成，哪些工作是可选的，即可以错过最后期限。事实证明，决定一组任务中每项任务是否存在强制性和可选截止日期模式的问题

这样

从强烈的意义上讲，是可调度的 NP-Hard [66]。此外，考虑到每项任务的强制性和可选截止日期模式

，问题是确定是否

也是可调度的 NP-Hard [66]。弱硬实时系统的可调度分析通常依赖于为特定调度算法得出的足够条件

。

下任务集可调度的充分条件

[66] 中推导了任务周期为谐波的特定情况的约束条件。对于具有谐波周期的任务，每个周期都是较短周期的整数倍。假定这些任务有隐含的截止日期，即

。 定理 1.

给定两个任务集

和

和

，

，

，

，以及

分裂

如果

。根据给定的

限制，

在以下情况下是可以安排的

：

(9)

在哪里

是任何长度时间间隔内的最大必需作业数

。

这种条件是Han和Tyan [84] 研究的延伸，在该研究中，作者引入了一种多项式时间算法，用于测试固定优先级下硬实时系统的可调性。这种方法的基础是将集合中的每个任务映射到一个新任务，使新周期小于或等于原始周期，而计算时间保持不变。任务周期的映射是以这样的方式进行的，即生成的任务集必须是谐波的。[84] 证明，如果新的谐波任务集是可调度的，那么原始任务集也是可调度的。给定 [84] 中描述的技术，可以找到相应的谐波任务集，并且可以根据定理1测试任务集的可调性。这种方法的实现需要

日志

计算时间，其中

最大

而 N 是任务的数量。

5.2。跳过系统模型的可调度性分析

跳过@@

任务模型背景下的可行性问题等于寻找可以跳过的最后期限，以免受到薄弱的硬约束

违规了，剩余的工作可以安排在最后期限之前完成。Koren和Shasha在 [47] 中证明，从薄弱的意义上讲，确定一组定期偶尔可跳过的任务是否可以调度并不难。此结果对应于一种一般情况，即任务在发布时可以是红色或蓝色。但是，可调度条件是通过假设特定情况得出的，即深红色的条件，即所有任务同时发布，每个任务的初始状态为红色。深红色条件是跳过模型的最坏情况假设；如果任务集在深红色条件下是可调度的，则在没有这个假设的情况下也可以调度。Koren和Shasha为一组可跳过的任务的可调度性引入了必要条件 [47]，如定理1所示

。 定理 1.

给定一套

允许跳过的定期任务，然后

(10)

是实现可行性的必要条件

，因为该总和表示基于必须进行的计算的利用率。

充分的条件是通过处理器需求标准构建的，即当且仅当所有间隔都可调度一组定期任务

[85]：

(11)

不等式@@

的右边是间隔内的处理器需求

，其中每项任务

确切地发布

酋长国

需要处理器时间为的作业

单位。如果间隔内的累积处理器需求

小于 L，所有工作都将在截止日期内完成。Koren和Shasha将这个条件扩展到了跳过系统模型，他们引入了足够的

调度条件： 定理 2.

给定一套

允许跳过的定期任务，然后

(12)

是可行性的充分条件

，在哪里

是任务所需的 CPU 时间

在时间间隔内

。

(13)

值得注意的是，要检查的 L 的最大值是超周期

lcm

而不是

lcm

因为算法在结尾处跳过

间隔。因此，如果时间间隔内计划是可行的

，在任何间隔内都是可行的。

5.3。广义弱硬实时系统模型的可调度性分析

广义弱硬约束的可调度性测试基于验证是否

-通过以最坏情况的执行时间执行作业获得的模式满足给定的弱硬约束。在一般情况下，

-pattern 是一个无限二进制序列，不适合执行可调度测试。在 [4] 中，作者证明了

如果任务集的总利用率不大于一，则模式是封闭的，即由单词的循环重复组成。因此，[81] 中提出的离线可调度分析仅适用于以下任务集

。可调度测试扩展到必须考虑任务间阻塞因素的系统，例如优先级上限协议 [86]。为了构建

-pattern，最坏情况下的完成时间

必须计算每项任务。

是最低限度

这使得以下方程成立：

(14)

Intf

空闲

哪里

Intf

是来自优先级较高的工作的干扰吗 t 和

空闲

是优先级低于以下优先级的任务可以使用处理器的时间

在一段时间内

。

可以通过以下方程中给出的递归公式来计算：

(15)

Intf

空闲

复发在以下情况下结束

。这个

-pattern 是根据 (5) 中的表达式构造的，它会评估是否

-pattern 满足给定的约束。这种方法的缺点是，它依赖于这样的假设，即所有任务的初始偏移量都是已知的，这可能过

于严格。

允许放宽了解初始系统状态要求的结果是在弱硬实时系统的过载条件的研究中得出的。提出的方法依赖于 [87] 中介绍的方法，该方法将系统的最坏情况行为分析为典型行为（即执行定期任务和零星过载）的组合。在 [22,88] 中，这种方法应用于弱硬实时系统分析。拟议方法的主要思想包括两步法：

使用经典分析检查系统在典型行为下是否可调度；

检查是否给定的

在过载条件下满足约束。

[25] 中克服了知道显式初始系统状态的局限性。作者提供了一个通用框架，用于在单处理器系统的弱硬约束下调度定期任务。所提出的方法基于混合整数线性规划（MILP）问题公式。该框架已扩展到具有共享资源和任务集的实时系统，任务集具有发布抖动，即作业实际激活时间相对于理想的定期激活时间的最大可能延迟。在 [48] 中，通过分析裁员策略来扩展这项工作，即在错过最后期限时终止工作，而不是继续执行工作（如

继续工作策略）。

6。弱硬实时模型在控制系统设计中的应用

本节概述了控制和调度协同设计的现有方法。该领域的研究主要集中在两个主要问题上：

稳定性分析：确定控制任务的时机约束，确保控制回路的稳定性；

最佳控制系统设计：利用薄弱的硬实时约束来降低过载条件下的系统利用率，同时旨在最大限度地提高控制性能。

在介绍该领域最重要的研究结果之前，概述了控制理论的基本概念。

6.1。控制理论基本概念概述

控制系统的基本组成部分是控制回路。控制回路执行三种功能：传感器采样、控制算法计算和向执行器发送输出命令。控制回路的功能通常定期执行，恒定的采样周期由过程动态决定。图 11 说明了控制回路的时序限制。理想情况下，采样和命令输出是在精确的时间段进行的。采样时间

smpl

由采样周期决定，

smpl

:

smpl

smpl

。在实际应用中，采样周期存在偏差，即采样抖动。命令输出时间取决于输入输出延迟，也称为控制延迟，应尽可能小。通常，与估计的输入输出延迟存在偏差，即延迟抖动。输入输出延迟或抖动的增加会降低控制系统的稳定余量。在 [2] 中，描述了如何设计控制器以主动补偿采样抖动，并通过重新计算控制器参数来减少定时不确定性的影响。但是，在更复杂的控制系统中，不适合应用此类技术，依赖实时调度理论中的机制更为合适。在计算机驱动的控制系统中，控制回路被实现为实时任务。从实时调度的角度来看，目标是配置任务并设计调度算法，以最大限度地减少输入输出延迟和抖动。另一方面，从控制理论的角度来看，目标是优化控制性能。衡量控制性能的常用指标是性能指数，其定义如下 [89]：

定义 8.

性能指标是衡量系统性能的定量指标，之所以选择性能指标，是为了将重点放在重要的系统规格上。

绩效指数是通过以下三种衡量标准之一计算得出的：

积分平方误差，ISE：

;

积分绝对误差，IAE：

;

积分时间加权绝对误差，ITAE：

。

哪里

是错误信号，对应于控制系统的输出与所需输出之间的偏差。控制系统的设计方法通常旨在将性能指数降至最低。

在许多实际应用中，由于其简单性和对现实场景的适用性，使用不太精确的指标来衡量控制性能。接受错误信号的指标

考虑的是稳态误差

：

(16)

lim

无穷大

通过系统的阶跃响应可以分析多个指标：延迟时间、上升时间、过冲、稳定时间等。延迟时间是指经过的时间

到步进响应达到其最终值一半的时刻。上升时间是步进响应从上升所需的时间

到

其最终值（或

到

90

）。过冲定义为最大值与步长值的偏差。步进响应达到稳定状态所需的时间定义为稳定时间。所描述的指标如图 12 所示。

6.2。弱硬实时模型在控制系统中的应用综述

[90,91] 中介绍了一些最早结合经典控制理论和实时约束的方法。这些方法引入了硬截止日期的概念，为控制系统留出了状态空间，以定义控制系统的时间约束。控制任务的硬性截止日期源于受控工厂的渐近稳定性条件。在这种情况下，硬性截止时间与控制任务的时间段无关，它是延迟抖动的临界值，超过该值系统会离开其允许的状态空间并变得不稳定。通过削弱硬实时限制，可以将允许的状态空间拆分为性能空间和宽限空间，如图 13 所示。性能空间的极限由传统的硬实时约束决定，而宽限空间的极限由弱的硬实时约束决定

。

当控制任务的所有工作及时完成时，系统就进入了性能空间。如果不是控制任务的所有工作都已完成，但稳定性条件产生的弱硬约束条件得到满足，则系统会发现自己处于宽限空间。表演空间和优雅空间构成了一个可容忍的空间，该空间受弱的硬约束约束

。

关于通过放松硬实时约束来解决实时控制系统中的过载条件的初步研究是基于调整控制器的采样周期（任务周期），使所有任务均可调度并优化系统性能指数 [92]。作者提出了一种确定任务周期的算法，使任务可以在EDF和速率单调调度计划（RMS）下进行调度。这种方法的新颖之处在于，它不仅考虑任务的可调度性，而且还优化了控制系统的性能。Shin和Meissner演示了一种在多处理器系统中在线应用这种方法的方法 [

93]。

[94] 中介绍的研究侧重于推导实时控制任务的时间限制，以保证可调度和控制性能规范。控制性能是通过稳态误差、最大过冲、稳定时间和上升时间来衡量的。应用启发式方法来优化性能，并在循环处理周期和输入输出延迟之间实现权衡

。

Buttazzo 等人 [95] 引入了一种新的任务模型，即弹性任务模型，以增加硬实时约束的灵活性。在该模型下，任务周期被赋予弹性系数，该系数指定了任务在一定范围内调整其周期的能力。在过载条件下，任务可以缩短其周期，从而降低其利用率。这样，即使在过载条件下，系统也能保持可行性。

由于调度和控制协同设计领域的主要研究问题是设计一种调度器，在将CPU利用率保持在所需值的同时优化控制性能，因此研究人员引入了将调度器本身解释为控制器的方法 [96]。受控变量是任务的最后期限错过率，控制信号是请求的 CPU 利用率。对于控制QoS的多媒体应用程序，也提出了类似的方法 [97,98]。在 [99] 中，在单个CPU上调度控制任务的问题被形式化为线性二次方（LQ）控制问题，其目的是最大限度地降低全局控制成本（所有任务的相应控制成本的总和），但须遵守由可调度条件确定的约束。这种方法是一个非线性编程问题；因此，由于嵌入式实时系统的计算资源通常有限，因此对计算要求过高，无法在嵌入式实时系统中实现

。

[100] 中研究了延迟抖动对控制系统稳定性的影响。作者提出了控制任务的抖动余量的概念，该概念定义了控制回路变得不稳定之前可以容忍的延迟抖动量。此外，作者引入了一种控制和调度协同设计方法，该方法可以保证在过载下每项任务的性能降级相同

。

弱硬实时约束包含在 [101] 中的实时调度和控制协同设计中。这项工作中提出的方法侧重于提供一种保证最坏情况性能要求的机制。它依赖于可加速控制任务的概念。每当执行一次及时完成时，可加速的控制任务都会降低性能指数的价值。换句话说，控制任务的执行次数越多，控制性能越好。在所提出的方法中，应用贝尔曼最优性原理设计可加速控制任务——目标是假设及时执行和跳过的作业的最坏情况分布，即最坏情况下的执行序列，即最坏情况下的执行序列，从而找到一个能最小化性能指标的控制序列。最坏情况下的执行顺序对应于

-根据可调度条件 (7) 假设最坏情况构造的模式，即只有调用次数的强制性任务

酋长国

将在最后期限之前完成任务。因此，强制性任务可以保证稳定性，而可选作业如果按时执行，则可以提高性能。在 [31] 中，这种方法应用于共享单个控制区域网络 (CAN) 的网络控制系统。另一种方法是找出可以容忍的最大连续错过截止日期，[102] 对此进行了研究。作者将所提出的方法应用于飞机俯仰控制的案例研究。

赫特内克等人 [103,104] 分析了非线性控制系统的稳定性。在这项工作中，使用了弱硬实时约束对控制系统进行建模，这些控制系统偶尔会由于反馈通道故障而以开环方式运行，控制回路执行失败对应于错过最后期限。控制回路执行失败受以下因素的限制

限制。所提出的方法基于 [105] 中介绍的一种技术，该技术限制了连续控制回路执行失败的最大次数（连续失败的最大次数）。关键区别在于，[103] 中提出的技术使用弱硬实时约束来推导稳定性条件，而不是控制回路执行失败的最大次数，这会导致稳定性条件不那么保守

。

Pazzaglia等人 [106] 提出了一种优化控制器设计策略，该策略明确考虑了错过最后期限的可能性——截止日期误差感知控制器（DMAC）。所提出的方法考虑了比任务最坏情况响应时间短的任务周期，并根据对任务执行时间的概率分析来估计错过最后期限的情况。分析中考虑了三种不同的处理截止日期错过的策略：中止错过截止日期的作业（job-kill），允许任务继续但跳过下一个作业，让所有作业执行直到完成，但限制包含待处理任务的队列的容量。在 [107] 中，一项稳定性分析

执行了弱硬实时约束。作者分析了几种处理截止日期错过的策略。从控制角度来看，在错过最后期限时，控制信号可以设置为零，也可以设置为前一个周期的值。另一方面，从任务管理的角度来看，错过最后期限的工作可以中止（裁员策略），也可以允许继续（继续工作策略）。稳定性标准是针对每种策略得出的。

Linsenmayer等人最近的研究 [13] 是第一项关于受弱硬实时约束的线性控制系统稳定性的研究，该研究不仅考虑了时间触发的稳定，还考虑了事件触发的稳定。该方法基于 [108] 的结果，其中推导了强大的事件触发控制策略。这种方法的重点是网络控制系统；但是，它可以转化为嵌入式控制系统的问题。

随着网络控制系统的发展，例如成群的AMR或现代飞机中的冗余和分布式飞行控制器，对非周期采样系统的研究已经出现。网络系统中的采样时间取决于采样抖动、数据包丢失以及由于每个控制器上的控制算法与实时调度算法之间的交互而导致的不确定性 [109]。[110] 中介绍了对非线性系统中不同采样周期影响的稳定性分析。[111,112] 中描述了一种基于模糊建模的方法。在 [113] 中，引入了具有非周期采样的线性时间不变（LTI）系统的稳健稳定框架。[114,115] 中介绍的研究使用弱硬实时约束来描述网络控制系统稳定性分析中错过的最后期限。作者推导了给定系统矩阵的稳定性条件和弱硬约束

。Ahrendts等人提出了一种在网络系统背景下验证弱硬实时约束的方法 [10]。该方法扩展了 [22] 中介绍的现有技术，以支持多种资源 (CPU)。汽车案例研究证明了所提出的方法的适用性。在 [116] 中，作者提出了一种自适应控制方法，该方法可以处理由于消息传输延迟而发生的超限。超限处理策略在运行时会根据延迟而改变，并实现了三种不同的操作模式：正常模式、中止模式和跳过

模式。

近年来，控制和调度协同设计问题也在信息物理系统的背景下得到解决。系统设计的主要目标是最大限度地提高控制性能。[117] 中介绍的研究侧重于优化过载条件下的控制性能（最小化性能指数），它引入了一种仲裁算法，该算法接受或跳过作业以降低系统利用率。在 [33] 中，引入了一种新的信息物理系统任务模型，该模型允许通过采样周期和最大可容忍的连续错过截止日期次数来限制系统的稳定性和控制性能。引入了两种调度机制：离线参数分配和在线状态感知调度。离线参数分配决定了采样周期，并对最大错过截止时间次数的弱硬实时约束，从而保持了系统的稳定性。在线状态感知调度使用离线参数分配确定的约束条件，并动态管理错过截止时间以提高系统性能。该调度方法以 [118] 中提出的类似方法和速率单调方法为基准。结果表明，所提出的方法在性能和可调性方面都优于先前提出的方法。在 [15] 中，引入了一种新的信息物理系统任务模型，该模型能够表达底层控制系统的稳定性。设计了相应的调度算法，该算法保证了系统的稳定性并提高了控制性能。[119] 中介绍的一种信息物理系统设计的新方法引入了另一个目标——资源效率。资源效率是一个指标，它取决于在特定时间段内可在给定资源上调度的任务数量。在这项工作中，控制性能是通过稳定时间来衡量的。对经典的实时任务模型进行了扩展，以支持两种控制任务的操作模式：慢速模式和快速模式。总体思路基于这样一个事实，即更长的采样周期可以改善系统的调度性，但这会导致控制性能变差甚至不稳定。[120] 中描述了类似的想法。在 [119] 中，每项任务可接受期限的上限和下限，

和

，定义如下：

-周期值，超过该周期值，控制系统的输出响应将不可接受；

-确保任务利用率的时段值

不会超过允许的最大值。

切换每项任务的操作模式的时刻取决于导致过载的零星事件的最短到达间隔时间。所描述的任务模型被称为双模式任务模型。双模模型的任务参数由基于遗传算法的启发式优化方法确定。对双模遗传算法获得的结果进行控制性能评估，并与传统均匀采样模型和具有随机任务参数分配的双模策略获得的结果进行比较。结果表明，双模遗传算法可将控制性能提高多达

与传统模式相比。在 [14] 中，作者提出了一种在弱硬实时约束下验证和验证信息物理系统的方法。自动驾驶汽车案例研究证明了该方法的适用性。

表 4 总结了本节中审查的调度和控制系统协同设计方法。概述了主要目标、目标应用程序和功能。

尽管它们与弱硬实时系统模型无关，但调度和控制协同设计领域的最新结果值得一提。[121] 中提出的研究调查了2型T-S模糊系统的调度算法和控制器协同设计。随机调度算法是通过粒子群优化设计的。在 [122] 中，提出了无线网络控制系统的估计-控制-调度协同设计框架。控制器和调度器优化是基于深度强化学习技术进行的。[123] 中提出的研究涉及时间敏感的网络。作者提出了一种固定优先级的调度方案和一种优化方法，该方法可以找到最佳采样周期，从而最大限度地减少控制器的建立时间。在 [124] 中，引入了一个新概念，即子调度。这个概念用于描述不可行但可以在相应截止日期之后的有限时间间隔内安排的任务。使用源自子调度概念的时序模型，作者开发了一种模型预测控制方法，该方法可以找到优先级分配方案，从而在满足子调度条件的同时对性能指标进行优化

。

7。结论

本文简要概述了弱硬实时系统和从弱硬时间约束中得出的系统模型。此外，还介绍了在弱硬实时约束下进行可行性分析的技术和调度任务的算法。概述了控制和调度协同设计方面的挑战，以及该领域的最新成果。在弱硬实时系统中调度任务的背景下，存在一些研究问题：提供可调度测试以确保在给定弱硬约束下的任务集是可调度的；提供降低动态故障概率的机制；设计最大限度地提高服务质量的调度算法等。先前针对这些问题的研究朝着两个方向发展：有保障的方法和尽力而为的方法。在有保障的调度方法的背景下，进一步的工作可以包括通过在调度可跳过任务的启发式设计中采用优化技术来提高服务质量。通过考虑特定的任务模型，例如带有谐波周期的任务集，可以扩展提供可调度测试的现有方法。在尽力而为算法的背景下，可以研究采用工作接受启发式方法的可能性，目的是将动态失效的概率降至最低。此外，大多数现有的调度弱硬实时系统的方法都集中在

仅限约束，因此，未来的研究可以解决支持其余弱实时约束的调度算法的实现问题。通过依赖控制系统稳定性分析和控制器设计中的弱硬实时约束的新方法弥合了实时调度领域与控制系统设计之间的研究差距。但是，在控制和调度协同设计方面存在着开放的挑战。所介绍的大多数技术仅侧重于控制性能优化，而考虑其他目标，例如资源利用率（服务质量）将是有益的。