学校代号 10532 学 号 S151000796

分 类 号 TP301 密 级 普通

湖南大學
HUNAN UNIVERSITY

硕士学位论文

新一代汽车电子系统功能安全可靠性目标保障研究

学位申请人姓名 袁娜

培 养 单 位 信息科学与工程学院

导师姓名及职称 李仁发 教授 曾庆光 教授

学 科 专 业 信息与通信工程

研 究 方 向 嵌入式系统与 CPS

论文提交日 期 2018 年 5 月 8 日

学校代号：10532

学　　号：S151000796

密　　级：普通

湖南大学硕士学位论文

新一代汽车电子系统功能安全可靠性目标保障研究

学位申请人姓名：　袁娜

导师姓名及职称：　李仁发 教授 曾庆光 教授

培　养　单　位：　信息科学与工程学院

专　业　名　称：　信息与通信工程

论 文 提 交日 期：　2018 年 5 月 8 日

论 文 答 辩日 期：　2018 年 5 月 20 日

答辩委员会主席：　赵欢 教授

Research on Reliability Goal Assurance of Functional Safety towards the New Generation Automotive Electronic System

by

YUAN Na

B.E. (China University of Mining and Technology) 2015

A thesis submitted in partial satisfaction of the

requirements for the degree of

Master of engineering

in

Communication and information engineering

in the

Graduate School

of

Hunan University

Supervisor

Professor Li Renfa

Professor Zeng Qingguang

May, 2018

湖 南 大 学

学位论文原创性声明

本人郑重声明：所呈交的论文是本人在导师的指导下独立进行研究所取得的研究成果。除了文中特别加以标注引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写的成果作品。对本文的研究做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律后果由本人承担。

作者签名:　　签字日期:　　年　　月　　日

学位论文版权使用授权书

本学位论文作者完全了解学校有关保留、使用学位论文的规定，同意学校保留并向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅。本人授权湖南大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。

本学位论文属于

1、保密□，在____年解密后适用于本授权书

2、不保密□。

(请在以上相应方框内打"√")

作者签名:　　签字日期:　　年　　月　　日

导师签名:　　签字日期:　　年　　月　　日

摘　要

汽车嵌入式系统固有的异构性、交互性和多样性要求信息世界(网络计算)和物理世界之间有着紧密的交互和深度的融合，使得新一代汽车电子系统成为典型的汽车信息物理系统（Automotive Cyber Physical System，ACPS）。作为高度安全关键的系统，ACPS有着系统级的安全可靠要求。同时，为应对汽车事故风险提出的功能安全以及专门针对汽车电子系统发布的功能安全标准ISO 26262 更是将汽车电子系统的安全可靠性需求提高到了一个新的高度。因而保障汽车电子系统的可靠性问题是长久以来的研究重点与难点。基于此，本文面向新一代汽车电子系统功能安全问题，以保障基于DAG模型的分布式汽车功能可靠性目标为主要目标，提出了高效非容错和容错两种情况下的可靠性目标保障方法，主要工作如下：

首先，针对新一代汽车电子系统复杂的系统结构以及分布式汽车功能任务之间的复杂关系，将汽车电子系统的异构计算单元以及功能应用进行模型抽象，以更精确的模型对其进行描述。同时，针对基于DAG模型的可靠性目标调度问题，对调度方法进行归类总结并对可靠性目标调度研究进展进行详细阐述。

其次，提出一种非容错情况下基于几何平均值的可靠性目标保障方法RGAGM。该算法引入数学上的几何平均值定义，采用预分配机制将功能的可靠性目标转换成每个子任务的可靠性目标。当满足了任务的可靠性目标，将任务分配至具有最小资源消耗成本的ECU上。RGAGM 算法利用几何平均值所具有的中心集中趋势，解决了现有算法针对未调度任务预分配可靠性最大值造成资源浪费，分配较小值无法满足可靠性目标的悲观性问题。

最后，提出一种基于几何平均值的容错可靠性目标保障方法GMFRP。该算法旨在采用容错技术，在满足分布式汽车功能可靠性目标的同时，最小化系统的响应时间。GMFRP同样引入数学上的几何平均值，定义容错情况下的未调度任务的可靠性预分配值并得到每个子任务的可靠性目标。迭代的分配备份副版任务至具有最小完成时间的ECU上，直至满足任务的可靠性目标。GMFRP算法利用几何平均值所具有的中心集中趋势，解决了现有算法分配给高优先级和低优先级任务的可靠性值不平衡的问题。

通过真实汽车功能和任务图生成器随机生成的任务图仿真实验表明，与现有算法相比，以上算法都能从整体上满足功能所认证的可靠性目标并优化其目标性能，是高效的可靠性目标保障方法。

关键词：汽车电子系统；功能安全；DAG调度；可靠性目标保障；几何平均值

Abstract

Due to the inherent heterogeneity, interaction, and diverse nature of automotive embedded systems, it requires the joint and tight interaction between the cyber(networked computational) and physical worlds, making the new generation automotive electronic systems a typical automotive cyber physical system (ACPS). As a highly safety-critical system, ACPS has system-level safety and reliability requirements. At the same time, the functional safety proposed in response to the risk of automotive functional failures and the functional safety standard ISO 26262 specifically targeted at automotive electronic systems have raised the safety requirements of the automotive electronic systems to a new level. Therefore, assuring the reliability of the automotive electronic systems is a long-term research focus and difficulty. Based on this, from the perspective of functional safety of the new generation automotive electronic systems, this paper aims to assure the reliability goal of a distributed automotive function based on the DAG scheduling model, and proposes effective reliability goal assurance methods under non-fault-tolerant and fault-tolerant manners, respectively. The main works are as follows:

First of all, aiming at the complex system structure of the new generation of automotive electronic systems and the the complex relationship between the tasks of distributed automotive functions, the heterogeneous computing units and distributed automotive functions should be abstracted and described by more accurate models. At the same time, for the reliability goal scheduling problem based on DAG model, we categorize and summarize the scheduling methods and elaborate the progress of reliability goal scheduling research.

Secondly, we propose a method called Reliability Goal Assurance using Geometric Mean (RGAGM) under non-fault-tolerant manner. The algorithm introduces the mathematical definition of the geometric mean, and uses the pre-allocation mechanism to transfer the reliability goal of function to that of each task. On the basis of satisfying the reliability goal of each task, we assign the task to the ECU with the minimum resource consumption cost. The RGAGM algorithm utilizes the central tendency of the geometric mean, thereby solving the problem that the existing algorithms cause a waste of resources for pre-assigning maximum reliability values to unassigned tasks, and fail to satisfy the reliability goal of function for pre-assigning minimum reliability values to unassigned tasks.

Finally, we propose a method called Geometric Mean-based Fault-tolerant Reliability Pre-assignment (GMFRP) under fault-tolerant manner. This algorithm is designed to use

fault-tolerance mechanism to minimize the response time the system while assuring the reliability goal of a distributed automotive function. GMFRP also introduces a mathematical geometric mean and defines pre-assigned reliability value for each unassigned task, and then confirms the reliability goal of each task. Then, GMFRP iteratively assigns the backups of each task to the ECU with minimum earliest finish time until assuring the task's reliability goal. The GMFRP method utilizes the central tendency of the geometric mean to solve the problem of unbalanced reliability values pre-assigned to the high-priority and low-priority tasks.

Experiments on the real-life automotive function and the randomly generated distributed automotive functions by the task graph generator show that compared with the existing algorithms, the above algorithms can meet the certificated reliability goal of a distributed automotive function and optimize the function's target performance metrics. Therefore, the two methods are effective reliability goal assurance method.

Key Words: Automotive electronic system, Functional safety, DAG scheduling, Reliability goal assurance, Geometric mean

目　录

插图索引

附表索引

第1章　绪论

汽车电子系统是嵌入式系统方向非常具有发展趋势的研究课题，是与国民生活和社会发展息息相关的关键科技问题。新一代汽车电子系统随着其异构性、分布式及集成化的发展，并因其与外部物理系统的紧密交互性，被称为汽车信息物理系统（Automotive Cyber Physical System，ACPS）。同时，功能安全（Functional Safety）的提出以及道路车辆-功能安全（Road Vehicles-Functional Safety）规范标准ISO 26262的发布不断提升了汽车工业对汽车安全的重视高度。因而保障ACPS功能安全所要求的可靠性目标对确保汽车安全运行十分关键。

1.1　研究背景及意义

1.1.1 汽车电子系统概述

汽车在诞生之初，只是简单的机械车概念，通过驾驶员手动操作其中的启动装置，发动机随之燃烧燃料向其他装置输送动力，最后在一系列传输设备的作用之下使汽车运动，因而机械和动力学等研究领域推动着传统汽车的发展。随着电气、电子以及软件技术的发展，汽车逐步发展到以分布式的电子控制单元（Electronic Control Units, ECUs）控制整车功能部件运行的汽车电子系统（Automotive Electronic Sytem）。随着人们对汽车的安全性和驾驶性能的日益追求，汽车电子系统不仅各类功能应用越发完善，系统结构同样变得更加精细。普通汽车一辆车身内有多达100个异构ECU，同时也包括各类用以采集数据的传感器以及用以执行命令的执行器。这些ECU通过CAN(Controller Area Network)、FlexRay、LIN(Local Interconnect Network)、MOST(Media Oriented System Transport) 以及Ethernet等多种异构网络总线实现互联及通信，而总线之间通过一个中央网关集成，整个系统能实现交互与协作功能且异构特征明显[1]。汽车电子系统在不断地刷新它的应用能力，突破单一的机械式代步工具的传统定义，以“会走的计算机”的形式为人类生活提供更便捷的载体。

如图1.1所示汽车电子系统的系统结构，以各类传感器为主的物理设备通过与物理环境进行交互来获取外界的环境信息，这些信息转换为实时数据通过CAN、FlexRay、Ethernet等网络总线进行传输至相应的ECU 计算单元进行处理，进而将处理后的执行命令发送至执行器，做出正确的行驶、刹车等等动作[2]。汽车电子系统作为一个复杂的异构分布式嵌入式系统，表现出如下几个特点：

（1）体系结构复杂。如上所述，系统中较大数量的异构ECU、各类传感器以及执行器组成多元化的物理处理单元，即大量异构的物理设备。大量不同性能参

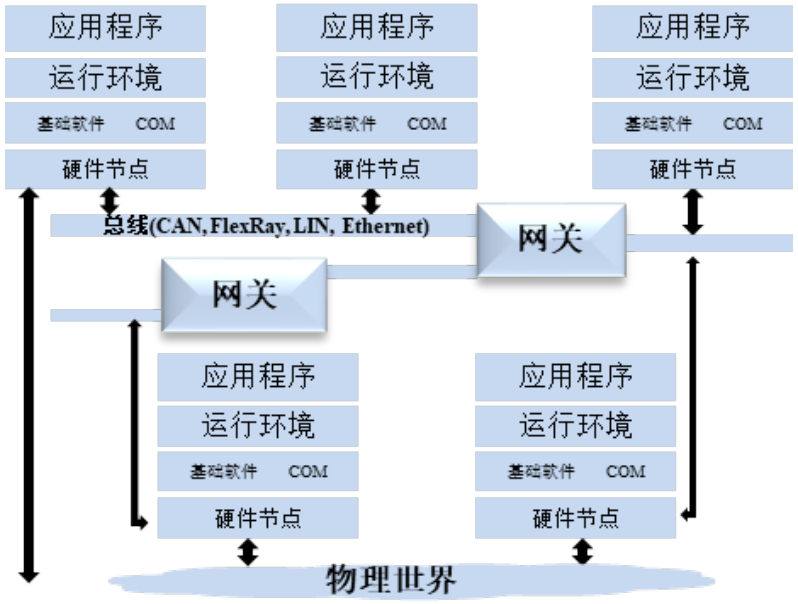

图 1.1 汽车电子系统的系统结构

数的CAN、FlexRay、LIN等网络总线实现互联，即异构的网络。各系统及子系统所需求的响应时间完全不同，即异构的实时性需求。另外从控制层面上来说，最初机械式的手动操作功能随着发展的过程逐步被电子化功能所取代，例如传统的线控刹车和线控转向功能，逐渐实现电动化，甚至有些功能如自适应巡航系统都能根据收集的信息协助驾驶员进行控制。控制功能的日益智能化使得体系结构更为复杂。

（2）由复杂子系统和分布式功能高度集成。在系统方面，汽车电子系统包括能量控制子系统、自动悬挂系统、主动和被动安全子系统以及车身控制子系统等多个子系统，每个子系统又可包括多个分布式功能。在功能方面，一个汽车电子平台可同时集成主动安全功能、被动安全和非安全关键功能等不同级别的功能，从而实现功能可跨越子系统的特性。一个分布式汽车功能包含许多任务以及各种通信消息，而不同的功能通过各类网络总线实现交互与反馈，功能的任务之间又可以共享车内的物理处理单元。因此，各复杂子系统以及分布式功能的高度集成使得一个功能可以跨越不同的子系统，而一个ECU 又可以支持多个功能的执行。

（3）需与物理环境具有联合且紧密的交互。汽车电子系统上配置了多种类型的环境感知设备，利用这些传感器对车内以及车身环境进行实时的检测，采集车内数据以及车外实时信息，通过网络总线与计算单元进行信息传输与交互，最后向物理执行设备发送执行命令反馈给物理环境，因而新一代汽车电子系统的每个动作的执行都离不开与物理环境的进行数据之间的实时通信和交互。

（4）安全可靠性需求。安全可靠性是汽车非常严格的要求，所以汽车电子

系统对每个功能的安全有着高度的需求。例如，当发生紧急情况驾驶员踩下刹车时，系统必须及时对刹车动作作出响应，太晚就会造成不可估量的后果；当发生碰撞时，安全气囊的打开也需要高度的可靠，过早或过晚都不能发挥其事故保护作用。因此，对于动力控制、底盘控制、主动与被动安全功能等与安全相关的功能而言，安全可靠是关键需求。

综上特点所述，汽车电子系统将计算系统、网络系统和控制系统进行了整合，形成了一个比较大型且复杂的系统，该系统实时的和车身外的环境通信和交互，从而使得汽车更安全的行驶。因此新一代汽车电子系统具备了信息物理系统（Cyber-Physical Systems, CPS）[3]的基本特性，可以说是一个非常典型的CPS。

1.1.2 ACPS角度研究新一代汽车电子系统

CPS是指基于并依赖计算算法和物理组件无缝集成的工程系统，它由一些能够相互通信的计算设备组成，这些计算设备能够通过传感器和作动器与物理世界实现反馈闭环式交互[3]。CPS作为新兴的领域，在全球工业领域迅速发展并占用重要地位，其无所不在，并且发展越来越猛，从智能建筑到医疗设备再到汽车工业，都成为其广泛且渗透的应用之处从交通运输、制造业。对于汽车电子系统来说，嵌入式技术在汽车上得到大量使用，如防抱死制动系统(Anti-lock Braking System, ABS)、电子稳定程序(Electronic Stability Program, ESP)、电子制动力分配(Electronic Brake-force Distribution, EBD) 等车载嵌入式应用遍及现代汽车应用之列。这些功能的实现都需要与物理环境通过多个ECU以及车内网络进行交互、反馈和协调来完成。同时，汽车电子系统本身具有的异构性（异构ECUs与异构网络）、交互性（传感器和执行器与物理世界进行实时交互）和分布性（网络分布和应用分布）要求其与外部物理系统具有联合且紧密的交互（joint and tight interaction）[2]。因而，新一代汽车电子系统这一典型的CPS也称为ACPS（Automotive CPS）。我们将新一代汽车电子系统作为ACPS的交互原理描述如图1.2 所示，通过汽车计算设备、无限通信以及汽车控制中心这一3C技术集成于一体，将物理层的多种智能设备与物理环境，信息层的软件与网络等进行融合相互作用，从而实现所有信息的交互与决策。

ACPS体系结构不断日益复杂的演化，新一代ACPS成为时间关键、安全关键、资源关键和性能关键四大关键特性于一体的异构分布式嵌入式系统。作为典型的ACPS，新一代汽车电子系统应具有以下特性：

首先，追求信息系统与物理系统的紧密融合，实现两者之间的互通互信互控。虽然汽车是人们生活所普遍使用的交通工具，但是汽车在执行过程中的感知数据、对数据进行计算处理进而执行输出命令这一过程要求汽车必须随时捕获所处的物理环境的详细信息，并与之进行交互以及深度的融合。

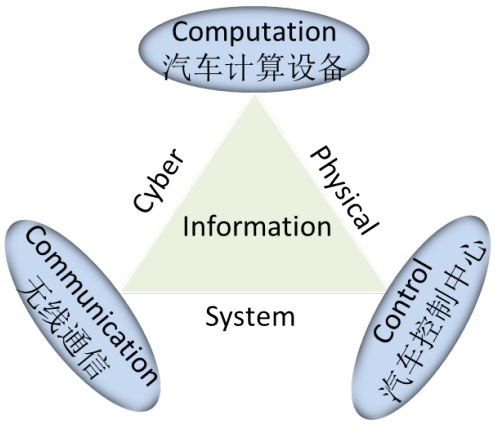

图 1.2 ACPS原理结构

其次，高度系统级别的安全可靠性需求。安全是汽车的最基础也是最主要的要求，任何微小的失误都很可能造成不可设想的后果，在行驶过程中必须保证人员安全和车辆无损，因而ACPS是绝对高度安全关键的系统。作为系统级的要求，安全可靠是指在运行时发生故障或失效时，ECU可以灵活迅速地对任务进行重分配，从而减少故障或失效带来的影响，提高整个系统的可靠性。

最后，必须具备自适应、重配置的能力。ACPS要求能够随着环境的变化而自主地进行动态调整，能够反复利用已有的功能组件实现重组配置，从而降低系统的复杂性并减少开发资源成本。

总而言之，ACPS强调信息系统与物理系统的紧密融合、高度安全可靠以及能自主地进行动态调整的特征，其发展程度成为各个国家衡量汽车信息技术产业发展及创新水平重要指标。其中，ACPS对安全可靠性提出系统级别的高度要求，促使新一代汽车电子系统需尤其关注高度可靠驾驶的保障问题。因此，从ACPS角度研究新一代汽车电子系统，推动汽车电子系统获得更高的驾驶性、安全性与自治性，对推动着汽车工业的高度安全化发展有着现实和长远的意义。

1.2 国内外研究现状

1.2.1 功能安全需求

1.汽车功能安全需求的提出

随着人们对更高驾驶性能与体验的追求，从普通车载功能到各类高端应用，汽车上配备的功能日益繁多。与此同时，汽车电子系统发生系统性失效、随机硬件失效、时序失常和功能执行的逻辑顺序打乱等危险事件发生的可能性也随之提升。因此，安全成为汽车工业长久以来的不懈追求。为了保障安全驾驶，人们最初采用的安全带以及安全气囊等被动安全措施发挥了极大作用。后来人们发展出在可能发生危险之前进行主动式干预的系列主动安全功能，例如防抱死制动系

统、电子制动力分配等。随着新一代汽车网联化、智能化、电动化的发展趋势，高级驾驶辅助系统（Advanced Driver Assistant System, ADAS）的出现与发展大大加强了汽车的安全防护能力[4]。ADAS 将主动安全和被动安全进行融合，通过车身搭载的摄像头、雷达、红外等传感器感知与检测汽车状态变量，然后将数据传至计算单元进行判断和处理，最后执行被动式报警或主动式干预。在人们不断提升ACPS 的安全层次的同时，却仍不可避免汽车突然或偶然性的失效发生，例如车辆意外的加速、减速与转向，安全气囊非正常弹开，高速行驶时车门突然打开等，从而导致相关事故风险的出现。这里的风险即指这些失效发生伤害或损害的可能性，及伤害或损害所造成的后果的严重性。

针对上述可能发生的事故风险，功能安全(Functional Safety)这一需求随之而生。最初功能安全的提出是为了应对石油化工领域的生产过程中相关安全控制系统或者安全功能失效导致的事故，并针对电子、电气、可编程逻辑控制器产品的安全设计提出了功能安全基础标准IEC 61508[5]。派生至汽车领域，为了避免因汽车系统功能性的失效可能导致的不可接受的安全伤害和风险，人们提出了专门针对道路车辆的功能安全需求。2011 年，专门针对汽车电子系统的功能安全标准——道路车辆——功能安全(Road Vehicles-Functional Safety) 标准规范ISO 26262 正式公布并成为国际标准[6,7]，适用于从传统汽车到现今发展迅速的新能源汽车所有汽车范围[6,7]。

2.ISO 26262标准对保障功能可靠性的要求

ISO 26262标准包括功能安全管理、系统开发设计、硬件设计、软件设计、生产运营等10个部分，提供了一个完整的汽车安全生命周期和风险等级的具体评估方法。同时，为对汽车安全提出详细且全面的定义与划分，ISO 26262结合了严重性、暴露率以及可控性三大安全属性提出对汽车安全的完整定义，如表1.1所示。其中，严重性表示功能发生安全事故的伤害的严重程度，从低到高分为S0、S1、S2、S3四个等级；暴露率定义了不同故障发生的概率，分为E0、E1、E2、E3和E4四个等级；而可控性则是指行驶过程中驾驶员的可控性，分为C0、C1、C2和C3四个级别。同时，ISO 26262结合严重性、暴露率和可控性将汽车安全完整性等级（Automotive Safety Integration Level, ASIL）划分为A，B，C，D四个等级。其中，A为最低等级，对安全性要求较低；而D为最高等级，对安全性要求最高。

因此通过上述分析，ISO 26262标准对汽车功能安全有着明确且详细的规定及要求。对于不同功能而言，具有不同的安全性需求，因而需要针对不同功能保障其不同的安全性需求。汽车功能的开发生命周期通常包括分析、设计、实现和测试阶段。ISO 26262 同时也指出，相关生产者在初期分析阶段需要分析系统或功能中所有可能存在的风险，从而在设计阶段采取高效的措施消除这些风险，保障汽车电子系统中以主动安全功能为主的各种功能的功能安全需求[6,8]。

表 1.1 ISO 26262标准对严重性、暴露率及可控性的定义

严重性	暴露率	可控性
S0 无伤害	E0 不可思议概率	C0 普遍可控
S1 不威胁生命的伤害	E1 非常低概率	C1 简单可控
S2 威胁生命的伤害	E2 低概率	C2 通常可控
S3 致命伤害	E3 中概率	C3 难于控制
	E4 高概率	

1.2.2 可靠性目标保障问题

依据ISO 26262标准可知，严重性以及可控性旨在对事故发生的严重程度以及驾驶员状态分类。而暴露率所定义的故障发生的概率是能在设计阶段控制的属性。可靠性则与暴露率成反比(即可靠性=1-暴露率)，指的是对于汽车内的分布式功能，其能在给定使用期限内保持无故障运行的概率，是通常用来评价汽车技术水平的综合性能指标。由于汽车电子系统包含了许多类型不同的子系统，子系统又包括各种功能，功能之间又有相互的依赖与约束关系，这样的复杂程度就使得系统的设计难度较高，而且修改时也会对彼此产生相互影响。其次，正是由于子系统之间的相互关系，如果某个子系统发生了故障，就会引发其他子系统的故障，这种相互影响会使整个系统面临严重威胁。另外，汽车在行驶过程中容易发生电磁干扰、消息丢失，温度较高等问题，这些问题往往就会导致ECU 和通信网络出现一些随机硬件失效的问题，可靠性引起的问题发生频率较高。因此，在经过长时间的使用后，汽车功能出现故障的概率也随之增加。所以评估功能的可靠性，采取有效的可靠性目标保障方法保障汽车安全可靠的运行对新一代汽车电子系统意义非凡。

1.非容错可靠性目标保障问题

面向分布式汽车功能的可靠性目标保障问题是指在保障汽车某一分布式功能认证的可靠性目标的前提下，优化汽车电子系统的系统资源、时间、性能等关键属性，这一问题实质上属于调度问题。ISO 26262 标准指出，暴露率（即对应的可靠性）和响应时间是汽车功能安全两个重要的属性[6]。以标准中定义的暴露率和响应时间为例，解释可靠性目标的调度问题。响应时间需求也称为时间需求、时间约束或期限约束。对于汽车电子系统来说，提升可靠性和减少响应时间都是其需要达到得到目标。然而，实际情况下响应时间和可靠性这两个安全属性相互影响，提升功能可靠性会增加功能最终的响应时间，可靠性最大化和响应时间最小化相互矛盾，很难同时满足响应时间和可靠性需求[9]。因此，基于靠性目标优化另外任何一个功能安全属性，如系统资源、能耗、功耗等性能指标都符合帕累托曲线，即分布式汽车功能可靠性目标调度问题均属于NP 难问题。

对于一个分布式汽车功能来说，无法保障其完全的可靠，只要能够采取有效的保障措施来保障功能所指定要达到的可靠性目标，那么就认为此功能可靠。近年来，不少相关学者专注于面向可靠性目标的调度问题，既证明了基于分布式功能可靠性评估的NP难问题，又提出了多种保障可靠性、可靠性最大化、可靠性与其他目标联合优化、双目标优化等问题的各类调度策略。然而，现有针对非容错情况下的可靠性目标保障方法要么都是针对于独立任务的可靠性目标调度，而不是针对于任务之间有优先级关系的整个功能；要么则针对系统功能提出高可靠性保障调度策略，却都是针对同构计算系统而不是异构计算系统。目前，对于异构计算系统功能的可靠性目标调度问题，相关学者提出一种预分配机制，为可靠性目标保障提供了一种高性能的解决方案。但是这种方式在一定程度上还存在悲观性处理，因而非容错可靠性目标保障方法还有较多的提升空间。

2.容错可靠性目标保障问题

容错（Fault-tolerance）是嵌入式系统保障可靠性目标调度问题的重要技术。它是指当系统内部在出现硬件故障或者软件故障情况时，系统仍能继续正确执行指定任务，并向外界环境提供所需的无差错服务[10]。容错机制主要是对任务或者消息进行复制成多个备份副版本，从而在发生错误时可以采用副版本任务或消息继续完成任务的执行，这就大大提高了功能的可靠性，容错也即为主/副版本技术(Primary/Backup Copy, P/B)[11]。主/副版本技术分为被动复制方式（Passive Backup Copy）、主动复制方式（Active Backup Copy）和混合复制方式（P/B Overlapping Backup Copy）。被动复制是指当主版任务发生故障失效后，才启用某个备份副版任务成为新的主版任务继续进行调度，属于故障恢复的复制方式。主动复制则指主版任务主动复制多个备份副版任务至处理器上，只要有一个备份副版任务执行成功，则整个任务即为成功调度，属于故障屏蔽的复制技术。混合复制则介于被动复制和主动复制之间，有主动副版和被动副版两种副版任务，任务类型则通过各个任务的时间特性来判定。随着容错技术的研究发展，主动复制成为分布式功能可靠性调度的主要技术。

近年来，不少相关学者利用基于任务复制的容错机制，对保障汽车分布式功能可靠性目标情况下优化系统的资源、时间、性能等目标的调度问题进行了深入的研究，有着丰富的研究成果[12,13]。通常根据任务复制次数来区分容错的方式，主要包括单副本的被动复制、固定副版本数量的主动复制以及不定量副版本的主动复制。对于单副本的复制方式，虽然性能较高，但是这种方式仅能容忍1个错误，过于理想化。后来人们对任务复制固定数量的副版本，提出了容忍错误能力强的高可靠性策略，但是这种复制方式复制时盲目，会导致资源冗余问题。不定量副版本的出现逐步解决这之间的问题，通过动态的对每个任务复制不同备份副版本数量实现冗余优化，但是现有策略的贪婪性仍是有待解决的问题。

基于上文所述，在不使用容错技术情况下以及采用容错技术来保障功能可靠性的问题都还存在较多的问题与可优化空间，这也促使本文基于上文所述的背景及意义开展了新一代汽车电子系统功能安全的可靠性目标保障问题的研究。

1.3 本文主要工作

安全关键的汽车功能的开发生命周期通常包括分析、设计、实现和测试阶段。本研究的目标是在设计阶段为分布式汽车功能提供高效的可靠性目标保障方法。本研究创新性的引入了数学上几何平均值的概念，利用几何平均值中心集中趋势的特点提出了非容错情况下和容错情况技术下未调度任务的可靠性预分配机制，并基于这种预分配机制提出了有效的可靠性目标保障方法。本文主要工作总结如下：

1.本文针对新一代汽车电子系统复杂的体系结构与功能的任务间的复杂关系，分别将系统中异构计算单元抽象成中央网关集成的ECU结构、分布式功能抽象成DAG模型。接着分别就调度、非容错可靠性目标调度及容错可靠性目标调度等研究进展及存在问题进行了阐述与总结。

2.本文提出一种高效的可靠性目标下的资源最小化非容错调度方法Reliability Goal Assurance Method using Geometric Mean（RGAGM）。首先，阐述了异构计算的可靠性模型以及作为实验指标的资源消耗成本模型。然后，针对现有方法中未调度任务可靠性预分配值的悲观性，提出一种基于几何平均值的可靠性预分配方法，并基于这种预分配机制提出一种非容错情况下的可靠性目标保障方法。最后通过真实汽车功能与随机生成的分布式功能比较不同可靠性目标以及不同任务数量下的功能实际可靠性及最终资源消耗成本来验证所提出的RGAGM算法的有效性。

3.本文提出了一种高效的容错技术机制下的高效可靠性目标保障调度方法Geometric Mean-based Fault-tolerant Reliability Pre-assignment（GMFRP）。首先，阐述了容错情况下的异构计算的可靠性模型以及作为实验指标的响应时间模型。然后，针对现有方法中未调度任务可靠性预分配值的不平衡性，提出一种基于几何平均值的可靠性预分配方法，并基于这种预分配机制提出一种容错情况下的可靠性目标保障方法。最后通过真实汽车功能与随机生成的分布式功能比较不同可靠性目标以及不同任务数量下的功能实际可靠性及最终资源消耗成本来验证所提出的GMFRP算法的有效性。

1.4 本文组织结构

本文的组织结构如下：

第1章：绪论。概述了汽车电子系统的产生背景与结构特点，并阐述新一代汽车电子系统演化为ACPS 的原理及特点。接着就新一代汽车电子系统功能安全需求的提出、功能安全标准ISO 26262 的发布阐述汽车功能安全的重要意义，并就保障可靠性问题提出本文的研究动机。最后简单叙述了本文的主要研究内容和本文的组织结构。

第2章：相关研究及进展。首先将新一代汽车电子系统中异构计算单元、功能应用进行抽象。然后，对基于DAG模型的任务任务调度算法进行分类，阐述没类算法的思想及经典算法。最后，分别论述非容错可靠性目标保障问题以及容错技术系下可靠性目标保障问题相关国内外研究进展及存在的问题。

第3章：可靠性目标下的资源最小化非容错调度。本文的第一大核心研究内容，针对非容错情况，主要研究可靠性目标下的资源最小化调度。提出了本章的核心算法RGAGM，对算法思想和步骤进行详细阐述，通过实验验证所提出的RGAGM算法。

第4章：容错技术下的高效可靠性目标保障调度。本文的第二大核心研究内容，针对容错情况，主要研究可靠性目标下的响应时间最小化调度。提出了本章的核心算法GMFRP，对算法思想和步骤进行详细阐述，通过实验验证所提出的GMFRP算法。

最后是本文的结论部分，对本文的工作进行总结并对后续研究进行展望。

第2章　相关研究及进展

新一代汽车电子系统正逐步向着更复杂的体系结构、更集成的应用功能发展，这就使得汽车电子系统的的建模问题相比通用的分布嵌入式系统更加复杂。所以，必须采用更形式化的并行与分布式模型，才可以真实与精确的反映汽车电子系统的CPS特征。同时，由于汽车内功能从简单的应用逐步形成复杂化与并行化的功能，简单的模型无法反映复杂的功能特性，这就需要一种更加精确的功能模型来反映功能的特性与任务之间的约束关系。

面向保障汽车电子系统分布式功能的可靠性目标，通过减少系统的资源、时间或性能是典型的调度问题。过去的调度问题通常只是在系统处理能力、电源、存储等有限的条件下，让系统达到可以调度的目的，或是以优化系统的实时性等性能指标为目的。然而作为高度安全关键的新一代汽车电子系统，不能只是简单的考虑系统资源的有限性，更应从汽车功能安全的角度，考虑面向可靠性、可调度性以及实时性等汽车功能安全相关的属性优化[14,15]。汽车电子系统作为分布式的可靠嵌入式系统，可靠性是本研究重要的切入点。本章旨在对汽车电子系统的体系结构及分布式汽车功能的模型进行抽象，并在回顾与归纳调度相关研究策略的同时，详述非容错及容错机制下的可靠性目标保障相关研究进展及存在的问题。

2.1　汽车电子系统调度模型抽象

2.1.1 异构计算单元抽象

汽车电子系统车身内密集分布100多个异构电子控制单元，通过CAN、Flexray、LIN 等多种异构网络总线连接，其中CAN总线是汽车工业中应用最为广泛的总线类型。CAN 是一种半双工、静态优先级和非抢占式调度通信总线，非常适合于分布式汽车系统[16–18]。总线之间通过中央网关互相连接，从而实现信息的互通[19]，因而汽车嵌入式系统是典型的控制器局域网络（CAN）集群[20]。同时，车身内分布多个用以采集本车与车身周围环境的各类传感器以及用以执行命令的各类执行器。如图2.1 所示的汽车电子系统抽象，因为物理过程是由多个并行进程组成的，因而一些ECU连接几个传感器，而另一些ECU连接几个执行器[2]。因此，部分ECU可以通过接收传感器收集到的数据来释放功能，而其他部分ECU可以通过向执行机构发送执行动作来完成该功能[2]。在CAN 集群的体系结构中，多个CAN总线实现互连。当一个任务在ECU 执行完成后，任务将发送通信消息给其所有的后继任务，这些后继任务可能会调度在不同的ECU 上。例如，当任务n_1

在ECU_1上执行完成后，该任务将通信消息$m_{1,2}$传送至位于ECU_4上的任务n_2。在这个架构中，功能的入口任务只能由连接传感器的特定的ECU执行，而该功能的出口任务只能由连接执行器的特定的ECU执行。

图 2.1 汽车电子系统抽象

如上所述，本研究中的异构分布式汽车嵌入式系统是由多个完全连接的异构ECU 组成，我们将这些异构的ECU 描述为$U = \{u_1,u_2,...,u_{|U|}\}$, 其中$U$ 表示系统中的ECU数。另外，对于文中的集合X，我们用X 表示该集合的大小。

2.1.2 功能应用的DAG抽象

汽车电子系统功能繁多，任何一个功能都是为了让整个汽车的驾驶更为方便。以线控刹车功能端到端执行过程为例，作为最基础的汽车功能，包括1 个传感器，2 根CAN总线、5个ECU以及2个执行器。当驾驶人员踩下踏板发出减速或停车指令时，汽车会立即通过传感器立即感知相关环境数据，然后将信息以电子形式传输给ECU 单元，这些ECU通过CAN、FlexRay 等总线实现互联。当ECU 收到传来的数据以后，对其进行处理并传送给其他相关ECU再进行处理，并最终将执行命令发出以实现刹车功能。整个执行过程的所有任务在5个ECU上完成，子任务先后执行且子任务之间又会有优先级约束关系。

现有研究中存在一些时序链[21]、功能链[22]以及任务链[23] 等用以描述简单应用的模型。然而，汽车电子系统功能中的任务不仅需要简单的描述任务个数，更需要描述其任务之间的优先级约束关系，因此上述模型并不适用。对于分布式功能应用，通常都将其抽象为有向无环图(Directed Acyclic Graph，DAG)模型，该模型用$G = \{N, M, C, W\}$ 表示，对该模型的详细描述如下。

（1）$N = \{n_1, n_2, ..., n_i\}$是DAG任务图中的节点集合，表示功能$G$中的任务集。对于一个任务$n_i$，它有前驱任务和后继任务，用$pred(n_i)$和$succ(n_i)$分别表示$n_i$的直接前驱集和直接后驱集。$n_i$只有当其所有的直接前驱任务完成执行后它才能成为就绪任务进行调度，而只有当任务n_i及其后继任务的所有前驱任务完成执行后，

任务n_i的后继任务才能成为就绪任务进行调度。例如，图2.2 展示了一个包含10 个任务的分布式功能示例运行于三个ECU $\{u_1,u_2,u_3\}$ 上。所以对于任务n_8，其直接前驱任务集为$pred(n_8) = \{n_2, n_4, n_6\}$，前驱任务集中的$n_2$、$n_4$、$n_6$全都调度完成后，任务$n_8$才可以开始进行调度；又如对于任务$n_2$，其直接后继任务集则为$succ(n_2) = \{n_8, n_9\}$，任务$n_2$完成调度后，其后驱任务集中的任务$n_8$和任务$n_9$才可以就绪开始调度。其中，任务$n_1$没有任何前驱任务集，而任务$n_{10}$没有任何后继任务集，将这两种任务分别定义为入口任务和出口任务，并分别用n_{entry} 和n_{exit} 表示。如果一个功能有多个n_{entry} 或者多个n_{exit} 任务，则在DAG 图中添加一个零权重依赖的虚拟入口或出口任务。

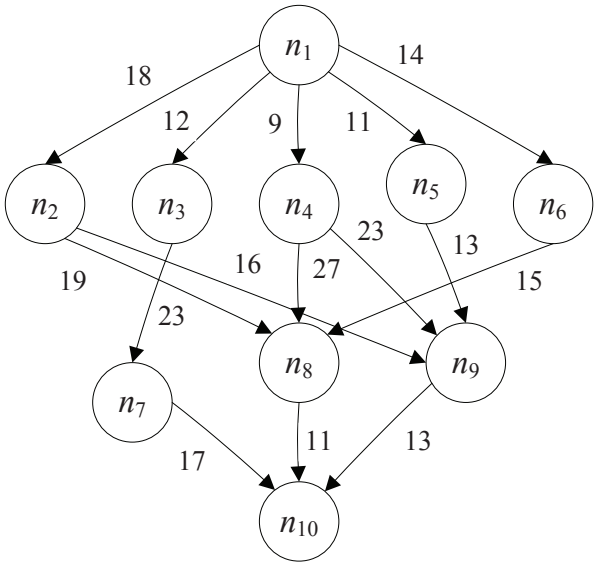

图 2.2 基于DAG模型的分布式功能示例

（2）W 是一个$|N| \times |U|$ 的计算矩阵，$w_{i,k}$指任务n_i 在ECU u_k 上运行的最差执行时间(WECT)。当任务在特定的ECU 中执行时，任务的WCET是所有可能的实际执行时间值中的最大执行时间。由于ECU的异构性，每个任务在不同的ECU 上会有不同的WCET。事实上，如果任务和系统中的其他任务共享高速缓存、内存线和其他硬件功能，则难以测量任务的WCET。本研究我们认为所有任务的WCET已知的，且都是在分析阶段通过WCET 分析确定[24]。表2.1 列出了每个任务在三个ECU$\{u_1,u_2,u_3\}$ 上各自的WCET，n_1和u_2的权重值16 表示了任务n_1在u_2上WCET，即表示为$w_{1,2}$=16。

（3）分配到不同ECU上的任务之间是通过总线传递消息来相互通信的。M 是通信边，$M = \{m_{1,2}, m_{1,3}, ..., m_{i,j}\}$ 表示任务间的通信消息集，且每条边$m_{i,j}$ 代表从任务n_i传递至任务n_j的通信消息。那么，$c_{i,j} \in C$ 指当n_i 和n_j 不在同一ECU上时$m_{i,j}$ 的最坏响应时间(WCRT)。消息的WCRT 是指在特定总线上传输消息时，所有可能的实际响应时间值中的最大响应时间。由于确定任务的WCRT是一个组合爆炸问题，因此本研究中的WCRT是理论上的上界。通常的WCRT分析是在一个伪多

表 2.1 图2.2中分布式功能的任务在不同ECU上的WECTs

Task	u_1	u_2	u_3	$rank_u$
n_1	14	16	9	108.000
n_2	13	19	18	77.000
n_3	11	13	19	80.000
n_4	13	8	17	80.000
n_5	12	13	10	69.000
n_6	13	16	9	63.333
n_7	7	15	11	42.667
n_8	5	11	14	35.667
n_9	18	12	20	44.333
n_{10}	21	7	16	14.667

项式计算时间内估计一个大于或等于精确WCRT 的紧凑WCRT上界[17]。对于一个基于DAG模型的功能中的一条CAN 消息$m_{i,j}$, 它的WCRT值$c_{i,j}$计算公式为:

$$c_{i,j} = \sum_{m_{x,y} \in (M - apred(m_{i,j}) - asucc(m_{i,j}))} e_{x,y}. \tag{2.1}$$

$apred\left(m_{i,j}\right)$和$asucc\left(m_{i,j}\right)$分别表示消息$m_{i,j}$的所有前驱与后继消息。$e_{x,y}$ 表示$m_{i,j}$ 的最坏传输时间（WCTT）。在500Kbps 的CAN 总线中，每条消息的WCTT 都属于$\{150\mu s, 170\mu s, 190\mu s, 210\mu s, 230\mu s, 250\mu s, 270\mu s\}$ 的集合范围内[17]。因为CAN 总线的最大传输速率为1Mbps，因而每条消息的最小时间单位为$1\mu s$，(即CAN总线上1 位的最小传输时间为$1\mu s$)。我们忽略WCRT 的计算，直接认为所有消息的WCRT 都是已知的，且都是在分析阶段通过WCRT 分析确定[17]。

汽车开放系统架构(AUTOSAR)是一种包含应用程序、运行环境(RTE) 以及基础软件(BSW)三层的汽车架构，它详细叙述了CAN网络中ECU内部和之间的通信进程[25]。如果两个任务属于同一个ECU，那么两个任务之间的通信就可以通过RTE提供的通信服务(即共享内存机制)来完成；而如果两个任务位于不同的ECU上，那么两个任务之间的通信就可能需要跨过三层中的不同层次，并最终使用网络来完成之间的信息互通(即消息传递机制)[17]。而这之中的传递时间，比如，在32位和400MHz 的共享存储机制中，对于128 位的扩展CAN 帧，存储器中数据包的最坏情况传输时间(WCTT) 约为0.01s。在速度为500kbit/s 的CAN 消息传递机制中，对于128 位的扩展CAN，总线中数据包的WCTT 约为$256\mu s$。不难看出，共享内存中的速度比消息传递速度快得多，基本可以忽略不计，因此分配给同一个ECU的任务之间的通信时间可以忽略不计。因此，对于本文的任务模型，如果任务n_i 和n_j 分配至同一ECU 上，则通信时间为0。例如，图2.2 中任务n_1

和n_2之间的值9就表示如果两任务分配至不同的ECU 上，两间的WCRT为$c_{1,2}=18$。根据AUTOSAR 标准，调度可以是抢占式调度，也可以是非抢占式调度。考虑到基于DAG模型的分布式功能调度算法通常采用非抢占式调度，且非抢占式调度比抢占式调度更容易跟踪，因而本文只考虑ECU的非抢占式调度。

2.2 研究进展

2.2.1 调度研究进展

基于汽车电子系统功能安全可靠性目标的研究实质上是一种调度问题。任务调度策略包括两个步骤，对任务调度顺序进行排序和给功能的任务分配合理的处理器，从而满足任务优先级要求并获得最短执行时间[26]。当任务的执行时间、任务之间通信数据的大小以及任务之间的约束关系等条件为已知时，我们就认为该功能是一个静态模型。DAG模型是静态任务调度问题的常用功能模型。现有研究已经证明，基于DAG模型的分布式功能任务的静态调度是一个NP 完全问题。静态任务调度算法主要分为启发式算法和随机搜索算法，而启发式算法作为性能最佳的算法类型被广为接受。启发式算法大致分为表调度算法、集群计算和任务复制算法等。本文对可靠性目标保障的研究是采用性能优良的启发式算法，因而主要对启发式算法研究进展进行叙述。

1.表调度启发式静态任务调度算法

表调度启发式算法[27–32]通过对每个任务分配优先级来构建一个有序的任务排序列表。根据任务的优先级排序来选择任务调度，然后每个任务被调度至能最小化预先定义的成本函数的处理器上。这种类型的算法通常需要两个过程：任务优先级排序（或称为任务选择）以选择高优先级的就绪任务，和处理器选择以选择合适的处理器是预定的成本函数最小化。典型的表调度算法如基于临界路径法的静态调度算法the Modified Critical Path （MCP）[27]、动态级调度（Dynamic Level Scheduling）的编译时间调度启发式算法[29]、并行任务调度启发式算法Mapping Heuristic(MH)[30]、并行程序优化方法Insertion-Scheduling Heuristic[31]、非抢占调度的Earliest Time First （ETF）算法[32]以及将任务图分配给全连通多处理器的静态调度算法Dynamic Critical Path(DCP)[28]。这些表调度算法在性能方面表现优异，然而，这些算法大都是针对于同构系统的调度问题。

2002年，Haluk Topcuoglu等人[33] 针对由一定数量的异构处理器组成的异构系统提出Heterogeneous Earliest-Finish-Time（HEFT）算法，如表2.2所示。作者提出一种新颖的向上排序值（upward rank value）来对任务进行优先级排序形成调度顺序，向上排序值$rank_{\mathrm{u}}$ 的计算如公式(2.2) 所示。其中$\overline{w_i}$和$\overline{c_{i,j}}$ 分别表示平均计算开

销和平均通信开销。

$$\begin{cases} rank_{\mathrm{u}}(n_{\mathrm{exit}})=\overline{w_{\mathrm{exit}}} \\ rank_{\mathrm{u}}(n_i)=\overline{w_i}+\max\limits_{n_j\in succ(n_i)}\{c_{i,j}+rank_{\mathrm{u}}(n_j)\} \end{cases}. \tag{2.2}$$

然后，HEFT算法根据优先级依次将任务分配至具有最小的最早完成时间（Earliest-Finish-Time, EFT）的处理器上，从而获得较短的系统响应时间。

表 2.2 异构最早完成时间HEFT算法

HEFT算法
1.计算DAG图中每个任务的平均计算开销以及每条通信边的平均通信开销；
2.从出口任务开始向上遍历，计算出每个任务的向上排序值$rank_{\mathrm{u}}$；
3.While 任务队列中仍有未调度任务do
3.　　选择任务队列中的第一个就绪任务n_i；
4.　　for 处理器集合中的任一个处理器p_k do
5.　　计算n_i在每个处理器上的最早完成时间；
6.　　将任务n_i分配至具有最小的最早完成时间的处理器p_j 上；
7. end while

HEFT算法在长度比、加速度、最好结果出现的次数以及平均调度时间上都优于其他算法，是针对异构计算环境下的高效任务调度算法，以其高性能、低时间复杂度的优点成为现今任务调度研究中广为接受的方法，也是本研究中任务排序的基础方法。

2.其他静态任务调度算法

集群启发式算法[34–36]是将给定图中的任务映射至无数的集群上。每次迭代都通过合并集群来细化集群，集群合并之后便将集群映射至可用处理器上。如文献[34]提出低复杂度的集群计算算法Dominant Sequence Clustering （DSC）；文献[35] 提出在多处理器上并行计算调度的广泛应用方法Liner Clustering Method；文献[27]同时也提出一种高效的集群算法Mobility Directed；文献[36]提出一种无任务复制的集群启发式算法Clustering and Scheduling （CASS）。

任务复制启发式算法[31,37–39] 是指在调度一个任务图时冗余的复制一些任务，从而减少进程间的通信开销。在文[37]中，作者提出一种新的Critical Path Fast Duplication 算法，该算法在通信-计算比低和高的情况下都表现出高效性。文献[38] 提出一种基于复制的调度算法，能在与现有算法相同或更小的时间复杂度下取得更显著的性能改善。基于任务复制的算法根据任务复制所选择的策略不同而有所不同，通常用于无限大数量级的完全相同的处理器，且通常相比其他类型算法具有较高的复杂度。

随机搜索技术则是在目标位置基本分布均匀的情况下，通过随机选择来穿过问题空间的一种搜索方式，包括模拟退火算法、遗传算法等。其中，遗传算法

（GAs）[40–44]是任务调度问题几个关键技术中最为广泛接受和使用的技术。遗传算法具备良好的输出质量，但是相比于基于启发式的技术，遗传算法的调度时间往往过长。

2.2.2 非容错可靠性目标保障研究进展

基于分布式功能的DAG模型的可靠性研究是当前异构分布式系统的热点。正如ISO 26262 标准所述，随机硬件故障(即瞬时故障)在硬件元素的生命周期中不可预测地发生，但遵循某种概率分布[6]。在ISO 26262 中，可靠性与暴露率成反比，它指的是可能发生危险和有害事件的操作条件的相对预期频率[6]。随机硬件故障服从泊松分布在很多研究被广泛提及[16,45,46]。在[47,48]，Benoit等人证明了评估一个基于DAG 模型的分布式功能的可靠性是一个NP-难问题。在早期的开发阶段，可靠性增长成为支持整个开发计划决策的关键。[49] 提出了一种多目标、多阶段可靠性增长规划方法。同时，可靠性感知设计的技术和算法通常是为了在保障可靠性目标的同时最小化某些目标，例如响应时间、能耗、资源成本等。对于非容错情况下的可靠性目标保障方法，我们从国内及国外两方面对其研究进展进行总结。

1.国外研究进展

文献[50]建立了可靠性与能耗之间的关系模型。文献[51] 通过将响应时间和可靠性的双目标合并为一个目标功能进行联合优化，提出双目标动态级调度算法和双目标遗传算法两个算法，研究了可靠性与响应时间之间的双目标优化问题。文献[51] 中提出的方法可靠性较高，但响应时间比文献[45]中提出的方法要长。[52] 中的作者为解决独立任务的可靠性最大化问题，提出一种近似于帕累托曲线的算法，实现了最大限度地提高可靠性和最小化响应时间的双目标优化。在[53]中，Guo等人提出了面向同构系统的可靠性感知的能耗管理方法，在保障系统可靠性在一定水平的同时节省系统能耗。Salehi等人[54] 提出了一种针对同构多处理器的动态冗余和电压缩放的高效可靠性管理方法。在[55] 中，Salehi等人提出了一种N-mmodular redundancy(NMR)技术，用于在同构多核处理器上实现高可靠性、低能量开销的硬实时应用。然而，[45,51,52] 中提出的方法虽然能在保障可靠性目标的前提下最小化响应时间目标，但是这些方法都不是针对于功能的可靠性目标。而[53–55] 中提出的方法虽然针对系统中的功能在保障高可靠性的同时实现了系统的能耗优化，但这些方法都是针对同构计算系统而不是异构计算系统。

2.国内研究进展

国内方面，Zhao等人[56]利用任务提前完成产生的时间松弛，提出一种基于共享恢复的频率分配算法SHR-DAG，针对单处理器系统研究了在保持分布式功能可靠性的同时最小化能耗的问题。同时，作者对SHR-DAG进行了动态扩展，达到

提高运行时的能源效率和系统可靠性的目的。张剑贤等人[57]通过基于处理器可靠性约束的动态频率岛划分，降低了片上网络能耗。另外，作者借助量子粒子群算法优化了NoC映射,降低了系统的通信能耗。然而，这些方法都未以异构系统的分布式功能为研究对象。张艺文等人[58]针对可靠性感知和资源受限的周期任务的调度问题，提出最长执行时间优先算法(LETE)和最短执行时间优先算法(SETF)两种启发式算法。Zhang 等人[59–61] 对可靠性及能耗的调度问题作了深入的研究，利用动态电压调节技术，针对异构系统的分布式功能分别就能耗约束的可靠性最大化，能耗与可靠性联合优化，以及能耗与可靠性的双目标优化等问题提出了有效的RMEC 等调度策略。针对可靠性目标保障问题，Xiao 等人[62]创新性的提出了一种预分配机制，通过为未调度任务预分配能耗最小值将功能的能耗约束转化为每个任务的能耗约束，在改进了RMEC 的基础上提出了MREC方法，实现了能耗约束下的可靠性最大化。但这些方法都旨在保障能耗目标而非首先保障可靠性目标。对于保障可靠性目标的同时最小化系统能耗问题，Xie等人[63] 利用预分配机制提出了可靠性目标的能耗有效性调度算法energy-efficient scheduling with a reliability goal (ESRG)，实现了异构处理器高效的节能调度。在可靠性目标的资源优化方面，Xie 等人[16] 针对分布式并行应用提出一种预分配方法Minimizing Resource Consumption Cost with Reliability Goal （MRCRG），通过为每个未调度任务分配可靠性最大值，将应用的可靠性目标转换成每个任务的可靠性目标。该方法虽然能有效的保障功能可靠性目标，但是其针对未调度任务预分配可靠性最大值过于悲观，造成不必要的资源浪费。

2.2.3 容错可靠性目标保障研究进展

尽管采用高效的调度策略能够达到可靠性目标保障的目的，但是当分布式汽车功能的可靠性目标极高，采用基于任务复制的容错技术提高可靠性，从而保障新一代汽车电子系统分布式功能的高可靠性就十分必要[64]。当前主要的两种容错任务复制方式就是被动复制与主动复制。当主任务在处理器上调度失败时，备份副版任务才得以激活成为新的主版任务继续进行调度，这种方式则为被动复制。国内外相关学者采用被动复制对可靠性问题进行了丰富的研究，但是被动复制所有任务都只有两个副版本，无法容忍超过1个的错误，所以容错能力非常有限。主动复制则是指每个任务同时在多个处理器上进行复制，只要有至少一个备份副版任务执行成功，则整个任务即为成功调度。目前，主动复制以其较高的容错能力成为主要容错方法。对于容错情况技术的可靠性目标保障方法，我们从复制方式的不同对其国内外研究进展进行总结。

1.单个副本的被动复制

容错技术起初是采用简单的单备份的被动复制方式。典型的方法包括Qin等

人提出的独立任务成本驱动的eFRCD(Efficient Fault-tolerant Reliability Cost Driven)算法[65] 以及考虑任务间约束关系，同时采用重叠机制以最大程度减少系统执行时间的eFRD(Efficient Fault-tolerant Reliability Driven) 算法[66]。此外，文献[67] 针对DAG 中的非独立任务和独立任务提出了MRC-ECT （Minimum Replication Cost with Early Completion Time）和MCT-LRC （Minimum Completion Time with Less Replication Cost）两个算法，研究了完成时间和复制开销两个目标相互的约束与优化问题。文献[68] 提出一种容错动态重调度算法FTDR以实现制造成本和资源消耗的优化。被动复制具有一定的有效性，能达到的性能比较高。但是这些算法首先只假定在同一时刻只有一次失效发生，过于理想化，无法容忍潜在的多重失效。另外，上述算法都只是利用容错来完成应用的调度，并没有将应用的可靠性作为优化目标。

2.固定副版本数量的主动复制

相对于被动复制，主动复制以其可以直接屏蔽执行失败的任务，并且故障恢复时间接近于零的优势成为后来容错技术的主要方式。针对于每个任务，固定的复制 ε 个副版本，从而解决了复制一次无法容忍多个失效的缺点。鉴于这种复制方式的高可靠性，较多文献都对这一复制方式进行了研究。Girault 等人[45]提出了以故障率为约束的条件下最小化应用程序调度长度的双目标启发式调度算法(BSH)，该方法生成一条帕累托曲线，可以根据需要选择适合需求的解。Benoit等人[69] 则提出了一种容错调度的FTSA算法，该算法首先把每个任务都复制出共 ε +1 个版本，每次选择优先级最高的任务调度。由于复制次数较多，系统能容忍 ε 个错误。该算法是高可靠性的调度方式，但是每次只调度一个任务就会影响效率。为改进这一缺点，作者在文献[70] 又提出一种CAFT （Contention Awareness and Fault-Tolerant）算法，通过同时选择一组任务进行调度，在同时将这组任务的所有副版本任务进行调度，所以此方法就在一定程度上使得负载更为均衡。上述方法都对每个任务复制 ε +1 次，虽然能够复制的数量使得能够容忍系统中可能存在的 ε 个故障，在一定程度上提高系统的可靠性，但是这种盲目复制容易造成任务冗余过高，使得执行过程中付出了更多的计算资源而造成性能下降。

3.不定量副版本的主动复制

针对上述复制策略的高冗余性，近年来的研究开始探索针对每种具有任务不同的数量备份，以满足应用程序的可靠性要求[71,72]。文献[71] 旨在提出更为准确的复制数量策略，认为合适的副版本数量才能在达到可靠性目标的同时提高性能。作者在文中提出一种最大可靠性算法MaxRe 算法，给定系统的可靠性目标，利用提出的策略将系统的可靠性目标分配至每个任务上，达到不同任务不同可靠性目标的目的。然后根据当前任务的可靠性目标，依次开始复制并将备份副版任

务调度至具有最大可靠性的处理器上，直至可靠性满足该任务的可靠性目标。这种方式针对任务不同计算出与之匹配的可靠性目标，同时动态的为此任务选择副版本数量，大大节省了副版本的冗余，改进了盲目复制的悲观性。但是这种可靠性目标保障方法认为功能的可靠性为每个任务的可靠性之积，因而每个任务的可靠性目标为功能可靠性的开方均值

$$r = \sqrt[n]{R}, \tag{2.3}$$

其中R为功能可靠性需求，n为任务个数。然而，由于DAG中的任务存在优先约束关系，对于当前调度的任务，其已完成调度的前驱任务对当前任务的约束关系会对当前任务的调度产生影响，所以这种功能的可靠性分配至任务的模型需进一步修订。基于此，作者在文献[72] 又对这一可靠性目标模型进行调整，提出可靠性目标下的冗余最小化策略least Resource to meet Reliability requirement （RR）算法以及对RR算法进行扩展将响应时间约束加入，可靠性及期限约束下的冗余最小化策略least Resource to meet Deadline and Reliability requirement （DRR）算法。对于可靠性目标的保障，两方法采用了相同的可靠性模型。考虑到前驱任务对当前任务的影响，当调度第j 个任务时，所有$[1, j-1]$ 个前驱任务的可靠性为则已知参数，那么当前任务可靠性目标则应为

$$r = \sqrt[n-j+1]{\frac{R}{\prod_{x=1}^{j-1} r_x}}. \tag{2.4}$$

这一可靠性模型考虑了前驱任务对当前任务的影响，能高效的减少资源冗余。但是DAG 中的任务不仅仅有前驱任务优先约束，当前任务与后继任务之间也有优先约束关系。针对这一针对上述问题，文献[46]首次提出一种容错技术下的任务预分配机制Heuristic Replication for Redundancy Minimization（HRRM）算法，在计算当前任务的可靠性目标时，不仅考虑了前驱任务的已获得的可靠性值，还为每个未调度后继任务预分配了可靠性上界值，从而基于计算可靠性进行了精确的DAG 建模，以精确量化需要复制的任务个数，达到资源冗余最小化。然而，HRRM采用贪婪的方法来分配高优先级的任务，从而可能由于低优先级任务的不可靠性而影响整个系统的响应性。

2.2.4 研究进展小结

启发式算法是静态任务调度算法中应用最广泛的算法类型，其中HEFT算法中的向上排序方法是异构系统调度研究中的常用任务排序方法。同时，本节对基于DAG 模型的可靠性目标保障调度问题现状及发展动态进行了详细的阐述与分析，并归纳了存在的问题。对于非容错情况，可靠性模型得到广泛的研究，并形成可靠性泊松分布的瞬时故障模型。相关学者对可靠性和响应时间的双准则联合

优化问题进行过深入研究，但这些方法都不是针对于功能的可靠性目标调度。目前针对非容错技术下功能或应用的可靠性目标保障问题所提出方法的还具有一定的悲观性。对于容错技术，作为保障可靠性的重要手段，有着丰富且比较成熟的研究，该机制消耗计算资源但是可靠性较高。且现有方法已突破过往盲目复制的缺点，提出针对不同任务的可靠性目标采用不同的副版本数量，大大减少了资源冗余。但是现有方法的贪婪策略所显示出的不公平性仍是研究的重点。

2.3 本章小结

由于汽车嵌入式系统的ECU的密集分布，计算数据的庞大精细，需要精确的并行与分布式模型描述ACPS 的高集成度、高并行性。本章分析了ACPS基于中央网关的ECU架构，并将汽车内任务之间具有明显约束关系的功能抽象成DAG模型。通过对四类静态调度算法详细分析，给出每类算法的经典算法特点及最新研究进展。本研究旨在研究非容错和容错情况下的基于DAG模型的可靠性目标保障调度，相关研究主要运用表调度算法启发式算法。因而，本章最后着重阐述了两者的相关研究及其进展，同时分析了现有研究所存在的问题。

第3章　可靠性目标下的资源最小化非容错调度

对于安全关键的分布式汽车功能，可靠性是重要的功能安全需求。由于容错技术以保障高可靠性为目标，会消耗较高的计算资源。那么在不采用容错技术情况下，资源就成为汽车嵌入式系统的重要优化性能指标。本章面向ACPS 功能安全可靠性需求，研究不采取基于任务复制的非容错技术下的可靠性目标保障方法，在保障分布式汽车功能可靠性目标前提下，最小化系统计算资源消耗成本的调度问题。

3.1　相关模型

3.1.1 异构计算任务的可靠性模型

通常对于一个功能故障有两种主要的类型，瞬间故障(即随机硬件故障)和永久性故障。由于功能安全标准ISO 26262 将随机硬件失效和可靠性结合起来，因而本研究我们只考虑瞬间故障。ISO 26262 在第3 部分附件B表B.2 中给出了随机硬件故障暴露率的定义，指出是指可能发生危险事件并造成危险和伤害的操作条件的相对期望频率。如表3.1 所示不同的发生概率，分为E1、E2、E3 和E4四个层次，分别表示非常低概率、低概率、中概率和高概率。ISO 26262 标准指出可靠性目标与暴露率成反比，即暴露率越高，可靠性目标越低。另外，ISO 26262 标准指出功能的可靠性是一个随机发生的概率值，所以无法保障功能完完全全的可靠，只要能够采取有效的保障措施来保障功能所指定要达到的暴露率（即可靠性目标），那么就认为此功能可靠。

表 3.1 ISO 26262中定义的暴露率及对应的可靠性目标

暴露层次	暴露率	可靠性目标
E1 非常低概率	未规定	至少高于0.99
E2 低概率	< 1%	0.99
E3 中概率	[1%,10%]	> 0.9
E4 高概率	> 10%	<= 0.9

任务的可靠性是指系统中任务无故障运行的概率。ISO 26262 标准指出ECU发生随机硬件失效的概率服从特定的分布[6]。通常来说，基于DAG 模型的分布式汽车功能中的任务发生瞬间故障的概率服从泊松分布[73], 即单位时间t 内的可靠性为$R(t) = e^{-\lambda t}$。令λ_k 代表ECUu_k单位时间内的恒定故障率，则任务n_i 在其执行时

间内在ECUu_k 上的可靠性表示为:

$$R(n_i, u_k) = e^{-\lambda_k \times w_{i,k}}. \tag{3.1}$$

安全关键系统的开发生命周期通常包括分析、设计、实现和测试阶段。本研究我们旨在设计阶段在满足分布式汽车功能可靠性目标的条件下最小化资源消耗成本。类似于上文中所述消息的WCTT 和任务的WCRT，我们认为故障率在分析阶段已经确定。另外，由于CAN网络本身具有较高的容错能力，因而本研究仅考虑ECU 故障，不考虑通信故障。

任务n_i通过遍历所有ECU上所获得的可靠性值，可以得出其最小和最大可靠性分别为:

$$R_{\min}(n_i) = \min_{u_k \in U} R(n_i, u_k), \tag{3.2}$$

和

$$R_{\max}(n_i) = \max_{u_k \in U} R(n_i, u_k). \tag{3.3}$$

那么，任务间具有优先级约束的功能G 的可靠性是所有任务可靠性的积，即，

$$R(G) = \prod_{n_i \in N} (R(n_i, u_{proc(n_i)})), \tag{3.4}$$

其中$u_{proc(n_i)}$表示任务n_i所分配的ECU。那么，功能G 的最小和最大可靠性值则分别为所有任务的可靠性最小值和最大值之积:

$$R_{\min}(G) = \prod_{n_i \in N} R_{\min}(n_i), \tag{3.5}$$

和

$$R_{\max}(G) = \prod_{n_i \in N} R_{\max}(n_i). \tag{3.6}$$

如前所述，如果可靠性目标$R_{goal}(G)$能够满足，那么该功能则认为是可靠的。需要强调的是$R_{goal}(G)$ 必须大于或等于$R_{\min}(G)$，否则可靠性目标值$R_{goal}(G)$ 总是能得到满足；同样，$R_{goal}(G)$ 必须小于或等于$R_{\max}(G)$，否则可靠性目标值$R_{\max}(G)$ 总是不能得到满足。因此，$R_{goal}(G)$ 必须属于$R_{\min}(G)$ 和$R_{\max}(G)$ 之间，即，

$$R_{\min}(G) \leq R_{goal}(G) \leq R_{\max}(G). \tag{3.7}$$

3.1.2 资源消耗成本模型

作为成本关键的系统，资源消耗是汽车电子系统重要的优化指标。从计算资源消耗成本和通信资源消耗成本两方面对系统的资源消耗成本进行建模。

定义单位时间内任务执行的计算资源成本率 γ，表明ECU 的计算资源成本以 γ 的速率逐步增长，且ECU 的计算资源随着计算资源速率的增长而不断增长，以

访问存储在存储器中的数据。也就是说，对于执行时间为w的任务，成本为$\gamma \times w$。

定义消息转换(传输)的固定成本。然而，当消息在分布式嵌入式系统上传输时，不同大小的消息会消耗不同的资源成本。这是因为每条消息在发送之前都应该打包，在收到之后解析，而打包和解析时间则取决于消息的大小。由于本研究采用同构通信链路，所以单位时间消息传输的通信成本γ_{comm}都是相同的。

当任务在ECU上完成执行后，该任务将消息传输至其所有的后继任务，这些后继任务可能位于不同的ECU 上。因此，令$cost(n_i, u_k)$代表任务n_i在ECUu_k上执行产生的资源消耗成本，表示为

$$cost(n_i, u_k) = w_{i,k} \times \gamma_k + \sum_{n_x \in pred(n_i)} c'_{x,i} \times \gamma_{\text{comm}}, \tag{3.8}$$

其中，$c'_{x,i}$指任务n_x和n_i间的实际通信时间。如果n_x和n_i分配至同一ECU 上，则$c'_{x,i} = 0$；反之若两者分配至不同ECU 上，则$c'_{x,i} = c_{x,i}$。那么，$cost(n_i, u_k)$包括任务n_i在ECU u_k 上执行的ECU 资源消耗成本以及消息在任务n_i 和其分配至不同ECU上的前驱任务间传输的通信资源开销。当前的任务仅考虑其前继任务，其后继任者将在计算通信资源消耗成本时考虑当前的任务。

最后，功能G的总资源消耗成本是所有任务资源消耗成本的和，即：

$$cost(G) = \sum_{n_i \in N} cost(n_i, u_k). \tag{3.9}$$

3.2 问题陈述

给定一个已知可靠性目标$R_{\text{goal}}(G)$的分布式功能G，该功能将在异构多处理器集U 上被执行。本研究旨在提出一种更有效的可靠性目标保障方法，以获得较低的资源消耗成本。具体来说，是为每个任务分配一个可用的ECU，同时降低功能G 的资源消耗成本，并满足其可靠性目标$R_{\text{goal}}(G)$。即将所有的任务分配给ECU以最小化功能的资源消耗成本G：

$$cost(G) = \sum_{n_i \in N} cost(n_i, u_{proc(i)}), \tag{3.10}$$

同时采取有效的可靠性保障方法来满足：

$$R(G) = \prod_{n_i \in N} R(n_i, u_{proc(i)}) \geq R_{\text{goal}}(G), \tag{3.11}$$

对于$\forall i : 1 \leq i \leq |N|, u_{proc(i)} \in U$.

3.3 高效的可靠性目标保障算法

3.3.1 任务排序

基于DAG模型的调度问题实质上是为任务找至ECU上的映射，所以首先需确定任务的分配顺序，即任务优先级排序。类似于文[46,72,74,75]，本研究采用HEFT算法中的向上排序值($rank_{\mathrm{u}}$) 作为任务优先级排序方法，表示为：

$$rank_{\mathrm{u}}(n_i) = \overline{w_i} + \max_{n_j \in succ(n_i)} \{\overline{c_{i,j}} + rank_{\mathrm{u}}(n_j)\}. \tag{3.12}$$

其中$\overline{w_i}$表示任务n_i的平均WCET且$\overline{w_i} = \left(\sum_{k=1}^{|U|} w_{i,k}\right)/|U|$。所有任务根据($rank_{\mathrm{u}}$)值的降序来排列形成调度顺序，表2.1展示了图2.2 中功能G 所有任务的向上排序值，因而任务调度顺序为$\{n_1,n_3,n_4,n_2,n_5,n_6,n_9,n_7,n_8,n_{10}\}$。只有当任务$n_i$ 的所有具有优先级约束的前驱任务执行调度完成后，任务n_i 才进入调度就绪状态。如果两个任务n_i 和n_j 满足$rank_i > rank_j$，但是n_i 和n_j 之间没有优先级约束，那么n_i 可能不会有比n_j 更高的优先级。

3.3.2 现有MRCRG方法

MRCRG方法[16]旨在研究非容错情况下获得最小的资源消耗成本同时满足并行应用的可靠性目标。在采用向上排序值对任务进行排序后，计算任务的可靠性目标和分配任务至处理器是两个重要的步骤。

为了让每个任务都满足应用的可靠性目标，作者假设当前调度的任务是$n_{seq(j)}$，$n_{seq(j)}$表示第j 个被调度的任务。那么$\{n_{seq(1)},n_{seq(2)},...,n_{seq(j-1)}\}$ 代表已经完成调度的任务集，$\{n_{seq(j+1)},n_{seq(j+2)},...,n_{seq(|N|)}\}$ 表示未被调度的任务集。然后，作者为每个未调度的任务$n_{seq(y)}$ 都预分配其最大可靠性值$R_{\max}(n_{seq(y)})$。因此，当调度任务$n_{seq(j)}$ 时，当且仅当$R_{seq(j)}(G) \geqslant R_{\mathrm{goal}}(G)$，即功能$G$ 的可靠性应该满足所有任务的可靠性乘积大于或等于功能的可靠性目标$R_{\mathrm{goal}}(G)$：

$$R_{seq(j)}(G) = \prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times R(n_{seq(j)}, u_{proc(seq(j))}) \times \prod_{y=j+1}^{|N|} R_{\max}(n_{seq(y)}) \geqslant R_{\mathrm{goal}}(G), \tag{3.13}$$

那么当前任务$n_{seq(j)}$的可靠性应满足：

$$R(n_{seq(j)}, u_{proc(seq(j))}) \geqslant R_{\mathrm{goal}}(G) \Big/ \left(\prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times \prod_{y=j+1}^{|N|} R_{\max}(n_{seq(y)}) \right) \tag{3.14}$$

令$R_{\text{goal}}(n_{seq(j)})$表示任务$n_{seq(j)}$应满足的可靠性目标，那么其值为：

$$R_{\text{goal}}(n_{seq(j)}) = R_{\text{goal}}(G) \Big/ \left(\prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times \prod_{y=j+1}^{|N|} R_{\max}(n_{seq(y)}) \right) \tag{3.15}$$

通过这种对未调度任务预分配可靠性最大值的机制，可以动态迭代的得到当前任务的可靠性目标。这一可靠性目标转换机制考虑了DAG 模型中任务之间的优先级约束关系。最后，MRCRG 筛选掉不满足任务$n_{seq(j)}$ 可靠性目标$R_{\text{goal}}(n_{seq(j)})$的ECU，并将该任务调度至具有最小资源消耗成本的ECU 上。

例1.为了解释MRCRG方法的步骤与过程，采用图2.2 中的分布式功能例子进行详细分析。假定已知的3 个ECU$\{u_1, u_2, u_3\}$ 的故障率和计算资源消耗成本如表3.2 所示，单位时间内消息传输的通信成本为$\gamma_{\text{comm}} = 1$。为了简单起见，在示例中忽略所有参数的单位。

表 3.2 ECU$\{u_1,u_2,u_3\}$的故障率和成本率

参数	ECU u_1	ECU u_2	ECU u_3
故障率λ_k	0.0005	0.0002	0.0009
计算资源消耗成本γ_k	5	9	2

根据公式(3.1)、(3.2)和(3.3)，计算出任务n_i 在每个ECUu_k 上运行获得的可靠性值，遍历所有ECU 得出其中的最小可靠性和最大可靠性值。然后根据公式(3.4)、(3.5) 和(3.6)，可以计算出功能G 的最小可靠性和最大可靠性值。假定功能G 的可靠性目标设置为$R_{\text{goal}}(G)$=0.95。表3.3 所示该分布式功能示例通过MRCRG 算法生成的任务调度结果，其中每行代表一个任务的分配结果。粗体文本表示该任务所分配的具有最小资源消耗成本的ECU，而“-”表示该ECU 无法保障此任务的可靠性目标。根据提出的可靠性目标计算方法3.15 计算出任务n_1可靠性目标$n_1 = 0.9919$，筛选后3个ECU 都能满足该可靠性目标。利用式(3.8) 计算出n_1 在3 个ECU 上资源消耗成本，并将任务调度至具有最小资源消耗成本的ECU u_3 上。根据式(3.4)和式(3.9)，最终的可靠性是$R(G)$=0.9502 > $R_{\text{goal}}(G)$=0.95，最终资源消耗成本为$cost(G)$=910。

结果表明，MRCRG方法在满足给定的可靠性目标的同时，在资源消耗成本最小化方面取得了很好的效果。然而，作者为每个未调度任务预分配其最大可靠性值的策略过于悲观，这种方法对于高优先级任务过于贪婪，造成低优先级任务的可靠性目标较高，因而低优先级任务可选择的ECU 少，这种不平衡性就会造成很大的资源浪费。

除了MRCRG，还有另外一种新的启发式方法叫做HRRM，也是后文我们研究的重点。该方法采用基于任务复制的容错技术，在满足并行应用的可靠性目标的同时，获得足够的最小副本冗余。与MRCRG相似的是，HRRM 在将功能可靠

表 3.3 分布式功能示例通过MRCRG算法生成的任务调度结果

n_i	$R_{\text{goal}}(n_i)$	$cost(n_i,u_1)$	$cost(n_i,u_2)$	$cost(n_i,u_3)$	$R(n_i)$
n_1	0.9919	70	144	**18**	0.9919
n_3	0.9830	55	117	**38**	0.9830
n_4	0.9925	**74**	81	-	0.9935
n_2	0.9952	-	**189**	-	0.9962
n_5	0.9964	-	**117**	-	0.9974
n_6	0.9958	-	**158**	-	0.9968
n_9	0.9966	-	**131**	-	0.9976
n_7	0.9960	**35**	135	-	0.9965
n_8	0.9873	**59**	133	-	0.9975
n_{10}	0.9884	-	**91**	-	0.9986
$cost(G)$=910, $R(G)$=0.9502 > $R_{\text{goal}}(G)$ =0.95					

性转换成任务可靠性时，也采用了对未调度任务进行预分配可靠性值的方法。不同于MRCRG 将最大可靠性值预分配给未调度任务，HRRM 假定未调度任务分配了一个可靠性上界值，该值计算方法为：

$$R_{\text{up\_goal}}(n_{seq(y)}) = \sqrt[|N|]{R_{\text{goal}}(G)}, \tag{3.16}$$

从而任务$n_{seq(j)}$的可靠性目标转换为：

$$R_{\text{goal}}(n_{seq(j)}) = R_{\text{goal}}(G) \Big/ \left(\prod_{x=1}^{j-1} R(n_{seq(x)}) \times \prod_{y=j+1}^{|N|} R_{\text{up\_goal}}(n_{seq(y)}) \right). \tag{3.17}$$

例2.虽然HRRM是针对于容错机制下的可靠性目标任务调度，但其这种预分配机制可以运用在所研究的非容错情况下。同样，我们用上述分布式功能示例解释HRRM预分配机制运用至本研究的过程与结果，将功能可靠性目标值同样设为0.95。根据式(3.16)，可靠性上界值为$R_{\text{up_goal}}(n_i) = \sqrt[10]{0.95} = 0.9949$。当调度任务$n_1$时，根据式(3.17)，任务$n_1$的可靠性目标为$R_{\text{goal}}(n_1) = \sqrt[10]{0.95} = 0.9949$，只有ECU u_2 能满足其可靠性目标，因而n_1 被调度至ECU u_2 上。然而，当调度任务n_2时，其可靠性目标计算为$R_{\text{goal}}(n_2) = 0.9980 > R_{\max}(n_2) = 0.9974$，没有ECU 能够达到$n_2$的可靠性目标，因而无法为其分配处理器。根据实验结果，当把HRRM 的预分配机制应用至非容错情况时，对未调度任务所预分配的可靠性上界值过低，造成任务的可靠性目标不能总是达得到。

3.3.3 几何平均值

综上所述，由于MRCRG预分配最大可靠性值给未调度任务过于贪婪，

且HRRM预分配可靠性上界值给未调度任务过低不适用于非容错情况，我们为未调度任务寻求更有效的可靠性预分配值，以解决现有可靠性目标保障方法的不平衡性。

在数学中，均值表示一组数字的中心集中趋势，它可以更均衡地分配数值，一般分为三中类型平均值：几何平均值、算术平均值以及调和平均值。

几何平均表示对$|N|$个变量值的连乘积开$|N|$次方根。对于一组数$t_1, t_2, ..., t_{|N|}$，它们的几何平均值为

$$GM = \sqrt[|N|]{t_1 \times t_2 \times ... \times t_{|N|}}. \tag{3.18}$$

算术平均值表示$|N|$个变量值的和除以数量$|N|$。对于一组数$t_1, t_2, ..., t_{|N|}$，它们的算术平均值为

$$M = \frac{t_1 + t_2 + ... + t_{|N|}}{|N|}. \tag{3.19}$$

调和平均值表示$|N|$除以每个变量值的倒数之和。对于一组数$t_1, t_2, ..., t_{|N|}$，它们的调和平均值为

$$HM = \frac{|N|}{\frac{1}{t_1} + \frac{1}{t_2} + ... + \frac{1}{t_{|N|}}}. \tag{3.20}$$

这三种方法都可以表示一组数字的中心集中趋势。我们使用几何平均值的原因如下：

（1）功能的可靠性是所有任务的可靠性值的乘积。几何平均值同样是利用数值的乘积并将结果趋于所有值的中心，而算术平均值和调和平均值只是利用数值的和。

（2）算术平均值和调和平均值都容易受到两端的极值的影响。上值越大，平均向上偏离中心趋势的偏差越大。反之，下值越小，平均向下偏离中心趋势的偏差越大。而几何平均值的优点是它不太容易受到极端值的影响。

如果我们针对未调度任务可以定义一个最大值和上界值之间的中心集中值，我们就可以减少现有方法的不平衡性。因此，我们为所要提出的调度方法引入几何平均值的计算策略。

首先，针对分布式汽车功能和任务我们分别定义两种类型的几何平均值。

令$R_{\mathrm{gmf}}(G)$表示功能G的几何平均值，我们将其设为：

$$R_{\mathrm{gmf}}(G) = \sqrt{\sqrt[|N|]{R_{\max}(G)} \times \sqrt[|N|]{R_{\min}(G)}}, \tag{3.21}$$

令$R_{\mathrm{gmt}}(n_{seq(y)})$表示未调度任务$n_{seq(y)}$ 的几何平均值，我们将其定义为任务$n_{seq(y)}$的最大可靠性和最小可靠性的几何平均值：

$$R_{\mathrm{gmt}}(n_{seq(y)}) = \sqrt{R_{\max}(n_{seq(y)}) \times R_{\min}(n_{seq(y)})}, \tag{3.22}$$

根据公式(3.16)，每个未调度任务的可靠性上界为：

$$R_{\mathrm{gmf}}(G) = \sqrt{\sqrt[|N|]{R_{\max}(G)} \times \sqrt[|N|]{R_{\min}(G)}}, \tag{3.23}$$

最后，令$R_{\mathrm{gmpr}}(n_{seq(y)})$表示未调度任务$n_{seq(y)}$基于几何平均值的可靠性预分配值，我们将其定义为：

$$\begin{aligned} R_{\mathrm{gmpr}}(n_{seq(y)}) &= \frac{R_{\mathrm{gmt}}(n_{seq(y)})}{R_{\mathrm{gmf}}(G)} \times R_{\mathrm{up\_goal}}(n_{seq(y)}) \\ &= \frac{\sqrt{R_{\max}(n_{seq(y)}) \times R_{\min}(n_{seq(y)})}}{\sqrt{\sqrt[|N|]{R_{\max}(G)} \times \sqrt[|N|]{R_{\min}(G)}}} \times \sqrt[|N|]{R_{\mathrm{goal}}(G)}. \end{aligned} \tag{3.24}$$

3.3.4 可靠性目标保障

根据以上定义，同样假设当前调度的任务是$n_{seq(j)}$，$n_{seq(j)}$表示第j个被调度的任务。那么$\{n_{seq(1)},n_{seq(2)},...,n_{seq(j-1)}\}$代表已经完成调度的任务集，$\{n_{seq(j+1)},n_{seq(j+2)},...,n_{seq(|N|)}\}$表示未被调度的任务集。因此，当调度任务$n_{seq(j)}$时，功能$G$的可靠性$R_{seq(j)}(G)$则为：

$$R_{seq(j)}(G) = \prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times R(n_{seq(j)}, u_{proc(seq(j))}) \times \prod_{y=j+1}^{|N|} R_{\mathrm{gmpr}}(n_{seq(y)}). \tag{3.25}$$

对于任何任务$n_{seq(j)}$，当且仅当$R_{seq(j)}(G) \geq R_{\mathrm{goal}}(G)$时，实际的可靠性$R(G)$才会大于或等于可靠性目标$R_{\mathrm{goal}}(G)$。此方法我们将其命名为定理1并在证明中予以证明。

定理1：分布式汽车的任一功能G的任一任务$n_{seq(j)}$总是能找到一个ECU的分配以满足：

$$R_{seq(j)}(G) = \prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times R(n_{seq(j)}, u_{proc(seq(j))}) \times \prod_{y=j+1}^{|N|} R_{\mathrm{gmpr}}(n_{seq(y)}) \geq R_{\mathrm{goal}}(G). \tag{3.26}$$

证明：为证明上述定理的正确性，我们只需证明所有任务预先分配的可靠性值的乘积大于或等于功能的可靠性目标即可。

首先，根据式(3.24)，所有任务基于几何平均值的预分配可靠性值的乘积表示为：

$$R_{\mathrm{gmpr}}(G) = \prod_{n_i \in N} R_{\mathrm{gmpr}}(n_{seq(i)}) = \prod_{n_i \in N} \left(\frac{R_{\mathrm{gmt}}(n_{seq(i)})}{R_{\mathrm{gmf}}(G)} \times R_{\mathrm{up\_goal}}(n_{seq(i)}) \right), \tag{3.27}$$

然后，根据式(3.21)、(3.22)和(3.16)，代入得到：

$$\begin{aligned} R_{\text{gmpr}}(G) &= \prod_{n_i \in N} \left(\frac{\sqrt{R_{\max}(n_{seq(i)}) \times R_{\min}(n_{seq(i)})}}{\sqrt{\sqrt[|N|]{R_{\max}(G)} \times \sqrt[|N|]{R_{\min}(G)}}} \times \sqrt[|N|]{R_{\text{goal}}(G)} \right) \\ &= \frac{\sqrt{R_{\max}(G) \times R_{\min}(G)}}{\sqrt{R_{\max}(G) \times R_{\min}(G)}} \times \prod_{n_i \in N} \left(\sqrt[|N|]{R_{\text{goal}}(G)} \right) \\ &= R_{\text{goal}}(G). \end{aligned} \tag{3.28}$$

因此，假定所有任务基于几何平均值的可靠性预分配值之积$R_{\text{gmpr}}(G)$等于$R_{\text{goal}}(G)$，任一任务$n_{seq(j)}$就能找到一个ECU来满足功能可靠性$R_{\text{goal}}(G)$。

在定理1的基础上，我们可以得出任务$n_{seq(j)}$需满足：

$$R(n_{seq(j)}, u_k) \geq R_{\text{goal}}(G) \Big/ \left(\prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times \prod_{y=j+1}^{|N|} R_{\text{gmpr}}(n_{seq(y)}) \right), \tag{3.29}$$

令任务$n_{seq(y)}$的可靠性目标$R_{\text{goal}}(n_{seq(j)})$为：

$$R_{\text{goal}}(n_{seq(j)}) = R_{\text{goal}}(G) \Big/ \left(\prod_{x=1}^{j-1} R(n_{seq(x)}, u_{proc(seq(x))}) \times \prod_{y=j+1}^{|N|} R_{\text{gmpr}}(n_{seq(y)}) \right), \tag{3.30}$$

那么，基于此预分配机制将功能G的可靠性目标转换成每个任务的可靠性目标，任务$n_{seq(j)}$只需要满足：

$$R(n_{seq(j)}, u_k) \geq R_{\text{goal}}(n_{seq(j)}). \tag{3.31}$$

因此，在接下来的调度过程中可以直接考虑当前任务$n_{seq(y)}$的可靠性目标$R_{\text{goal}}(n_{seq(j)})$而不是功能$G$的可靠性目标。

3.3.5 RGAGM算法

在上述分析的启发下，我们提出了一种更有效的可靠性目标保障方法– 基于几何平均值的可靠性目标保证方法Reliability Goal Assurance Method using Geometric Mean(RGAGM)，并在算法3.3.1中给出了具体算法步骤。

RGAGM的主要思想是为每个未调度的任务确定一个基于几何平均值的可靠性预分配值，以将功能的可靠性目标转换为每个任务的可靠性目标，然后将任务分配给满足其可靠性目标且具有最小资源消耗成本的ECU。RGAGM的主要步骤解释如下：

1）第1行，根据式(3.12)按向上排序值$rank_{\text{u}}$的降序对任务列表*task_list*中的所有任务进行优先级排序。

2）第2-6行，根据式(3.2)和(3.3)计算每个任务n_i在不同ECU上的可靠性，遍历所有ECU得到任务可靠性最大值和最小值。根据式(3.5)和(3.6)得出功能G的可靠性最大值和最小值。根据(3.21)–(3.23)，分别确定所定义的针对任务和功能的几何

平均值。

3）第9-14行，根据式(3.30)计算基于几何平均值的可靠性值预分配值，得出当前任务的可靠性目标，并筛掉不属于最小和最大可靠性范围内的任务的可靠性目标，从而最终确定当前任务的可靠性目标。

4）第18-25行，RGAGM筛选掉不满足任务可靠性目标的ECU，并将任务调度至具有最小资源消耗成本的ECU上。

5）第28-30行，计算出功能的实际可靠性和最终的资源消耗成本。

RGAGM的时间复杂度分析如下。调度功能的所有$|N|$个任务需要遍历所有的任务，这一步骤的时间复杂度为$O(|N|)$；计算每个任务在不同ECU上的资源消耗成本并得出最小资源消耗成本需要每个任务遍历所有的ECU，这一步骤的时间复杂度为$O(|N| \times |U|)$。因此，RGAGM的时间复杂度为$O\left(|N|^2 \times |U|\right)$。

算法 3.3.1 RGAGM方法

Input: G,$R_{\text{goal}}(G)$,$U = \{u_1, u_2, ..., u_{|U|}\}$,$\{\gamma_1, \gamma_2, ..., \gamma_{|U|}\}$,$\gamma_{\text{comm}}$
Output: $R(G)$, $cost(G)$

1: 根据式(3.12)对功能G中的任务按照$rank_u$的降序排列，并放入到任务优先级队列$task_list$中;
2: **for** ($\forall i, n_i \in N$) **do**
3: 使用式(3.2)和(3.3)分别计算任务n_i的最小可靠性$R_{\min}(n_i)$和最大可靠性$R_{\max}(n_i)$;
4: **end for**
5: 使用式(3.5)和(3.6)分别计算功能G的最小可靠性$R_{\min}(G)$和最大可靠性$R_{\max}(G)$;
6: 使用式(3.21)、(3.22)和(3.23)分别计算针对功能和任务的几何平均值$R_{\text{gmf}}(n_{seq(i)})$、$R_{\text{gmt}}(n_{seq(i)})$和$R_{\text{up_goal}}(n_{seq(i)})$、;
7: **while** (there are tasks in *dlist*) **do**
8: $n_i = dlist.out()$;
9: 使用式(3.30)计算$R_{\text{goal}}(n_i)$;
10: **if** ($R_{\text{goal}}(n_i) < R_{\min}(n_i)$) **then**
11: $R_{\text{goal}}(n_i) \leftarrow R_{\min}(n_i)$;
12: **else if** ($R_{\text{goal}}(n_i) > R_{\max}(n_i)$) **then**
13: $R_{\text{goal}}(n_i) \leftarrow R_{\max}(n_i)$;
14: **end if**
15: **for** each ECU ($u_k \in U$) **do**
16: 使用式(3.1)计算$R(n_i, u_k)$;
17: 使用式(3.8)计算$cost(n_i, u_k)$;
18: **if** ($R(n_i, u_k) < R_{\text{goal}}(n_i)$) **then**
19: continue;
20: **end if**
21: **if** ($cost(n_i, u_k) < cost(n_i, u_{proc(i)})$ **then**
22: $proc(i) \leftarrow k$;
23: $R(n_i, u_{proc(i)}) \leftarrow R(n_i, u_k)$;
24: $cost(n_i, u_{proc(i)}) \leftarrow cost(n_i, u_k)$;
25: **end if**
26: **end for**
27: **end while**
28: 使用式(3.4)计算功能G的实际可靠性$R(G)$;
29: 使用式(3.9)计算功能G的最终资源消耗成本$cost(G)$;
30: **return** $R(G)$和$cost(G)$.

3.3.6 RGAGM算法示例

例3.我们用上文同样的分布式功能示例来解释RGAGM的过程与结果。同样将功能可靠性目标设置为$R_{\text{goal}}(G) = 0.95$。不同于上述MRCRG算法，RGAGM 根据式(3.24)定义未调度任务的基于几何平均值的可靠性预分配值，然后得到当前任务的可靠性目标值。同样，任务可靠性目标调整必须在其最大可靠性和最小可靠性之间。将不能达到任务可靠性目标的ECU排除，并将任务调度至具有最小资源消耗成本的ECU上。表3.4 是采用RGAGM 算法的任务分配结果。表中粗体文本中的值表示该任务所选择的具有最小资源消耗成本的ECU。“-”表示的值表示将任务分配给该ECU无法达到任务的可靠性目标。通过RGAGM 算法，功能最终的可靠性为$R(G)$=0.9579 > $R_{\text{goal}}(G)$ =0.95，总资源消耗成本为$cost(G) = 824$。

表 3.4 分布式功能示例通过RGAGM算法生成的任务调度结果

n_i	$R_{\text{goal}}(n_i)$	$cost(n_i, u_1)$	$cost(n_i, u_2)$	$cost(n_i, u_3)$	$R(n_i)$
n_1	0.9919	70	144	**18**	0.9919
n_3	0.9830	-	**117**	-	0.9974
n_4	0.9848	-	**81**	-	0.9984
n_2	0.9839	-	**189**	-	0.9962
n_5	0.9910	71	128	**31**	0.9910
n_6	0.9919	79	158	**32**	0.9919
n_9	0.9822	-	**121**	-	0.9976
n_7	0.9901	**58**	158	-	0.9965
n_8	0.9875	**86**	160	-	0.9975
n_{10}	0.9857	-	**91**	-	0.9986
$cost(G)$=824, $R(G)$=0.9579 > $R_{\text{goal}}(G)$ =0.95					

表3.5是采用MRCRG和RGAGM方法调度示例功能所获得的实际可靠性和最终资源消耗成本。通过比对发现，MRCRG 和RGAGM方法都可以保障该分布式功能的可靠性目标，但是相对于MRCRG，RGAGM能达到更高的可靠性值，且可以较大的减少系统资源消耗成本。

表 3.5 MRCRG和RGAGM调度分布式功能示例结果对比

方法\结果	$R(G)$	$cost(G)$
MRCRG方法	0.9502	910
RGAGM方法	0.9579	824

3.4 实验

为了验证所提出的RGAGM方法的有效性，我们以真实的汽车功能和通过图形生成器随机生成的功能进行实验。以实际可靠性$R(G)$ 和最终的资源消耗成

本*cost*(*G*) 作为实验性能指标，并以MRCRG 算法作为对比算法。

与能耗这些可测值不同，可靠性是一个概率值，无法在运行过程中测量得出，因此可靠性实验在实际硬件平台上都不能很好的实现。由于我们的目标是在设计阶段通过使用所提出的算法计算出每个任务分配的ECU、开始时间和完成时间（即任务映射），在接下来的实现阶段可以根据已建立的任务映射来实现该功能。因此，实验中分布式功能将通过设置不同的功能参数值得出，并在仿真系统上进行测试，以反映实际部署。该仿真系统配置16个异构ECU，根据已知的参数值在2.6GHz 英特尔CPU和4GB内存的标准桌面计算机上使用Java创建出这16个异构ECU对象。

3.4.1 真实汽车功能实验

由于本研究主要针对汽车嵌入式系统的分布式功能，我们首先采用来自[76]的真实汽车功能来实验。如图所示，该功能包括6个功能模块：具有7 个任务的发动机控制器(n_1-n_7)，具有4 个任务的自动变速箱(n_8-n_{11})，具有6 个任务的防抱死制动系统(n_{12}-n_{17})，具有2 个任务的车轮角度传感器(n_{18}-n_{19})，具有5 个任务的悬浮控制器(n_{20}-n_{24})，具有7 个任务的主体工作(n_{25}-n_{31})。处理器和功能的参数为：$100\mu s \le w_{i,k} \le 400\mu s$, $100\mu s \le c_{i,j} \le 400\mu s$, $0.000001/\mu s \le \lambda_k \le 0.000009/\mu s$, $0.5Kb/ms \le \gamma_k \le 1.5Kb/ms$, 以及$\gamma_{comm} = 0.5Kb/ms$.

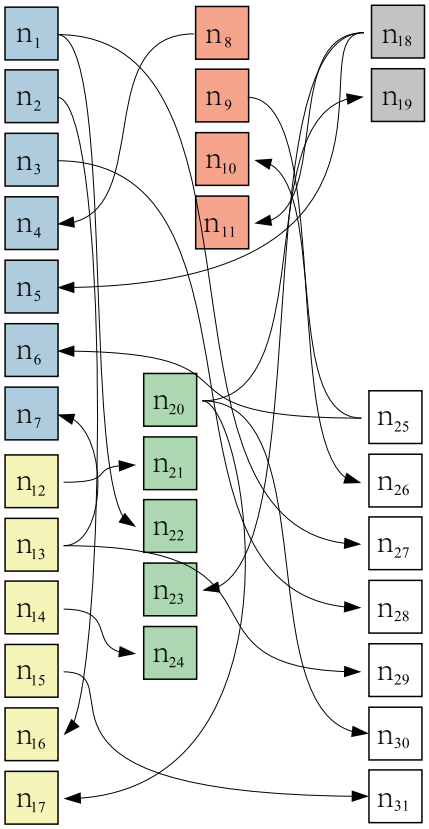

图 3.1 真实汽车功能

实验1.实验1旨在比较功能在不同可靠性目标下的最终资源消耗成本。功能的可靠性目标以0.01 的增量从0.9 增至0.99，因为这些值属于ISO 26262标准中暴露率E3和E2范围之内。得出的不同可靠性目标下的实际可靠性如图3.2所示，不同可靠性目标下的最终资源消耗成本如图3.3 所示。

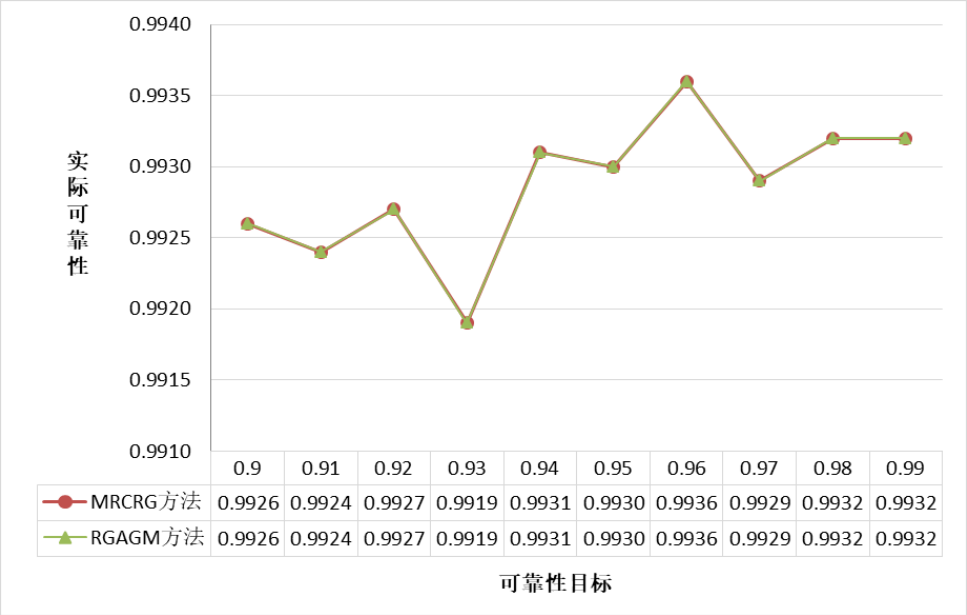

图 3.2 真实汽车功能不同可靠性目标下的实际可靠性

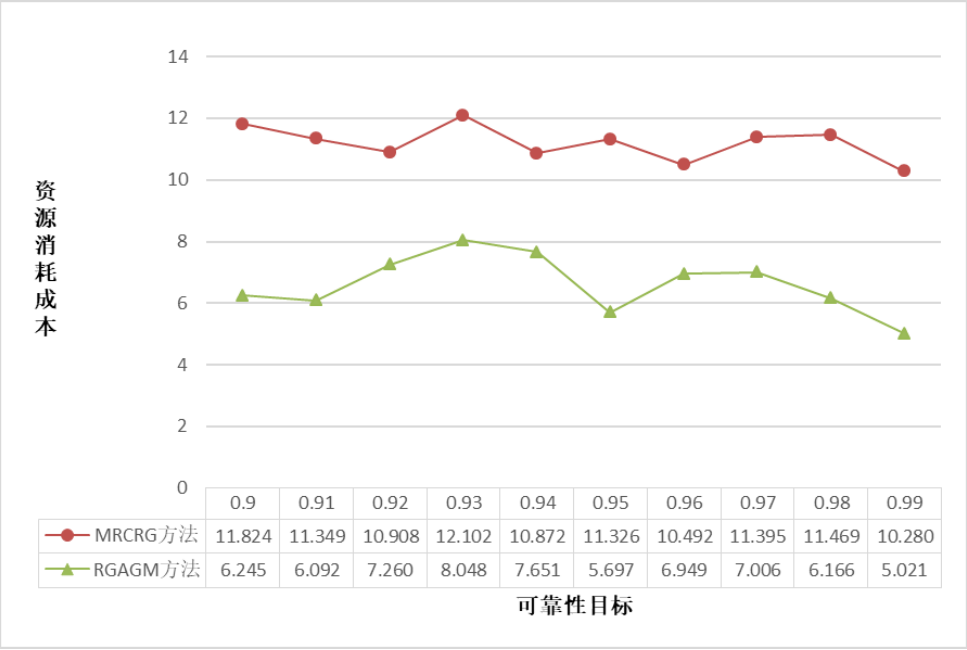

图 3.3 真实汽车功能不同可靠性目标下的资源消耗成本(单位:Mb)

根据实验结果，我们可以得出以下结论：

（1）在不同可靠性目标值下，MRCRG和RGAGM都能保障功能指定的可靠性目标，功能实际所得到可靠性随着可靠性目标值的不同而不同。

（2）MRCRG和RGAGM所获得的实际可靠性目标值都相同，从而表明MRCRG 和RGAGM 都是有效的可靠性目标保障方法。

（3）随着可靠性目标的提升，MRCRG和RGAGM都能产生相对稳定的资源

消耗成本，表明两个方法都能有效的控制系统不同可靠性目标下的资源消耗成本。而所提出的RGAGM方法显然比MRCRG 方法获得的资源消耗成本少得多。如结果所示，当可靠性目标为0.94时，RGAGM算法产生的资源消耗成本为7.651Mb，是MRCRG所产生的10872Mb 的70.4%。在可靠性目标为0.99的最佳情况下，RGAGM 产生的资源消耗成本仅为5.021Mb，这比MRCRG产生的10.280Mb 减少高达51.2%。因而，在不同可靠性目标下，RGAGM 较MRCRG能减少29.6%-51.2% 的资源消耗成本。RGAGM 较MRCRG 能降低资源消耗成本的原因正是在于它利用了几何平均值中心集中趋势的特点，改进了可靠性目标保障的策略，从而减少了不必要的资源消耗。

3.4.2 随机生成的功能实验

为了进一步验证RGAGM算法的有效性，本实验我们采用任务图生成器随机生成的分布式汽车功能进行实验。由于ECU 的异构程度也可能影响功能的资源消耗成本，那么可以通过图形生成器调整异构因子值来调整异构性。任务图生成器的参数为：形状因子集为$\partial = \{0.5, 1.0, 2.0\}$，通信计算比为$CCR = \{0.1, 0.5, 1.0, 5.0, 10.0\}$。图形生成器异构度（因子）值属于(0,1] 范围内，其中0和1分别代表最低和最高异构因子。通过输入任务数量、ECU数量、形状因子以及通信计算比四个参数，就可以生成不同的任务图模型。

实验2.本实验旨在比较不同任务数下的实际可靠性和最终资源消耗成本，利用任务图生成器生成含有不同任务数的功能任务图模型。将这些功能的可靠性目标同样设为$R_{\text{goal}}(G) = 0.95$。任务数以10的增量从30增至100。图3.4和图3.5分别给出了不同任务数下的实际可靠性和最终资源消耗成本。

根据所示实验结果，可得出一下实验结论：

（1）随着任务数量的不断增多，功能的可靠性会越来越低，这也反映了任务数量会影响功能的故障概率。但是MRCRG和RGAGM在不同任务数情况下都能保障功能既定的可靠性目标，且最终的可靠性值都相同。

（2）随着任务数量的不断增多，功能所产生的资源消耗成本也会随之增加。而采用RGAGM 算法在不同的任务数情况下相比MRCRG可减少29.4%-55.2%的资源消耗成本。例如，在最好情况下，当任务数是|N|=40 时，使用RGAGM产生的资源消耗成本为4.268Mb，相比于MRCRG产生的90523Mb可减少近55.2%。当任务数是|N|=50 时，RGAGM产生的资源消耗成本是MRCRG的62.5%；任务数是|N|=60时，RGAGM产生的资源消耗成本14.741Mb相比于MRCRG 产生的资源消耗成本20.881Mb减少29.4%。但针对于汽车嵌入式系统，该提升比例是非常可观的改进。该结果表明，RGAGM 能较大程度的减少ECU的计算资源消耗成本，是非常有效的可靠性目标保障方法且非常适用于不同任务规模的分布式汽车功能。

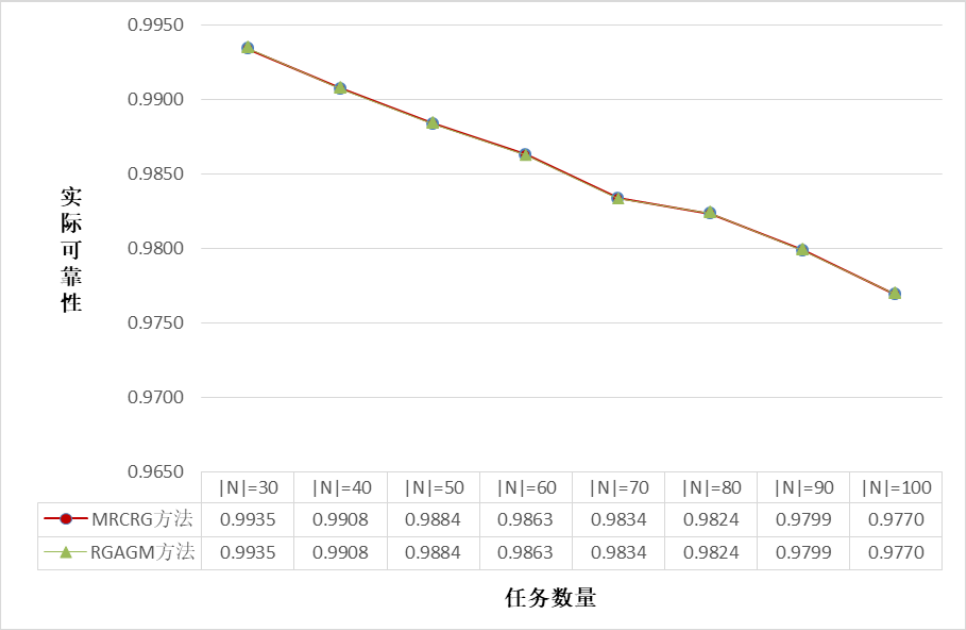

图 3.4 随机生成的功能不同任务数下的实际可靠性

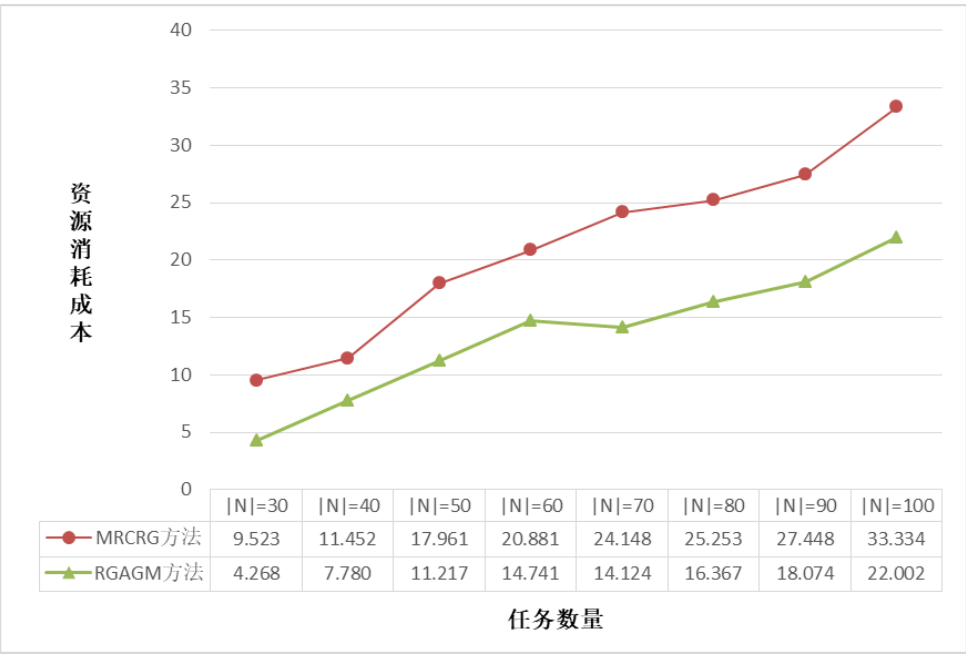

图 3.5 随机生成的功能不同任务数下的资源消耗成本(单位:Mb)

实验3.本实验比较随机生成的功能在不同可靠性目标下的资源消耗成本，利用任务图生成器生成一个任务数为100的功能任务图模型。功能的可靠性目标以0.01的增量从0.9增至0.99。图3.6和3.7 是不同可靠性目标下的实际可靠性和最终资源消耗成本实验结果。

结果表明，在不同可靠性目标的条件下，采用RGAGM与MRCRG方法所获得的实际可靠性相同，都能有效保障功能既定的可靠性目标值。随着功能可靠性目标值的增加，RGAGM 相较于MRCRG 在所有情况下都能获得较少的资源消耗成本。在最好的情况下，当功能可靠性目标为0.96时，MRCRG 消耗35.571Mb 资源

成本，RGAGM 仅消耗26.531Mb，RGAGM方法可减少多达25.4% 的资源消耗成本。另外，在标准ISO 26262 暴露率E3 中规定了可靠性目标0.99，但由于该值大于功能的最大可靠性值，因而此处不将其列出。

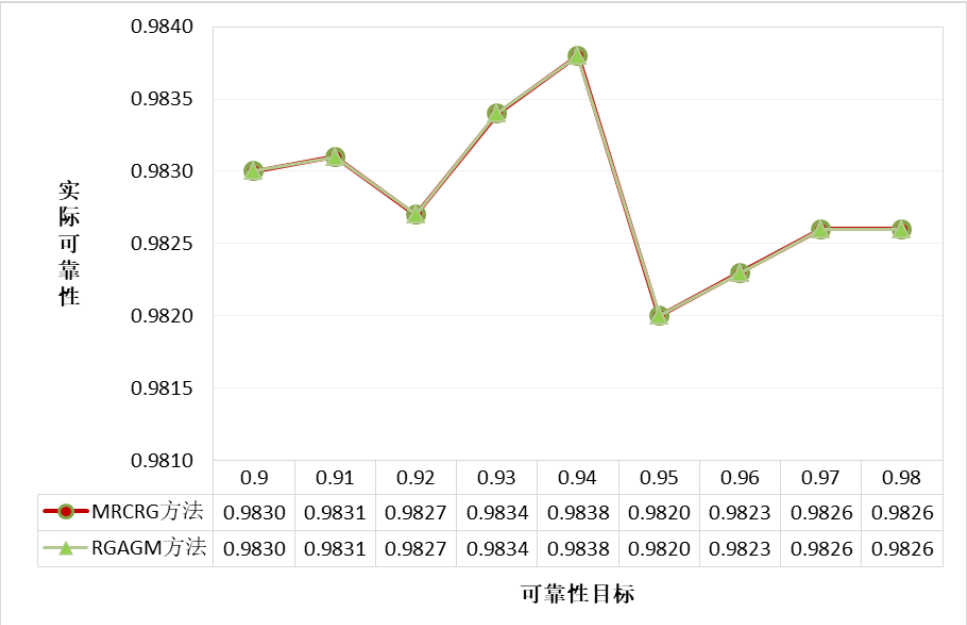

图 3.6 随机生成的功能不同可靠性目标下的实际可靠性

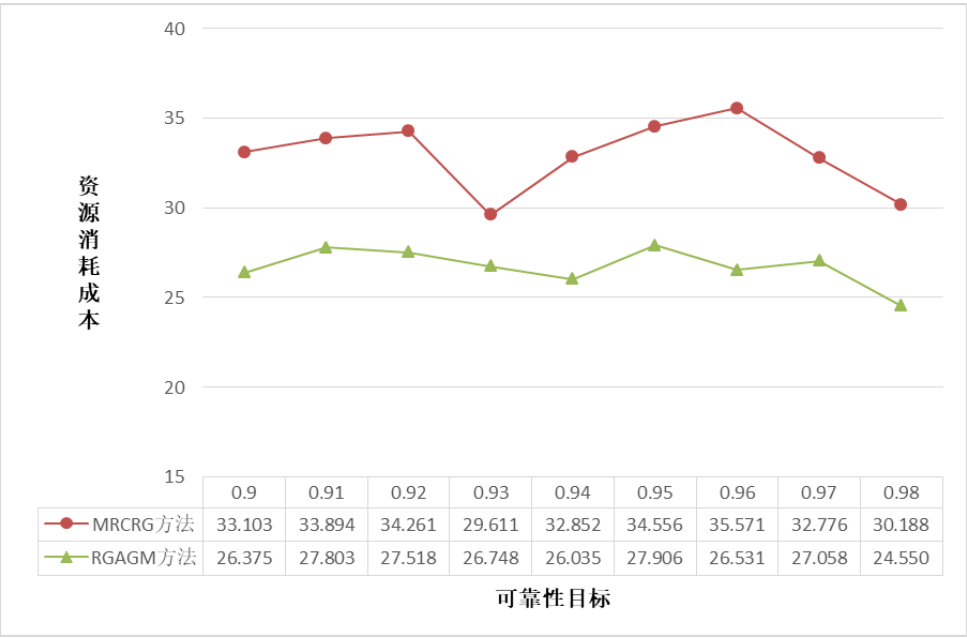

图 3.7 随机生成的功能不同可靠性目标下的资源消耗成本(单位:Mb)

3.5 小结

本章针对汽车功能安全可靠性问题，提出了一种基于几何均值的可靠性目标保障方法RGAGM。首先建立了分布式汽车功能、可靠性和资源消耗成本模型。然后，我们针对未分配任务提出一种基于几何平均值的可靠性预分配，通过预分配机制将功能的可靠性目标转换成每个任务的可靠性目标，并最小化系统资源消

耗成本。仿真实验基于真实汽车功能和随机生成的分布式功能两类DAG图进行。真实汽车功能具有一定的任务数量，通过比较不同可靠性目标来分析RGAGM的有效性，而随机生成的功能通过调整不同的参数来比较指定可靠性目标不同任务数以及指定任务数不同可靠性目标下的效果。与现有的MRCRG方法相比，RGAGM在保障汽车功能可靠性目标的前提下，在降低资源消耗成本方面取得了显著的提高。从广义上讲，RGAGM可以有效地改善汽车嵌入式系统设计阶段的资源优化问题。

第4章　容错机制下的高效可靠性目标保障调度

尽管我们提出的算法在不采取容错技术的情况下可以保障分布式汽车功能既定的可靠性目标，但是当可靠性目标超过功能可靠性最大值$R_{\max}(G)$时，就无法满足给定的可靠性目标。由于对于每个任务都只有一个副本，所以不采用容错技术功能能达到的最大可靠性目标值就是$R_{\max}(G)$。因此，当分布式汽车功能对可靠性要求较高时，采用容错机制来保障汽车功能的高度安全可靠就很有必要。本章我们则采用基于任务复制的容错机制，提出容错机制下高效的可靠性目标保障方法。

4.1　相关模型

采用容错机制时，每个任务会有多个备份副版本任务，只要一个副版本执行成功，任务则为执行成功。因此，对任务进行复制的容错机制下的可靠性模型就会相较于非容错情况不同。由于容错本身就是一种资源的浪费，以资源消耗成本作为性能评价指标不能良好的体现提出的可靠性目标保障方法的有效性。而新一代汽车电子系统是集资源关键、安全关键与时间关键这些特性于一体的系统，且响应时间是ISO 26262标准提出的重要功能安全属性，因而本章采用响应时间作为评价提出的可靠性目标保障方法的性能指标。

4.1.1 容错机制下的可靠性模型

在容错机制下，一个任务有$|U|$副本（包括主版本和副版本），每个任务的最小可靠性值是指该任务在所有ECU 上能获得的可靠性值的最小值，其计算方法和公式(3.2) 一致：

$$R_{\min}(n_i) = \min_{u_k \in U} R(n_i, u_k), \tag{4.1}$$

而任务的最大可靠性则为：

$$R_{\text{fmax}}(n_i) = 1 - \prod_{k=1}^{|U|} (1 - R(n_i, u_k)). \tag{4.2}$$

假定任务n_i有num_i个副版本，至少一个副版本成功则任务调度成功，那么任务n_i的实际可靠性表示为

$$R(n_i) = 1 - \prod_{k=1}^{num_i} (1 - R(n_i, u_k)), \tag{4.3}$$

其中u_{k_l}任务n_i表示第l个副版本所分配的ECU。

CAN协议不仅提供了物理层的差分传输规范，而且通过在数据链路层中使用CRC和ACK段提供了子包验证规则。这些措施可以确保CAN 数据包的误码率低于10^{-9}，远低于ECU 10^{-6} 的误码率。因此，本研究仅考虑ECU故障而不考虑通信故障。功能G的可靠性是所有任务可靠性的积，即

$$R(G) = \prod_{n_i \in N} R(n_i). \tag{4.4}$$

其中，功能G的最小可靠性值与非容错机制下的一致，即同公式(3.5)一致：

$$R_{\min}(G) = \prod_{n_i \in N} R_{\min}(n_i), \tag{4.5}$$

而由于容错机制下任务的最大可靠性发生改变，从而功能G的最大可靠性值变为：

$$R_{\mathrm{fmax}}(G) = \prod_{n_i \in N} R_{\mathrm{fmax}}(n_i), \tag{4.6}$$

4.1.2 响应时间模型

由于响应时间是重要的功能安全属性，因而我们将响应时间作为评价指标。对于DAD模型的功能来说，响应时间即为出口任务n_{exit} 的完成时间。本研究所采用的响应时间模型即为上述异构最早完成时间算法（HEFT）中所采用的模型，任务n_i 在ECUu_k 上的最早完成时间EFT 为

$$EFT(n_i, u_k) = EST(n_i, u_k) + w_{i,k}, \tag{4.7}$$

其中$EST(n_i, u_k)$是指任务n_i在ECUu_k上的最早开始时间EST，计算方法为

$$\begin{cases} EST(n_{\mathrm{entry}}, u_k)=0, \\ EST(n_i, u_k)=\max\left\{ avail[k], \max\limits_{n_h \in pred(n_i)} \{AFT(n_h)+c'_{h,i}\} \right\}, \end{cases} \tag{4.8}$$

其中$pred\,(n_i)$是指任务n_i的直接前驱任务集，$avail[k]$ 是指ECU u_k 可以准备执行任务的最早可用时间，$AFT(n_h)$ 是指任务n_h 的实际完成时间（Actual Finish Time），且$c'_{h,i}$表示任务n_h 和n_i 之间的WCRT。如果任务n_h 和n_i 被分配至相同一ECU 上，那么$c'_{h,i} = 0$; 反之，$c'_{h,i} = c_{h,i}$。

4.2 高效的可靠性目标保障算法

4.2.1 现有HRRTM方法

文献[46]中，Xie等人创新性的提出了一种容错机制下任务的预分配机制，提出可靠性目标保障下的冗余最小化方法Heuristic Replication for Redundancy Minimization（HRRM）。该调度算法同样分为任务排序和处理器选择两个步骤，

采用向上排序值对任务进行排序，通过预分配机制将功能的可靠性需求转换为任务的可靠性需求，每个任务逐步的选择具有最高可靠性值的副版本及其对应的处理器，直到满足任务的可靠性需求。我们将HRRM 的可靠性保障策略应用至响应时间最小化，本研究中称为响应时间最小化的启发式复制算法（HRRTM）。依据前面所述，对于未调度任务$n_{s(z)}$，HRRTM针对定义了一个可靠性上界值，即

$$R_{\mathrm{up}}(n_{s(z)}) = \sqrt[|N|]{R_{\mathrm{goal}}(G)}. \tag{4.9}$$

其中$R_{\mathrm{goal}}(G)$表示容错机制下功能的可靠性目标，当调度任务$n_{s(y)}$ 时，HRRTM 将可靠性上界值预分配给未调度任务，从而得到任务$n_{s(y)}$ 的可靠性目标

$$\begin{aligned} R_{\mathrm{goal}}(n_{s(y)}) &= \frac{R_{\mathrm{goal}}(G)}{\prod_{x=1}^{y-1} R(n_{s(x)}) \times \prod_{z=y+1}^{|N|} R_{\mathrm{up}}(n_{s(z)})} \\ &= \frac{R_{\mathrm{goal}}(G)}{\prod_{x=1}^{y-1} R(n_{s(x)}) \times \prod_{z=y+1}^{|N|} \sqrt[|N|]{R_{\mathrm{goal}}(G)}}. \end{aligned} \tag{4.10}$$

作者采用$R_{\mathrm{up}}(n_{s(z)})$而不是$R_{\mathrm{fmax}}(n_i)$作为可靠性预分配值，是因为当系统中存在大量ECU 时，$R_{\mathrm{fmax}}(n_i)$ 非常大且接近于1，那么容错机制中$R_{\mathrm{fmax}}(n_i)$是不平衡的。

4.2.2 容错下的几何平均值

尽管HRRTM能减小不平衡，但该方法仍可以进一步优化。本节我们提出容错机制下基于几何平均值可靠性预分配方法如下：

（1）首先我们定义功能的几何平均值。依据上述定义，几何平均值是指|N| 个值的乘积开|N| 次方。考虑到每个任务的最小、上界、最大可靠性值都是已知的，我们首先定义|N| 个值：

$$\begin{aligned} t_1 &= \frac{R_{\mathrm{up}}(n_1)}{R_{\mathrm{min}}(n_1) \times R_{\mathrm{max}}(n_1)}, \\ t_2 &= \frac{R_{\mathrm{up}}(n_2)}{R_{\mathrm{min}}(n_2) \times R_{\mathrm{max}}(n_2)}, \\ &\cdots, \\ t_{|N|} &= \frac{R_{\mathrm{up}}(n_{|N|})}{R_{\mathrm{min}}(n_{|N|}) \times R_{\mathrm{max}}(n_{|N|})}. \end{aligned} \tag{4.11}$$

因此，功能的几何平均值为

$$\begin{aligned} GM(G) &= \sqrt[|N|]{t_1 \times t_2 \times, ..., \times t_{|N|}} \\ &= \sqrt[|N|]{\prod_{i=1}^{|N|} \frac{R_{\text{up}}(n_i)}{R_{\min}(n_i) \times R_{\text{fmax}}(n_i)}} \\ &= \sqrt[|N|]{\frac{R_{\text{goal}}(G)}{R_{\min}(G) \times R_{\text{fmax}}(G)}}. \end{aligned} \tag{4.12}$$

由于$R_{\text{fmax}}(G)$非常接近1且不平衡，我们用$R_{\text{up}}(G)$代替该参数。这种替换是指HRRTM 方法中$R_{\text{fmax}}(n_{s(z)})$被$R_{\text{up}}(n_{s(z)}) = \sqrt[|N|]{R_{\text{goal}}(G)}$替换。于是该功能的几何平均值就转换为：

$$GM'(G) = \sqrt[|N|]{\frac{R_{\text{goal}}(G)}{R_{\min}(G) \times R_{\text{up}}(G)}}. \tag{4.13}$$

（2）定义未调度任务任务$n_{s(z)}$的可靠性预分配值，将其设为：

$$R_{\text{pre}}(n_{s(z)}) = GM'(G) \times R_{\min}(n_{s(z)}) \times R_{\text{fmax}}(n_{s(z)}). \tag{4.14}$$

由于在HRRTM中$R_{\text{fmax}}(n_{s(z)})$被$R_{\text{up}}(n_{s(z)})$所替换，我们同样让$R_{\text{up}}(n_{s(z)})$替换$R_{\text{fmax}}(n_{s(z)})$，那么任务$n_{s(z)}$的可靠性预分配值转变为

$$\begin{aligned} R'_{\text{pre}}(n_{s(z)}) &= GM'(G) \times R_{\min}(n_{s(z)}) \times R_{\text{up}}(n_{s(z)}) \\ &= \sqrt[|N|]{\frac{1}{R_{\min}(G)}} \times R_{\min}(n_{s(z)}) \times R_{\text{up}}(n_{s(z)}) \\ &= \sqrt[|N|]{\frac{R_{\text{goal}}(G)}{R_{\min}(G)}} \times R_{\min}(n_{s(z)}). \end{aligned} \tag{4.15}$$

4.2.3 可靠性目标保障

基于上述所定义的未调度任务的可靠性预分配值，我们将功能可靠性目标转换成任务可靠性目标，在容错机制下实现响应时间最小化。依据上述过程，当调度任务$n_{s(y)}$时，其可靠性目标为

$$R'_{\text{goal}}(n_{s(y)}) = \frac{R_{\text{goal}}(G)}{\prod_{x=1}^{y-1} R(n_{s(x)}) \times \prod_{z=y+1}^{|N|} R'_{\text{pre}}(n_{s(z)})}. \tag{4.16}$$

由于任务$n_{s(y)}$的最小可靠性为$R_{\min}(n_{s(y)})$，那么可靠性目标$R'_{\text{req}}(n_{s(y)})$必须满足

$$R'_{\text{goal}}(n_{s(y)}) = \max\left\{R_{\min}(n_{s(y)}), R'_{\text{goal}}(n_{s(y)})\right\}. \tag{4.17}$$

由于任务$n_{s(y)}$的可靠性上界为$R_{\text{up}}(n_{s(y)})$，那么可靠性目标$R'_{\text{req}}(n_{s(y)})$必须进一步满足

$$R'_{\text{goal}}(n_{s(y)}) = \min\left\{R_{\text{up}}(n_{s(y)}), R'_{\text{goal}}(n_{s(y)})\right\}. \tag{4.18}$$

对于任何一个任务$n_{s(y)}$，当且仅当

$$R\left(n_{s(y)}\right) \geqslant R'_{\text{goal}}\left(n_{s(y)}\right), \tag{4.19}$$

那么功能实际可靠性值$R(G)$就会大于等于$R'_{\text{goal}}(G)$.

定理2: 分布式汽车的任一功能G的任一任务$n_{s(y)}$总能找到一个ECU分配以满足:

$$R(G) = \prod_{x=1}^{y-1} R(n_{s(x)}) \times R(n_{s(y)}) \times \prod_{z=y+1}^{|N|} R'_{\text{pre}}(n_{s(z)}) \geqslant R_{\text{goal}}(G). \tag{4.20}$$

证明:为证明上述定理的正确性，我们只需证明所有任务预先分配的可靠性值的乘积大于或等于功能的可靠性目标即可。首先，令所有任务预分配的可靠性值的积为

$$R'_{\text{pre}}(G) = \prod_{z=1}^{|N|} R'_{\text{pre}}(n_{s(z)}). \tag{4.21}$$

如果我们可以证明$R'_{\text{pre}}(G)$ 大于等于$R_{\text{goal}}(G)$，那么该可靠性预分配方法就是可行的并证明了该定理。然后，我们将公式Eq. (4.15)代入公式Eq. (4.21)，代入得到:

$$\begin{aligned} R'_{\text{pre}}(G) &= \prod_{z=1}^{|N|} R'_{\text{pre}}(n_{s(z)}) \\ &= \prod_{z=1}^{|N|} \left(\sqrt[|N|]{\frac{R_{\text{goal}}(G)}{R_{\min}(G)}} \times R_{\min}(n_{s(z)}) \right) \\ &= \frac{R_{\text{goal}}(G)}{R_{\min}(G)} \times R_{\min}(G) = R_{\text{goal}}(G). \end{aligned} \tag{4.22}$$

因此，只要所有任务基于几何平均值的可靠性预分配值之积$R'_{\text{pre}}(G)$ 等于$R_{\text{goal}}(G)$，任一任务就能找到一个ECU 来满足功能可靠性$R_{\text{goal}}(G)$。

4.2.4 最小化响应时间

在确定每个任务的可靠性目标后，需要为每个任务选择近似的副本，以保证其可靠性目标，并缩短其完成时间。

主副版本复制包括被动复制和主动复制两种方式[46,71,72]。容错调度有两种类型，即编译时间严格调度和运行时间通用调度[46]。对于编译时间严格调度，每个任务必须等待其前驱任务的所有副本完成(包括成功和失败)才能开始执行。对于运行时间通用调度，只要前驱任务的一个副版本完成执行，就可以立即开始执行当前任务。与[46,71,72] 类似，本研究在设计阶段采用主动复制和编译时间严格调度，以获得可预测的调度结果。

在主动复制和严格调度的基础上，容错机制下任务n_i 在ECUu_k上的EFT为

$$EFT(n_i^{\beta}, u_k) = EST(n_i^{\beta}, u_k) + w_{i,k}, \tag{4.23}$$

其中n_i^{β}表示任务n_i选择的第β个副版本。任务n_i 在ECUu_k 上的EST 更新为

$$\begin{cases} EST(n_{\text{entry}}, u_k) = 0; \\ EST(n_i, u_k) = \max \left\{ avail[k], \max\limits_{n_h \in pred(n_i), \alpha \in [1, num_h]} \left\{ AFT(n_h^{\alpha}) + c'_{h,i} \right\} \right\}, \end{cases} \tag{4.24}$$

num_h表示任务n_h指定的副版本的数量，而num_i表示任务n_i实际分配的副版本的数量。因此，任务n_i 的可靠性则为

$$R(n_i) = 1 - \prod_{\beta=1}^{num_i} \left(1 - R\left(n_i^{\beta}, u_{pr(n_i^{\beta})}\right)\right), \tag{4.25}$$

其中$u_{pr(n_i^{\beta})}$表示任务n_i^{β}所分配的ECU。我们迭代地将当前任务n_i的副本分配给具有最小EFT的可用ECU，直到满足任务n_i 的可靠性目标。也就是说，在每次迭代中任务n_i 分配的ECU$u_{pr(n_i^{\beta})}$ 和相应的$EFT(n_i, u_{pr(n_i^{\beta})})$为

$$\begin{aligned} AFT(n_i) &= EFT(n_i, u_{pr(n_i^{\beta})}) \\ &= \min_{u_k \in U, u_k \text{ is available}} \{EFT(n_i, u_k)\}, \end{aligned} \tag{4.26}$$

其中，“u_k is available”意味着没有将n_i的其他副版本分配给u_k，因为根据主动复制原理[46,71,72]，不允许在同一ECU 上复制多个副版本。功能的最终响应时间是出口任务n_{exit}的副版本的AFT，且该副版本在n_{exit} 的所有副版本中具有最大的AFT。因此，功能G 的最终响应时间为:

$$RT(G) = AFT(n_{\text{exit}}) = \max_{\beta \in [1, num_{\text{exit}}]} \{AFT(n_{\text{exit}}^{\beta})\}. \tag{4.27}$$

4.2.5 GMFRP算法

在上述分析的启发下，我们提出了一种容错机制下更高效的可靠性目标保障方法——基于几何平均值的容错可靠性预分配方法Geometric Mean-based Fault-tolerant Reliability Pre-assignment （GMFRP），并在算法4.2.1中给出了具体算法步骤。

GMFRP主要思想是为每个未调度的任务预分配基于几何平均值的可靠性值，从而将功能的可靠性目标转换为每个任务的可靠性目标。同时使用任务可靠性上界值$R_{\text{up}}(n_{s(z)})$ 替换最大可靠性$R_{\text{fmax}}(n_{s(z)})$，从而减少容错机制中的不平衡性。然后，GMFRP 逐步的分配当前任务$n_{s(y)}$ 的副版本到具有最小EFT的可用ECU 上，直到满足任务$n_{s(y)}$ 的可靠性目标。从而通过容错机制在满足功能可靠性目标的前提下，实现响应时间的最小化。GMFRP方法的主要细节如下:

1）第1行，GMFRP按向上排序值$rank_{\text{u}}$的降序对任务列表*task_list* 中的所有任务进行排序。

2）第2-3行，根据公式(4.13)，GMFRP计算出功能的几何平均值。

3）第4-6行，根据公式(4.15)，GMFRP计算出基于几何平均值的未调度任务的可靠性预分配值。

4）第7-9行，对所有任务进行遍历，根据公式(4.16)–(4.18)，计算当前任务$n_{s(y)}$的可靠性目标。

5）第15-18行，GMFRP迭代的将当前任务$n_{s(y)}$的副版本分配给具有最小EFT的ECU 上，直到满足该任务的可靠性目标。

6）第20-21行，GMFRP分别计算出功能的实际可靠性和响应时间。

算法 4.2.1 GMFRP方法

Input: $U = \{u_1, u_2, ..., u_{|U|}\}$, G, $R_{\text{goal}}(G)$
Output: $R(G)$, $RT(G)$

1: 根据式(3.12)对功能G中的任务按照$rank_{\text{u}}$的降序排列，并放入到任务优先级队列$task_list$中;
2: 使用式(3.5)计算$R_{\min}(G)$;
3: 使用式(4.13)计算$GM(G)'$;
4: **for** (z=1;z⩽|N|;z++) **do**
5: 使用式(4.15)计算$R'_{\text{pre}}(n_{s(z)})$;
6: **end for**
7: **for** (y=1;y⩽|N|;y++) **do**
8: $n_{s(y)} \leftarrow task\_list.out()$;
9: 使用式(4.16)–(4.18)计算$R'_{\text{goal}}(n_{s(y)})$;
10: **for** (each ECU $u_k \in U$) **do**
11: 使用式(3.1)计算$R(n_{s(y)}, u_k)$;
12: 使用式(4.23)计算$EFT(n_{s(y)}, u_k)$;
13: **end for**
14: $R(n_{s(y)}) \leftarrow 0$; //初始值为0
15: **while** ($R(n_{s(y)}) < R'_{\text{goal}}(n_{s(y)})$) **do**
16: 使用式(4.26)分配任务$n_{s(y)}$的可用副版本至具有最小EFT的ECU 上;
17: 使用式(4.25)计算$R(n_{s(y)})$;
18: **end while**
19: **end for**
20: 使用式(4.4)计算功能G的最终可靠性$R(G)$;
21: 使用式(4.27)计算功能G的响应时间$RT(G)$.

GMFRP的时间复杂度分析如下。调度功能的所有|N|个任务需要遍历所有的任务，这一步骤的时间复杂度为$O(|N|)$；计算每个任务在不同ECU上的资源消耗成本并得出最小资源消耗成本需要每个任务遍历所有的ECU，这一步骤的时间复杂度为$O(|N| \times |U|)$。因此，GMFRP的时间复杂度为$O\left(|N|^2 \times |U|\right)$，这与RGAGM一致。

4.2.6 GMFRP算法示例

例4. 采用上述分布式功能示例解释GMFRP的过程与结果。由于非容错机制下功能示例的最大可靠性值为0.975603，本例假设可靠性目标为0.988，这是MRCRG 和RGAGM 方法无法保证的，但可以用GMFRP 方法来满足。三个ECU

的故障率仍然是$\lambda_1 = 0.0002$，$\lambda_2 = 0.0005$，和$\lambda_3 = 0.0009$。表4.1 是采用示例功能在GMFRP 方法的任务分配。

表 4.1 分布式功能示例在GMFRP方法下的任务分配表

n_i	$R_{goal}(n_i)$	$R(n_i,u_1)$	$EFT(n_i,u_1)$	$R(n_i,u_2)$	$EFT(n_i,u_2)$	$R(n_i,u_3)$	$EFT(n_i,u_3)$	$R(n_i)$
n_1	0.998793	**0.997204**	**14**	0.992032	16	**0.991933**	**9**	0.999977
n_3	0.998150	**0.997802**	**32**	**0.993521**	**39**	0.983045	45	0.999986
n_4	0.994540	**0.997403**	**45**	0.996008	47	**0.984816**	**40**	0.999961
n_2	0.990077	**0.997403**	**58**	0.990545	58	0.983931	58	0.997403
n_5	0.995301	0.997603	70	**0.993521**	**52**	**0.991040**	**50**	0.999942
n_6	0.998793	0.997403	71	**0.992032**	**68**	**0.991933**	**59**	0.997403
n_9	0.992665	**0.996406**	**83**	0.994018	86	0.982161	94	0.996406
n_7	0.998002	0.998601	90	**0.992528**	**83**	**0.990149**	**73**	0.999926
n_8	0.997141	**0.999000**	**88**	0.994515	94	0.987479	97	0.999000
n_{10}	0.995410	0.995809	121	**0.996506**	**106**	0.985703	116	0.996506
$R(G) = 0.989088 > R_{goal}(G) = 0.988, RT(G) = 106$								

（1）如表中结果所示，与只有一个副版本的非容错MRCRG和RGAGM方法不同，容错GMFRP方法为每个任务选择一个或多个副版本，这就能大大提高功能的可靠性。

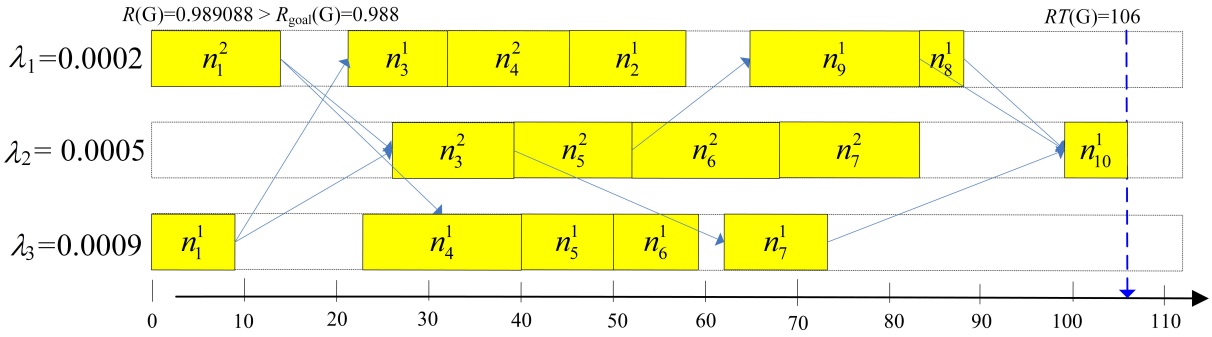

图 4.1 分布式功能示例在GMFRP方法下的任务映射图

（2）GMFRP首先根据确定的任务优先级调度任务n_1。根据公式(4.16)–(4.18)可计算出任务n_1的可靠性目标为0.998793。根据公式(3.1)，任务n_1在ECUu_1、u_2和u_3上的可靠性值为0.997204、0.992032和0.991933。根据公式(4.7)，任务n_1在ECUu_1、u_2和u_3上的最早完成时间EFT为14、16和9。由于u_3具有最小EFT值9，GMFRP首先为任务n_1选择ECUu_3，且其可靠性值为0.991933。然而0.991933仍小于可靠性目标值0.998793。因此，GMFRP进一步为任务n_1选择EFT值为14的ECUu_1。随后，根据式(4.3)，n_1的可靠性值达到0.999977(0.999977= 1- (1-0.991933)×(1-0.997204))。相应地，任务n_1在ECUu_3和u_1上的任务映射如图4.1 所示。总之，GMFRP迭代地为每个任务选择可用的副

版本和具有最小EFT的ECU，直到其可靠性要求得到保障为止。

（3）经过整个的任务分配过程，根据公式(4.4)和(4.27)，功能G的最终可靠性为0.989088，最终响应时间为106。通过该示例并与前述方法比较发现，GMFRP在容错机制下能保障较高的功能可靠性目标。最终的任务映射图如图4.1 所示。

4.3 实验

为了评价所提出的GMFRP方法的有效性，我们同样以真实的汽车功能和随机生成的功能为实验对象。前文中所提到的MaxRe和RR方法也适用于和GMFRP比较，但是在文献[46] 中，HRRM 和这两个算法进行了充分的实验比较，且这两个算法都具有一定的缺点，因而本研究我们不再提供MaxRe 和RR 方法的结果，而着重于GMFRP 相较于HRRMT 的详细改进。鉴于本研究的目的在于缩短汽车功能的响应时间，同时确保其可靠性目标，因此我们以功能的实际可靠性值和响应时间作为评价指标。

由于本研究的重点是设计阶段，本阶段所使用的功能参数是基于实际部署的，我们以实际汽车系统的参数值作为实验数据。在1 μs 的时间单位内，每个任务的故障率在10^{-6}–9×10^{-6} 之间，而任务的WCET 和消息的WCRT 都在100 μs–400 μs 范围之间，且这些值是以均匀分布产生的。

与能耗这些可测值不同，可靠性实验在实际硬件平台上都不能很好的实现。这是因为可靠性是一个概率值，无法在运行过程中测量可靠性值。我们的目标是在设计阶段通过使用所提出的算法计算出每个任务分配的ECU、开始时间和完成时间（即任务映射）。在接下来的实现阶段可以根据已建立的任务映射来实现该功能。因此，本研究中的分布式功能将在基于上述功能参数值的仿真系统上进行测试，以反映实际部署。该模拟系统配置16 个异构ECU，根据已知的参数值在2.6GHz 英特尔CPU 和4GB 内存的标准桌面计算机上使用Java 创建出这16 个ECU 对象。

4.3.1 真实汽车功能实验

由于本研究主要针对汽车电子系统的功能安全，和实验1一样，我们首先采用来自[76] 的真实汽车功能来实验。如图所示，该功能包括6个功能模块：具有7个任务的发动机控制器(n_1-n_7)，具有4 个任务的自动变速箱(n_8-n_{11})，具有6 个任务的防抱死制动系统(n_{12}-n_{17})，具有2 个任务的车轮角度传感器(n_{18}-n_{19})，具有5 个任务的悬浮控制器(n_{20}-n_{24})，具有7 个任务的主体工作(n_{25}-n_{31})。处理器和功能的参数为：$100\mu s \le w_{i,k} \le 400\mu s$, $100\mu s \le c_{i,j} \le 400\mu s$, $0.000001/\mu s \le \lambda_k \le 0.000009/\mu s$, $0.5Kb/ms \le \gamma_k \le 1.5Kb/ms$, 以及$\gamma_{comm} = 0.5Kb/ms$.

实验4.该实验旨在比较真实汽车功能在不同可靠性目标下的最终响应时间

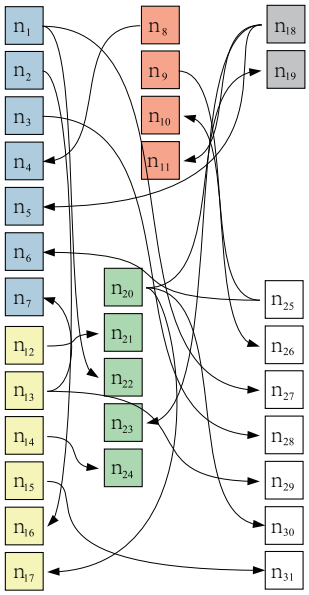

图 4.2 真实汽车功能.

值。我们根据上述参数值生成了1000 种汽车功能，然而，根据我们的计算，对于所有1000 个功能在不使用容错情况下最大可以达到的可靠性值在0.992-0.993范围内。而对于一个高度安全-关键的功能，其暴露率层次为E1，可靠性要求达到0.999999 甚至更高，非容错方法在此情况下无法使用，因而我们用容错GMFRP方法来分析结果。可靠性目标从0.999增至0.99999999，因为根据表3.1这些可靠性目标值属于ISO 26262 标准暴露率层次E1 范围之内。实验结果如表4.2所示。

表 4.2 1000个真实功能使用HRRTM和GMFRP的实验结果

可靠性目标	方法	较短响应时间次数	平均响应时间(单位: μs)
0.999	HRRTM	223	921.510
	GMFRP	431	911.419
0.9999	HRRTM	36	939.446
	GMFRP	497	919.246
0.99999	HRRTM	337	812.953
	GMFRP	465	805.140
0.999999	HRRTM	253	1009.605
	GMFRP	675	971.372
0.9999999	HRRTM	300	1154.407
	GMFRP	668	1106.138
0.99999999	HRRTM	106	1227.347
	GMFRP	843	1105.908

我们根据真实汽车功能在不同可靠性目标下采用不同参数形成各1000个不同

任务模型，首先比较不同可靠性目标下1000个结果中HRRTM 和GMFRP 方法获得较短响应时间的次数，结果如图4.3所示。

根据实验对比，在1000次比较中，GMFRP在不同的可靠性目标上有431-843次的概率比HRRTM 获得更短的响应时间，而HRRTM 获得比GMFRP 更短的响应时间的概率为36-337。当可靠性目标为0.99999999时，GMFRP的优势更加明显，有843 次得出更小的响应时间。

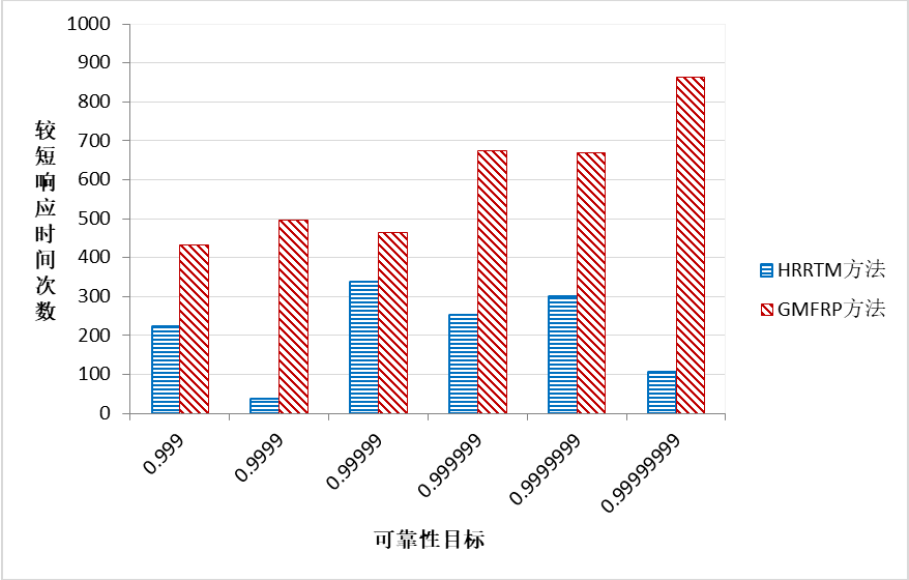

图 4.3 1000个真实功能不同可靠性目标下获得的较短响应时间次数

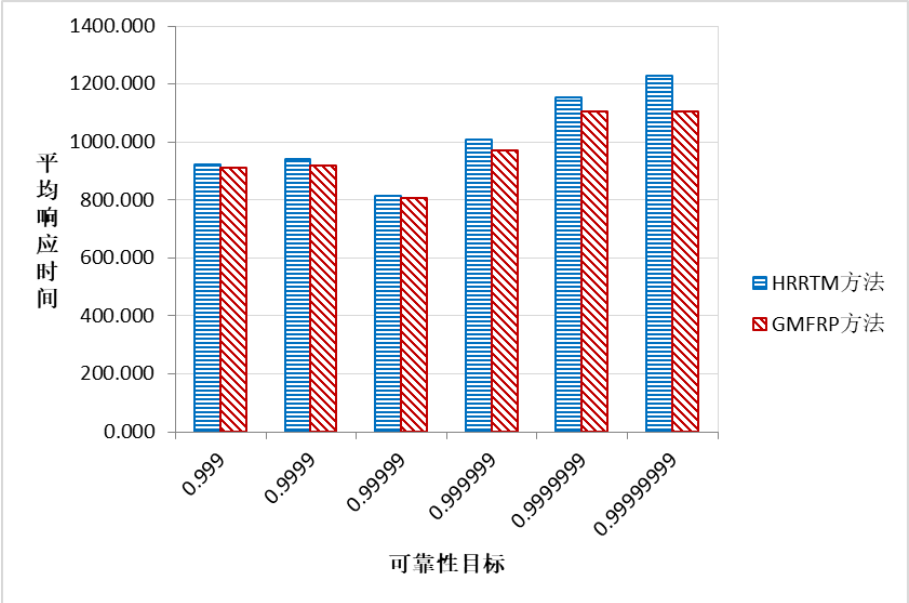

图 4.4 1000个真实功能不同可靠性目标下的平均响应时间

由于HRRTM和GMFRP都能获得一定次数的较短响应时间，为了进一步进行验证，我们将两个方法不同可靠性下1000次实验的平均响应时间进行对比，实验结果如图4.4 所示。在不同可靠性目标情况下，使用GMFRP 获得的平均响应时间值仍然比通过使用HRRTM 获得的平均响应时间短。随着可靠性目标的增加，使

用HRRTM和GMFRP所获得的平均响应时间也随之增加。总的来说，相较于HRRTM，GMFRP可缩短9.89%的响应时间。尽管该优化比例不是很高，但是对于一个高度安全关键且时间敏感的汽车功能，响应时间都是精确微秒级别，这个提升结果就显得极为可观。例如，当可靠性目标为0.99999999时，所缩短的平均响应时间达到121.439 μs，这就反映出GMFRP非常可观的改进。

此外，HRRTM和GMFRP方法所获得的较短响应时间次数总数并未达到1000次，这是由于在少数情况下，HRRTM和GMFRP在获得相同的响应时间值。

4.3.2 随机生成的功能实验

采用单一的真实汽车功能来验证所提出的方法的高效性是不够充分的，且由于汽车系统的日益复杂，未来汽车功能预计将达到100个任务。因而为进一步验证算法的有效性，我们采用与真实汽车功能一样参数的随机生成的功能来分析结果。如上章所述，随机生成的功能可由任务图生成器生成。任务图的参数如下：

（1）通信与计算比的集合为{0.1, 0.5, 1.0, 2.0, 5.0}。

（2）ECU的异构因子集合为{0.2, 0.4, 0.5, 0.6, 0.8, 1.0}；异构因子越大，ECU异构程度越明显。

（2）形状因子是一个随机参量，该值的范围为[0.35, $\sqrt{|N|/3}$]。令形状因子集为{0.35, 1.0, 2.0, 3.0, 4.08, 4.47, 4.83, 5.16, 5.47, 5.77}。其中50个任务和100个任务的最大形状因子分别为4.08和5.77。

表 4.3 1000个随机生成的功能使用HRRTM和GMFRP的实验结果

任务数量	方法	较短响应时间次数	平均响应时间(单位: μs)
50	HRRTM	282	3,017.512
	GMFRP	717	2,897.834
60	HRRTM	306	3,751.897
	GMFRP	694	3,625.888
70	HRRTM	181	4,155.137
	GMFRP	818	4,043.44
80	HRRTM	353	4,577.684
	GMFRP	647	4,457.324
90	HRRTM	321	4,593.646
	GMFRP	678	4,442.077
100	HRRTM	352	4,687.464
	GMFRP	645	4,527.251

每个功能的参数都在上述参数范围内随机选取并合成，可靠性目标设置为0.999，功能任务的数量从50增至100，HRRTM和GMFRP方法的实验结果数据如

表4.3所示。

我们根据任务图生成器采用不同参数随机生成不同任务数量下的各1000个不同任务模型，比较同任务数量下1000个结果中HRRTM 和GMFRP 方法获得较短响应时间的次数，结果如图4.5所示。同时，由于两个方法都能在不同任务数量下获得一定数量的较短响应时间次数，将两个方法不同任务数下1000次实验的平均响应时间进行对比，实验结果如图4.6所示。通过随机功能实验对比，我们得出以下结论：

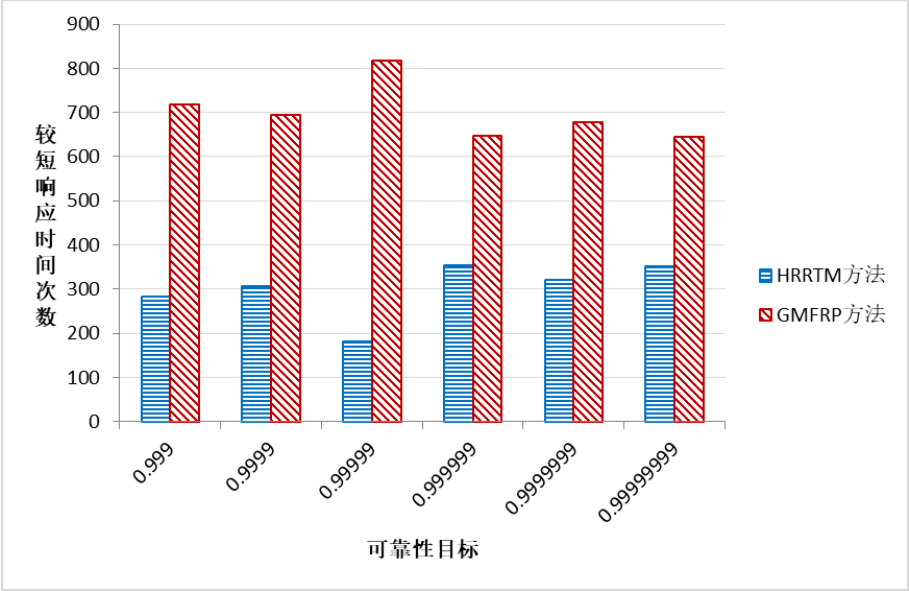

图 4.5 1000个随机功能不同任务数量下获得的较短响应时间次数

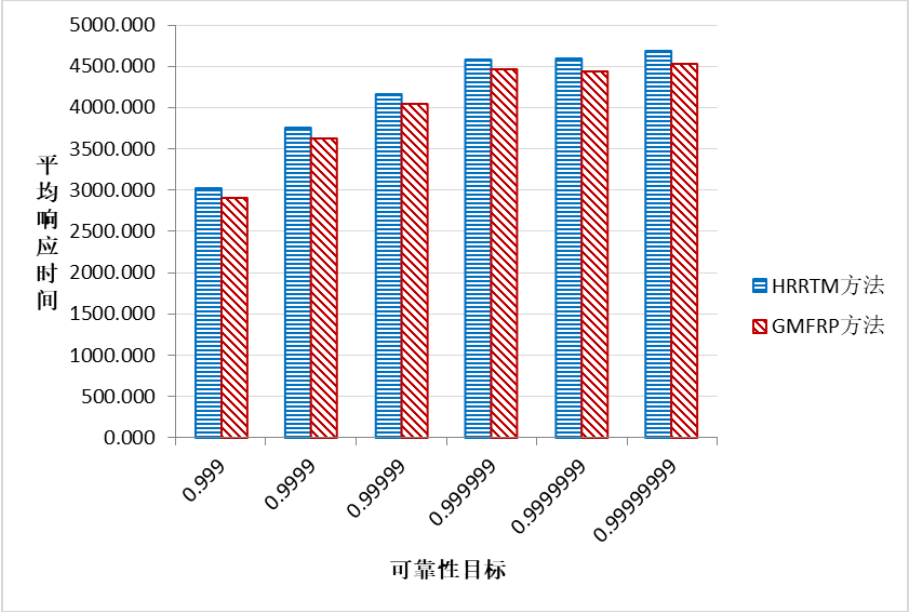

图 4.6 1000个随机功能不同任务数量下的平均响应时间

（1）对于不同的任务数量，GMFRP 都有明显的优势。在1000 次比较中，相较于HRRTM 的181-353次数，GMFRP有645-818 的次数获得更短

的响应时间。当任务数量为70时下更为明显，GMFRP获得的818次较短响应时间比HRRTM 的181次高出约77.9%。总的来说，在所有情况下GMFRP比HRRTM有45.2%-77.9%的概率获得较短响应时间。

（2）对于平均响应时间，GMFRP和HRRTM的响应时间随着任务数量的增加而不断增加，这符合任务调度的基本规律。但在不同任务数量的情况下，GMFRP方法都能获得远低于HRRTM方法的平均响应时间。GMFRP可以缩短多达160 μs 的平均响应时间，这样的提升对于时间敏感的汽车功能是十分可观的。

（3）与上述相同，由于在少数情况下，HRRTM和GMFRP获得相同的响应时间值相同，所以实验结果中不同任务数量下两者次数之和略少于1000次.

4.4 小结

鉴于非容错情况下无法保障分布式汽车功能较高的可靠性需求，本章提出一种容错机制下基于几何平均值的可靠性目标保障方法。首先，重新建立了容错机制下的可靠性模型、响应时间模型。然后针对未分配任务提出一种基于几何平均值的可靠性预分配策略GMFRP，将功能的可靠性目标转换成任务的可靠性目标，并在此基础上最小化整个响应时间。与现有HRRTM方法相比，GMFRP 能在保障分布式汽车功能可靠性目标的前提下，较大的缩短了响应时间，并能从汽车功能安全需求角度同时优化其可靠性与响应时间属性。

结　论

1.总结

随着无人驾驶、智慧城市、以及物联网的迅速发展，汽车电子系统这一典型的ACPS 不断的得以变革与发展。汽车电子系统的系统结构高度集成化，功能应用日益精细化，使得新一代汽车电子系统越来越成为学术界和工业界竞相研究的热点。新一代汽车电子系统高度安全可靠性的需求，功能安全标准ISO 26262及AUTOSAR 自适应平台的出台，人们对汽车驾驶性、舒适性、自治性以及安全可靠性提出更高的要求。

基于任务复制的容错技术是保障系统高可靠性的常用和有效手段，但对于时间关键、安全关键同时也是资源关键的的ACPS来说，可靠性目标相对较低的情况下，采取容错本身就是一种资源浪费。但当汽车功能可靠性目标较高时，就必须采取容错来保障功能的高可靠性。本文对可靠性调度问题进行了研究，详细分析了非容错和容错两种情况下的可靠性调度研究现状，归纳出其中存在的问题：1）现有研究大多采用容错以保障功能或应用的可靠性目标，对于不采用任务复制，在满足功能或应用的可靠性目标同时最小化系统资源消耗成本的研究较少；已有非容错下的相关可靠性目标保障方法虽然能达到所期望的目的，但是调度策略存在一定的悲观性；2）针对容错情况下的盲目复制问题相关研究提出了解决方案，但是可靠性模型要么未考虑DAG任务之间的优先级约束关系，要么过于贪婪导致可靠性目标分配不均，从而影响整个系统的响应能力。本文针对上述问题，重点研究了两种情况下可靠性目标保障方法，考虑了非容错情况下保障ACPS安全关键特性同时优化其资源关键特性，容错情况下保障ACPS安全关键特性同时优化其时间特性，具体成果体现在一以下几个方面：

（1）针对新一代汽车电子系统可靠性目标任务调度问题抽象出两个模型，分别是基于中央网关的ECU 分布模型以及基于DAG 图的分布式汽车功能模型，并对这两方面的调度模型进行详细阐述。广泛研究的静态任务调度算法主要分为启发式算法和随机搜索算法两大类，启发式算法作为其中性能最优的算法分为表调度启发式算法、集群计算启发式算法以及任务复制启发式算法三类。对每一类算法的主要含义进行了总结并对算法进行归纳概述，同时着重阐述了本研究所采用的基础表调度启发式算法。针对汽车功能安全可靠性这一研究重点，总结了基于分布式功能DAG 模型的可靠性调度研究进展及存在的问题。分析容错技术的原理及方式，并总结了基于容错技术的可靠性调度问题研究进展及存在的问题。

（2）提出了非容错情况下基于几何平均值的可靠性目标保障方法RGAGM（Reliability Goal Assurance Method using Geometric Mean）。该算法采用预分配机制，未每个未调度任务预分配一个可靠性值，从而将功能的可靠性目标转换成每个子任务的可靠性目标，在选择ECU时就只需考虑当前任务的可靠性目标。引入数学上的几何平均值定义，分别定义针对功能和任务两类几何平均值，并根据这两类平均值定义未调度任务的可靠性预分配值，利用预分配机制得到任务的可靠性目标。最后筛选不满足任务可靠性目标的ECU，并将任务分配至具有最小资源消耗成本的ECU上。RGAGM算法利用几何平均值所具有的中心集中趋势，解决了现有算法针对未调度任务预分配可靠性最大值造成资源浪费，分配较小值无法满足可靠性目标的悲观性问题。

（3）提出一种基于几何平均值的容错可靠性预分配策略GMFRP（Geometric Mean-based fault-tolerant Reliability Pre-assignment）。鉴于可靠性和响应时间是功能安全的两个重要属性，该算法旨在采用容错技术，在满足分布式汽车功能可靠性目标的同时，最小化系统的响应时间。GMFRP同样引入数学上的几何平均值定义，分别定义针对功能和任务两类几何平均值，并根据这两类平均值定义容错情况下的未调度任务的可靠性预分配值。采用预分配机制将功能的可靠性目标转换成每个子任务的可靠性目标，迭代的分配备份副版任务至具有最小完成时间的ECU上，直至满足任务的可靠性目标。GMFRP算法利用几何平均值所具有的中心集中趋势，解决了现有算法针对未调度任务预分配可靠性高优先级和低优先级任务不平衡问题。

（4）通过真实汽车功能和任务图生成器随机生成的任务图仿真实验表明，与以往算法相比，提出的RGAGM算法能在满足既定功能可靠性目标下较大程度的降低资源消耗成本，提出的GMFRP算法能在满足既定功能可靠性目标下较大程度的减少系统响应时间。以上算法都能从整体上满足功能所认证的可靠性目标并优化其目标性能，是高效的可靠性目标保障方法。

2.展望

本文面向新一代汽车电子系统功能安全可靠性调度问题进行了研究，提出了针对非容错和容错情况下的两种可靠性目标保障方法。由于研究时间的限制，接下来的工作可继续从以下几个方面进行：

（1）多功能满足可靠性目标保障方法

本文研究只是面向单个分布式汽车功能，而汽车电子系统是包含一系列不同子系统的混合系统，子系统之间的可靠性关键级别又有所差异。对于多个分布式汽车功能，就需要用多DAG的模型来描述任务彼此之间具有优先级约束关系的一组功能。下一步工作中，将研究多DAG混合关键级的高效可靠性调度策略问题。

（2）通信可靠性问题

本文研究仅只考虑了计算可靠性，忽略了链路之间的通信可靠性。汽车系统内的总线种类和数量都比较多，因而可能出现通信链路故障而导致通信可靠性问题。目前对融合计算可靠性和通信可靠性的研究并不多，要么不是从系统角度实现容错，要么只研究了单个冗余链路，且当前对通信可靠性的精确建模与验证，也没有通信链路复制个数的量化研究，因此融合计算可靠性和通信可靠性的研究是需要解决的关键问题。

（3）汽车电子系统信息安全问题

汽车在受到可能的随机系统功能性失效的同时，随着汽车越来越网联化的发展，系统内部也越来越存在信息攻击的风险。攻击者可以通过向终端发动网络攻击，访问车内的重要信息，甚至达到恶意控制汽车的目的，这就不仅造成车辆财产损失问题，甚至会威胁车内人员的人身安全。因此，研究汽车信息安全，避免汽车因入侵攻击而导致不可接受的伤害和风险有着重要的现实意义。此外，同时保障功能安全与信息安全而不是单一关注一方面也是本文后续工作的研究重点。

（4）分析阶段的精确WCRT分析

本文中主要面向汽车功能开发生命周期的设计阶段，因而在研究中假定最差响应时间（WCRT）为已知的值。面向分析阶段的WCRT精确分析一直是汽车电子系统的研究热点，但目前主要是获得安全但紧凑的WCRT,尚未从功能级的角度对计算和网络考虑端到端的WCRT分析，也没有对计算异构性和网络异构性的精确描述。因此后续研究中，可研究的获得更加紧凑甚至完全精确的WCRT问题。

参考文献

[1] Teepe G, Remboski D, Baker R. Towards Information Centric Automotive System Architectures. Technical report, SAE Technical Paper, 2002

[2] Xie G, Zeng G, Li Z, et al. Adaptive Dynamic Scheduling on Multifunctional Mixed-Criticality Automotive Cyber-Physical Systems. IEEE Transactions on Vehicular Technology, 2017, 66(8):6676-6692

[3] 李仁发, 谢勇, 李蕊,等. 信息-物理融合系统若干关键问题综述. 计算机研究与发展, 2012, 49(6):1149-1161

[4] Kang J S, Kim J, Lee M. Advanced driver assistant system based on monocular camera. In: Proc of IEEE International Conference on Consumer Electronics. IEEE, 2014:55-56

[5] Brown S. Overview of IEC 61508. Design of electrical/electronic/programmable electronic safety-related systems. Computing & Control Engineering Journal, 2000, 11(1):6-12

[6] Sato Y. IEC 61508 and ISO 26262: Functional Safety of Road Vehicles. Journal of Reliability Engineering Association of Japan, 2011, 33:408-415

[7] 刘佳熙, 郭辉, 李君. 汽车电子电气系统的功能安全标准ISO26262. 上海汽车, 2011(10):57-61

[8] 耿莉莉. 基于ISO 26262标准的安全关键嵌入式软件开发技术与工具: [浙江大学硕士学位论文]. 杭州: 浙江大学, 2013

[9] 张君雁, 杨国纬, 罗旭斌. 可靠性代价驱动的实时任务调度算法. 计算机科学, 2003, 30(1):53-56

[10] 刘琳. 基于分层调度的实时系统容错技术研究. [湖南大学硕士学位论文]. 长沙: 湖南大学, 2013

[11] 谢国琪, 李仁发, 刘琳,等. 异构分布式系统DAG可靠性模型与容错算法. 计算机学报, 2013, 36(10):2019-2032

[12] 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. 计算机科学, 2011, 38(3):87-92

[13] 景维鹏, 吴智博, 刘宏伟,等. 可靠性代价和Makespan驱动的分布式容错调度算法. 高技术通讯, 2012, 22(5):477-482

[14] Edward Ashford Lee, Sanjit Arunkumar Seshia. 嵌入式系统导论: CPS方法: a cyber-physical systems approach. 北京: 机械工业出版社, 2012

[15] Anssi S, Tucci-Piergiovanni S, Kuntz S, et al. Enabling Scheduling Analysis for AUTOSAR Systems. In: Proc of IEEE International Symposium on Object/component/service-Oriented Real-Time Distributed Computing. IEEE, 2011:152-159

[16] Xie G, Chen Y, Liu Y, et al. Resource Consumption Cost Minimization of Reliable Parallel Applications on Heterogeneous Embedded Systems. IEEE Transactions on Industrial Informatics, 2017, 13(4):1629–1640

[17] Xie G, Zeng G, Kurachi R, et al. WCRT Analysis of CAN Messages in Gateway-Integrated In-Vehicle Networks. IEEE Transactions on Vehicular Technology, 2017, 66(11):9623-9637

[18] Zeng H, Natale M D, Giusto P, et al. Stochastic Analysis of CAN-Based Real-Time Automotive Systems. IEEE Transactions on Industrial Informatics, 2009, 5(4):388-401

[19] Gerónimo D, López A M, Sappa A D, et al. Survey of pedestrian detection for advanced driver assistance systems. IEEE Trans Pattern Anal Mach Intell, 2010, 32(7):1239-1258

[20] 谢勇. 新一代汽车电子系统的网络体系结构若干关键技术研究: [湖南大学博士学位论文]. 长沙: 湖南大学, 2013

[21] Zeller M, Prehofer C, Weiss G, et al. Towards Self-Adaptation in Real-Time, Networked Systems: Efficient Solving of System Constraints for Automotive Embedded Systems. 2011, 58(24):79-88

[22] Katoen J P, Noll T, Wu H, et al. Model-based energy optimization of automotive control systems. In: Proc of Design, Automation & Test in Europe Conference & Exhibition. IEEE, 2013:761-766

[23] Heinrich P, Prehofer C. Network-wide energy optimization for adaptive embedded systems. Acm Sigbed Review, 2013, 10(1):33-36

[24] Bernat G, Colin A, Petters S M. WCET analysis of probabilistic hard real-time systems. In: Proc of IEEE Real-Time Systems Symposium. IEEE Computer Society, 2002:279-288

[25] Navet N, Simonot-Lion F. The Automotive Embedded Systems Handbook. Automotive Embedded Systems Handbook, 2008, 53(10):751—763

[26] 唐小勇. 异构并行分布式系统可信调度理论与方法研究: [湖南大学博士学位论文]. 长沙: 湖南大学, 2013

[27] Wu M Y, Gajski D D. Hypertool: A Programming Aid for Message-Passing Systems. IEEE Transactions on Parallel & Distributed Systems, 1990,

1(3):330-343

[28] Kwok Y K, Ahmad I. Dynamic critical-path scheduling: An effective technique for allocating task graphs to multiprocessors. In: Proc of IEEE Trans Parallel Distributed Systems, 1996, 7(5):506–521

[29] G. C, Lee, E. A. A Compile-Time Scheduling Heuristic for Interconnection -Constrained Heterogeneous Processor Architectures. Parallel & Distributed Systems IEEE Transactions on, 1993, 4(2):175-187

[30] El-Rewini H, Lewis T G. Scheduling Parallel Program Tasks onto Arbitrary Target Machines. Journal of Parallel & Distributed Computing, 1998, 9(2):138-153

[31] Kruatrachue B, Lewis T. Grain Size Determination for Parallel Processing. IEEE Computer Society Press, 1988, 5(1):23–32

[32] Hwang J J, Chow Y C, Anger F D, et al. Scheduling Precedence Graphs in Systems with Interprocessor Communication times. Siam J Comp, 1989, 18(2):244-257

[33] Topcuoglu H, Hariri S, Wu M Y. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE transactions on parallel and distributed systems, 2002, 13(3):260–274

[34] Yang T, Gerasoulis A. DSC: scheduling parallel tasks on an unbounded number of processors. IEEE Transactions on Parallel & Distributed Systems, 1994, 5(9):951-967

[35] Kim S, Browne J. approach to mapping of parallel computation upon multiprocessor. 1988

[36] Liou J C, Palis M A. An efficient clustering heuristic for scheduling DAGs on multiprocessor. Proceedings of Symposium on Parallel & Distributed Processing, 1996

[37] Ahmad I, Kwok Y K. A New Approach to Scheduling Parallel Programs Using Task Duplication. In: Proc of International Conference on Parallel Processing. IEEE Computer Society, 1994:47-51

[38] Park G L, Shirazi B, Marquis J. DFRN: A New Approach for Duplication Based Scheduling for Distributed Memory Multiprocessor Systems. In: Proc of Parallel Processing Symposium, 1997. Proceedings. International. IEEE, 1997:157-166

[39] Chung Y C, Ranka S. Applications and performance analysis of a compile-time optimization approach for list scheduling algorithms on distributed memory multiprocessors. IEEE, 1995, 11:512-521

[40] Hou E S H, Ansari N, Ren H. Genetic algorithm for multiprocessor scheduling. IEEE Trans Parallel & Distributed Systems, 1994, 5(2):113-120

[41] Singh H, Youssef A. Mapping and scheduling heterogeneous task graphs using genetic algorithms. 5th IEEE Heterogeneous Computing Workshop (HCW '96, 1995

[42] Shroff P, Watson D W, Flann N F, et al. Genetic simulated annealing for scheduling data-dependent tasks in heterogeneous environments. In: Proc of Heterogeneous Computing Workshop. 1996

[43] Wang L, Siegel H J, Roychowdhury V P, et al. Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach. Journal of Parallel & Distributed Computing, 1997, 47(1):8-22

[44] Correa R C, Ferreira A, Rebreyend P. Integrating list heuristics into genetic algorithms for multiprocessor scheduling. In: Proc of IEEE Symposium on Parallel and Distributed Processing. IEEE Computer Society, 1996:462

[45] Girault A, Kalla H. A Novel Bicriteria Scheduling Heuristics Providing a Guaranteed Global System Failure Rate. IEEE Transactions on Dependable & Secure Computing, 2009, 6(4):241-254

[46] Xie G, Zeng G, Chen Y, et al. Minimizing Redundancy to Satisfy Reliability Requirement for a Parallel Application on Heterogeneous Service-oriented Systems. IEEE Transactions on Services Computing, 2017, PP(99):1-1

[47] Benoit A, Canon L C, Jeannot E, et al. Reliability of task graph schedules with transient and fail-stop failures: complexity and algorithms. Journal of Scheduling, 2012, 15(5):615-627

[48] Benoit A, Dufosse F, Girault A, et al. Reliability and Performance Optimization of Pipelined Real-Time Systems. In: Proc of International Conference on Parallel Processing. IEEE Computer Society, 2010:150-159

[49] Li Z, Mobin M, Keyser T. Multi-objective and multistage reliability growth planning in early product-development stage. IEEE Transactions on Reliability, 2016, 65(2):769–781

[50] Zhu D, Melhem R, Mosse D. The effects of energy management on reliability in real-time embedded systems. In: Proc of IEEE/ACM International Conference on Computer Aided Design. IEEE, 2004:35-40

[51] Doğan A, Özgüner F. Biobjective Scheduling Algorithms for Execution Time–Reliability Trade-off in Heterogeneous Computing Systems. The Computer Journal, 2005, 48(3):300–314

[52] Dongarra J J, Jeannot E, Saule E, et al. Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. In: Proc of Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures. ACM, 2007, 280–288

[53] Guo Y, Zhu D, Aydin H. Reliability-aware power management for parallel real-time applications with precedence constraints. In: Proc of Green Computing Conference and Workshops. IEEE, 2011:1-8

[54] Salehi M, Tavana M K, Rehman S, et al. DRVS: Power-efficient reliability management through Dynamic Redundancy and Voltage Scaling under variations. IEEE/ACM International Symposium on Low Power Electronics and Design. IEEE, 2015:225-230

[55] Salehi M, Ejlali A, Alhashimi B. Two-Phase Low-Energy N-Modular Redundancy for Hard Real-Time Multi-Core Systems. IEEE Trans. Parallel Distrib. Syst., vol. 27, no. 5, pp. 1497–1510, May 2016

[56] Zhao B, Aydin H, Zhu D. Shared recovery for energy efficiency and reliability enhancements in real-time applications with precedence constraints. ACM Transactions on Design Automation of Electronic Systems, 2013, 18(2):1-21

[57] 张剑贤, 周端, 杨银堂,等. 处理器可靠性约束的电压频率岛NoC能耗优化. 电子与信息学报, 2011, 33(9):2205-2211

[58] 张忆文, 王成. 可靠性感知周期任务能耗管理调度算法. 计算机科学与探索, 2017, 11(5):833-841

[59] Zhang L, Li K, Xu Y, et al. Maximizing reliability with energy conservation for parallel task scheduling in a heterogeneous cluster. Information Sciences An International Journal, 2015, 319(C):113-131

[60] Zhang L, Li K, Li K, et al. Joint optimization of energy efficiency and system reliability for precedence constrained tasks in heterogeneous systems. International Journal of Electrical Power & Energy Systems, 2016, 78:499-512

[61] Zhang L, Li K, Li K. Bi-objective Optimization Genetic Algorithm of the Energy Consumption and Reliability for Workflow Applications in Heterogeneous Computing Systems. 2015

[62] Xiao X, Xie G, Xu C, et al. Maximizing reliability of energy constrained parallel applications on heterogeneous distributed systems. Journal of Computational Science, 2017

[63] Xie G, Chen Y, Xiao X, et al. Energy-efficient Fault-tolerant Scheduling of Reliable Parallel Applications on Heterogeneous Distributed Embedded Systems.

IEEE Transactions on Sustainable Computing, 2017, PP(99):1-1

[64] 秦啸，庞丽萍. 分布式实时系统的容错调度算法. 计算机学报，2000, 23(10):1056-1063

[65] Qin X, Jiang H, Swanson D R. An efficient fault-tolerant scheduling algorithm for real-time tasks with precedence constraints in heterogeneous systems. In: Proc of International Conference on Parallel Processing. IEEE Computer Society, 2002:360

[66] Qin X, Jiang H. A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems. Parallel Computing, 2006, 32(5):331–356

[67] Zheng Q, Veeravalli B, Tham C K. On the Design of Fault-Tolerant Scheduling Strategies Using Primary-Backup Approach for Computational Grids with Low Replication Costs. IEEE Transactions on Computers, 2009, 58(3):380-393

[68] Mei J, Li K, Zhou X, et al. Fault-Tolerant Dynamic Rescheduling for Heterogeneous Computing Systems. Journal of Grid Computing, 2015, 13(4):1-19

[69] Benoit A, Hakem M, Robert Y. Fault tolerant scheduling of precedence task graphs on heterogeneous platforms. In: Proc of IEEE International Symposium on Parallel and Distributed Processing. IEEE, 2008:1-8

[70] Benoit A, Hakem M, Robert Y. Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems. Parallel Computing, 2009, 35(2):83–108

[71] Zhao L, Ren Y, Yang X, et al. Fault-tolerant scheduling with dynamic number of replicas in heterogeneous systems. In: Proc of IEEE International Conference on High Performance Computing and Communications. IEEE, 2011:434-441

[72] Zhao L, Ren Y, Sakurai K. Reliable workflow scheduling with less resource redundancy. Parallel Computing, 2013, 39(10):567–585

[73] Shatz S M, Wang J P. Models and algorithms for reliability-oriented task-allocation in redundant distributed-computer systems. IEEE Transactions on Reliability, 2002, 38(1):16-27

[74] Ovatman T, Brekling A W, Hansen M R. Cost Analysis for Embedded Systems: Experiments with Priced Timed Automata. Electronic Notes in Theoretical Computer Science, 2010, 238(6):81-95

[75] Qiu M, Sha H M. Cost minimization while satisfying hard/soft timing constraints for heterogeneous embedded systems. ACM Transactions on Design Automation

of Electronic Systems (TODAES), 2009, 14(2):25

[76] Gan J, Pop P, Madsen J. Tradeoff Analysis for Dependable Real-Time Embedded Systems during the Early Design Phases: [Dissertation]. Univ. of Technical University of Denmark Danmarks Tekniske Universitet, Department of Informatics and Mathematical ModelingInstitut for Informatik og Matematisk Modellering, 2014

致　谢

积一时之跬步，臻千里之遥程，三年攻读硕士学位的厚实积累，终于结束到新的起点。荏苒的学习韶光，过程艰辛却获益良多，每一步都离不开老师、亲人、同学和朋友所给予的谆谆教诲、无私帮助与殷切关爱。此刻，向他们表达我最诚挚的感谢。

衷心感谢我的导师李仁发教授。从引领我参观实验室的那刻起，李老师就以他渊博的理论学识和丰富的人生阅历影响着我的人生方向以及人格塑造。在之后选题、开题、论文撰写以及论文发表的每个研究阶段，李老师都紧密关注我的进展和问题。在我遇到困难时，李老师总是耐心解答我的疑惑，给予方向上和实操上的细心指导，并鼓励我以更高的要求朝更远的目标前行。在我迷茫与困惑时，李老师总是设身处地的为我考虑，让我以最好的状态走最适合自己的前行之路。李老师的学术造诣与人格涵养将是我一生学习和奋斗的榜样。

衷心感谢谢国琪副教授。本研究的工作也离不开谢老师的倾囊相授，选题时的细心指导、研究时的耐心解惑、撰写小论文和大论文时的严格把关，促成我顺利交上这张研究答卷。谢老师在我的研究领域有着深刻的见解和丰硕的成果，其精湛的专业知识以及刻苦钻研的精神是我研究生涯的丰厚收获。

衷心感谢嵌入式与网络计算重点实验室的李肯立老师、曾庆光老师、徐成老师、杨科华老师、李蕊老师、刘彦老师、付彬老师等在学习方面给予的指导与意见；感谢白洋师姐、黄一智师兄、吴武飞师兄、黄晶师兄、周佳师兄、宋金林、邓湘军、李婵娟、屠晓涵等同门的友爱与帮助，能在这个小家庭里共同成长让我倍感荣幸；感谢我的室友高慧、钟涛、胡小花，我的朋友伍凯、师哲、潘浩，三载来一路风雨一路温情的情谊让我倍感珍惜，愿彼此都有似锦前程。

特别感谢我的父母，给了我生命和无私的爱，含辛茹苦的养育我，毫无条件的支持我，让我在前行中有着坚强的后盾，让我懂得珍惜所爱并为之更努力的实现自己的追求。

最后，向所有百忙中参与论文评审和答辩的各位专家学者致以最诚挚的谢意，你们所提出的宝贵意见和建议是我继续不懈努力的动力。

袁娜

2018年5月5日于湖南大学

附录A　攻读硕士期间发表的学术论文

论文发表：

[1] Yuan N, Xie G, Li R, Chen X. An Effective Reliability Goal Assurance Method using Geometric Mean for Distributed Automotive Functions on Heterogeneous Architectures. In: Proc of IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA), 2017:667-674. (已收录，CCF C 类).

[2] Xie G, Li Z, Yuan N, Li R, et al.Toward Effective Reliability Requirement Assurance for Automotive Functional Safety. ACM Transactions on Design Automation of Electronic Systems (TODAES). (已接收，CCF B类).

附录B　攻读硕士学位期间所参与的项目

[1] 国家自然科学基金[61672217]：新一代汽车嵌入式系统功能安全的建模与算法研究.

[2] 国家自然科学基金[61702172]：基于AUTOSAR新平台标准的汽车CPS自适应安全调度.