

基于 VxWorks 的高可用容错系统的设计与实现

孙 锴, 慕德俊, 张慧翔

(西北工业大学 自动化学院, 陕西 西安 710072)

摘 要:文中设计了一种容错系统,该系统是建立在应用程序层之下、操作系统层之上位置结构的中间件。系统由三模冗余与其之间的通信链路组成系统的硬件结构以及由故障检测模块(对节点的检测和对应用程序的检测)、故障处理模块所组成的软件部分构成。基于 VxWorks 实时操作系统,设计了一种高可用的容错中间件系统,分析了系统的组成原理,给出了基于心跳检测的故障检测机制和 N 版本编程方法进行故障检测,以及前向和后向任务恢复方法进行故障恢复,并实现了原型系统。试验表明:给出的容错中间件系统具备了基本的容错能力,可有效提高系统的可用性和可靠性。

关键词:VxWorks;容错中间件;心跳检测;任务恢复

中图分类号:TP302.8

文献标识码:A

文章编号:1673-629X(2012)04-0123-03

Design and Implementation of High Available Fault-Tolerant System Based on VxWorks

SUN Kai, MU De-jun, ZHANG Hui-xiang

(School of Automation, Northwestern Polytechnical University, Xi'an 710072, China)

Abstract: The following system which is between the application layer and OS layer is a middleware. The three modules redundant structure and the communication bus compose the hardware shap; The software system contains fault-detection module and fault-handle module. The hardware shap and the software system form the whole fault-tolerant system. Based on VxWorks OS, firstly, a high available fault-tolerant middleware system is designed. Then principle of composition which belongs to the system is analyzed, and it presents a fault detection mechanism based on heartbeat detection and the N-VERSION programming detection method. As well as, the backward and forward task recover method is utilized. Finally the prototype system is implemented. Experiments demonstrate the fault-tolerant middleware is helpful for enhancing the software fault tolerance of system.

Key words: VxWorks; fault-tolerant middleware; heartbeat detection; task recover

0 引 言

运行于较高轨道的飞行器会受到来自地球辐射带、银河宇宙射线、太阳粒子事件等高强度的辐射环境。采用实时操作系统进行任务管理成为必要, VxWorks 系统以其良好的可靠性和卓越的实时性被广泛地应用在通信、军事、航空、航天等高精尖技术及实时性要求极高的领域中,如卫星通讯、军事演习、弹道制导、飞机导航等,由于受到成本、功耗、体积、重量等客观因素的限制,这将引起器件失效并最终导致系统可靠性下降甚至失效^[1]。为了保障飞行器能够可靠、稳定、高效地运行,在软硬件的设计当中应加入故障恢复机制。故障恢复通常采用容错技术,容错(Fault-Tol-

erance)是指系统在出现硬件故障或软件错误的情况下让其能继续正确执行指定任务,提供外界环境所需的服务。飞行器中的计算机系统和普通的计算机类同,分为软件系统和硬件系统。硬件系统一般都采用冗余设计,软件系统则采用微内核和可重构设计技术。与普通的计算机系统不同的是,飞行器计算机系统软件和应用^[2]不是独立的,二者合成一个软件核在硬件平台上运行。在一个嵌入式系统中设计及运行,文中的系统是实现在 VxWorks 操作系统与应用层之间的中间件,基于前向恢复技术和后向恢复技术即基于检查点(checkpoint)技术实现故障的恢复。

1 系统组成模型

整个系统在硬件上采用飞行器普遍适用的三模冗余设计,软件是在应用层 FT(Fault-Tolerant)中间件,算法的模拟和仿真拟在 WindRiver 公司的 VxWorks 软件环境下进行。整个系统将在 PowerPC 模块上实现。

收稿日期:2011-09-01;修回日期:2011-12-03

基金项目:国家自然科学基金(60803158);研究生创业种子基金(Z2011051)

作者简介:孙 锴(1985-),男,硕士研究生,研究方向为高可用嵌入式系统;慕德俊,教授,博导,研究方向为自动控制。

采用上位机—下位机模式,上位机为普通的 PC 机,负责下位机程序的下载、程序的调试、程序的监控、控制软件的开发;上位机与下位机通过 CAN 总线相连接,CAN 总线负责两方数据的传输;下位机则执行各种应用任务,并向上位机反馈信息。整个开发系统的结构^[3]如图 1 所示。

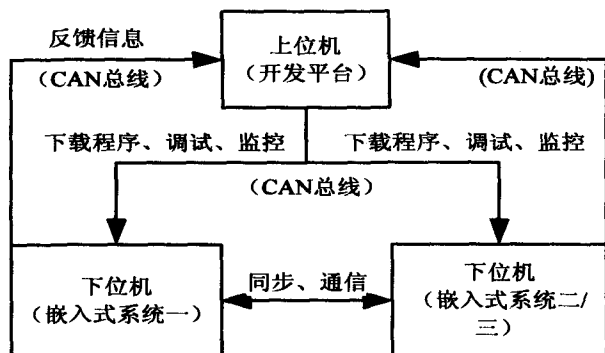


图 1 开发系统的结构图

2 容错中间件系统设计

系统由硬件和软件两大部分构成。硬件结构组成不是文中讨论的主要对象。文中的总体目标是在容错硬件和系统内核之上,在应用软件之上建立一个容错软件的模块。

2.1 软件环境及结构分析

基于 VxWorks 的软件环境:

VxWorks 操作系统是嵌入式开发环境的关键组成部分,它具有以下优点^[4]:

- 1) 实时性能:支持硬件实时和软实时(可抢占内核)。
- 2) 具有友好的开发调试环境,便于操作、配置和应用程序的开发调试。
- 3) 提供完整 BSP^[5] 的 TCP/IP 协议栈。
- 4) 支持文件系统^[6](tffs 等)。

容错层次体系与组件结构:

在以上三个处理器完全相同的节点所组成的系统中,从逻辑上划分为主、从节点,节点间形成相互备份关系。三节点通过完全相同的通信线路—CAN 总线相连接。PC 机负责对应用任务在系统中的安排、应用程序所需数据的分发、数据的交换等;节点运行应用程序负责特定的应用任务。系统^[7]主要由如图 2 的层次构成,在此主要考虑 FT 中间件与系统核,其它层不做分析。

容错系统主要包含:应用程序检测模块、节点检测模块(心跳模块)、故障处理模块。

主节点包括上层的应用控制程序、处于之下的心跳模块和故障处理模块;从节点一、二则有上层的应用程序、之下的故障处理模块与包含看门狗程序和 N 版

本编程方法在内的应用程序故障检测模块。主从节点间的故障恢复模块和心跳模块,分别通过传递故障信息、心跳信息来完成恢复和对节点的检测。

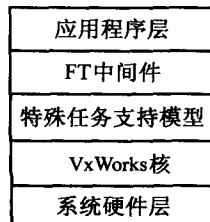


图 2 系统层次结构

故障检测:对故障的检测主要依靠应用故障检测模块与节点故障检测模块(心跳模块)实现。应用程序故障检测模块负责对应用程序的故障进行检测,采用的是 N 版本编程^[8]的技术,必要时模块亦可利用看门狗机制自检实现,对于节点级的软件故障检测则依靠心跳模块采用心跳机制实现。

故障处理:对应用软件和节点级软件的故障,统一可采用前向恢复和基于检查点(checkpoint)的后向恢复技术^[9]来实现对故障的恢复。

2.2 系统状态转换分析

鉴于对系统及各步骤更深刻的认识,定义了五种状态^[9],分别是:

- (1) 双工状态:两个处理机组均处于正常的工作状态;
- (2) 热备状态:指一个机组在系统中正常运行,而另一个机组正在加入系统;
- (3) 温备状态:指一个机组在系统中正常运行,而另一个机组已与系统在逻辑上断开了,只可接收输入,但不可输出;
- (4) 冷备状态:指一个机组在系统中正常运行,而另一个机组与系统在物理上脱离了;
- (5) 单工状态:指只有一个机组在系统中正常运行时系统的状态。温备、冷备状态均属于单工状态。

当双工状态的节点出现暂时错误或得到切出命令时,转换为温备状态并自检,出现永久故障则转换为冷备状态并维修;冷备状态的节点加电即转换为温备状态;自检中的温备状态节点正常时,向前恢复切入转换为热备状态等待与主节点同步,如故障则转换为冷备状态。

2.3 容错中间件实现

2.3.1 故障检测模块的实现

要设计一个高可用高可靠的系统,在太空环境中实际发生干扰的类型及发生的频率,用何种策略检测故障等这类问题是我们必须考虑的。对于应用程序故障检测,使用了 N 版本程序设计方法和看门狗机制对其进行实现,针对不同情况选择不同的策略。

软件看门狗机制^[10],是一个看门狗守护进程,周期性地发送值为零的信号给程序并且检查反馈值,从而检测出程序是否运行,持续地观察应用程序的生命周期,如无法回应则等待由应用程序设定的时间并且重复发送,再次失败表明进程阻塞或崩溃。如检测到程序阻塞或崩溃,看门狗将程序恢复到初始的内在状态,或故障前最后的检查点状态。

N 版本编程,这里提到的多版本,只是对关键软件进行多版本设计。其思想是用 *N* 个具有相同功能的程序同时(或先后)执行一项计算,结果通过多数表决来选择。

主从节点身份确认^[11],将根据如图 3 的流程对系统进行功能设计。

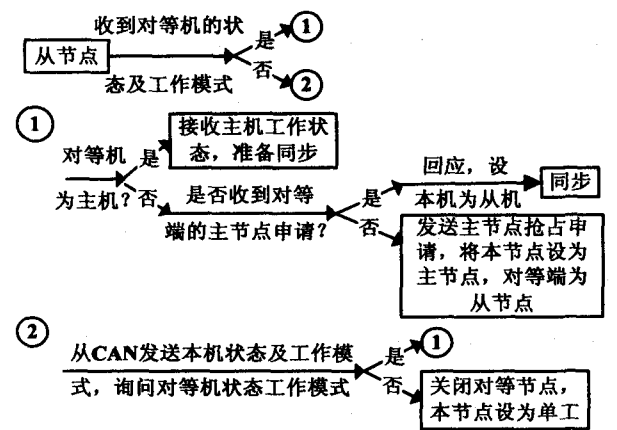


图 3 主从节点身份确认流程

对于节点故障的检测,系统正常启动运行后,两从节点定时向主节点容错模块发送心跳^[12]计数,主节点容错模块据此判断各节点的状态。当节点超时未发出心跳信息,则容错模块判定此节点失效。调用故障恢复模块对其进行恢复。

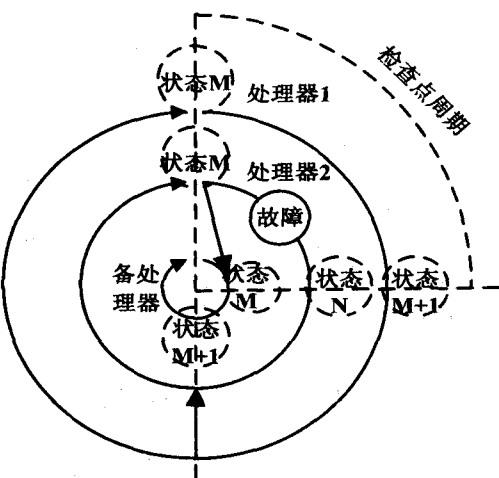
2.3.2 故障处理模块的实现

1)前向恢复。鉴于系统采用的是双模冗余结构,采用前向恢复技术,指系统从故障中恢复时,从出错时刻以后的某一时刻点开始恢复。当一个独立的瞬时故障发生时,采用拥有空闲处理器的前向恢复技术可避免卷回。

过程如图 4 所示,描述如下:两个独立的处理器同时处理一个任务,相互比较检查点。如果检查点不匹配则利用空闲处理器进行确认操作,在确认时,前两个处理器继续执行,同时第三个处理器回溯任务来决定故障处理器,最后将正确状态拷贝到故障处理器,完成恢复。

2)后向恢复。在发现错误后,从发现错误的检查点向后退回到最近的一个检查点,也就是退回到最近的一个无故障状态,重新执行那一段程序。它是一个逐级审计的过程。

具体实施:VxWorks 系统提供一组用户接口 API 函数库^[13],供用户在源程序中调用,以满足高可靠系统对软件容错的要求。能够满足容错系统对文件、串口 I/O、检查点、任务的管理。



M、N: 对应检查点周期, 处理器不同的状态;
→ : 各处理器状态的复制

图 4 前向恢复过程

3 试验验证

至此,基于 VxWorks 的容错系统原型基本实现。现将对原型进行试验测试,将使用三块完全相同的 PowerPc8313 开发板完成。利用板载 Flash 作为外部持久存储器保存检测的信息。测控主机与相联于定制的机箱内的三块开发板,通过 CAN 总线连接通信,利用 CAN 总线向开发板发送命令,根据总线中输出的数据判断测试结果正确性。首先启动测试用任务,运行一段时间后,测控计算机发送检查点设置命令,然后人为对开发板断电重启,模拟系统故障。系统重启后进行任务恢复,检测任务是否在检查点设置时刻继续执行。

表 1 原型系统的测试用例

测试用例	测试任务
单任务恢复	单个计算 π 任务
多个独立任务恢复	一个计算 π 任务和一个计算素数任务同时运行
多任务通信恢复	一个任务打开文件,利用消息队列发送文件;一个任务从消息队列接收数据,保存到新的文件
多任务协同工作恢复	一个任务用于读取密文数据,一个任务用于解密密文,另一个任务把明文数据写入文件,三个任务之间利用消息队列传递加解密数据

采用了四个测试用例,如表 1 所示。通过测试,任务能利用检查点文件进行恢复,继续执行。目前,基于检查点的任务恢复原型具有针对全局变量、动态分配数据、二进制信号量、消息队列、文件和串口的检查点设置和任务的恢复能力。

- Piscataway, NJ: IEEE, 2009: 1-4.
- [8] Xia Feng. Cyber-physical systems research group[EB/OL]. [2010-03-10]. <http://www.cpschina.org/>.
- [9] Puschner P, Burns A. Guest Editorial: A Review of Worst-Case Execution-Time Analysis[J]. Real-Time Systems, 2000, 18(2-3): 115-128.
- [10] Wegener J, Grochtmann M. Verifying Timing Constraints of Real-Time Systems by Means of Evolutionary Testing[J]. Real-Time Systems, 1998, 15(2): 275-298.
- [11] Tracey N, Clark J, Mander K. The Way Forward for Unifying Dynamic Test Case Generation: The Optimization-Based Approach[C]//Proc of International Workshop on Dependable Computing and Its Applications (IFIP'98). Johannesburg, South Africa: [s. n.], 1998: 12-14.
- [12] Petters S M. How Much Worst Case is Needed in WCET Estimation[C]//2nd International Workshop on Worst-Case Execution Time Analysis. Technical University of Vienna, Austria: [s. n.], 2002.
- [13] Bemat G, Colin A, Petters S M. WCET Analysis of Probabilistic Hard Real-Time Systems[C]//Proc of the 23rd Real-Time Systems Symposium. [s. l.]: [s. n.], 2002.
- [14] Kiltler R, Puschner P, Wenzel I. Measurement-Based Worst-Case Execution Time Analysis Using Automatic Test-Data Generation[C]//Proc of the 4th International Workshop on Worst-Case Execution Time (WCET) Analysis. Catania, Sicily, Italy: [s. n.], 2004.
- [15] Betts A, Bemat G. Issues Using the Nexus Interface for Measurement-Based WCET Analysis[C]//5th International Workshop on Worst-Case Execution Time Analysis. Palma de Mallorca, Spain: [s. n.], 2005.
- [16] Engblom J. Processor Pipelines and Static Worst-Case Execution Time Analysis[D]. Uppsala, Sweden: Acta Universitatis Upsalienensis, 2002.
- [17] Vivancos E, Healy C, Mueller F, et al. Parametric Tuning Analysis[C]//Workshop on Languages, Compilers, and Tools for Embedded Systems. Utah: ACM Press, 2001: 88-93.
- [18] Kirner R, Puschner P. Classification of WCET Analysis Techniques[C]//Proc of 8th IEEE International Symposium on Object-oriented Real-time Distributed Computing. Seattle, WA: [s. n.], 2005: 190-199.
- [19] Harel D, Naamad A. The STATEMATE Semantics of Statecharts[J]. ACM Transactions on Software Engineering and Methodology (TOSEM), 1996, 5(4): 293-333.
- [20] Shaw A C. Reasoning about time in higher level language software[J]. IEEE Transactions on Software Engineering, 1989, 15(7): 875-889.
- [21] Blieberger J. Daut-flow framework for worst-case execution time analysis[J]. Real-Time Systems, 2002, 22: 183-227.
- [22] van Engelen R A, Gallivath K, Walsh B. Parametric Timing Estimation with the Newton-Gregory Formulae[J]. Journal of Concurrency and Computation: Practice and Experience, 2004, 18(11): 1435-1463.

(上接第125页)

4 结束语

文中尝试在 VxWorks 操作系统环境下实现嵌入式故障恢复系统各模块原型。通过测试表明,该原型具有基本的故障恢复能力。对在 VxWorks 系统下,利用系统方法提高软件系统的容错能力具有一定的参考意义。进一步的工作主要是对检查点设置^[14]过程、故障更进一步的细分和针对各类型恢复策略进行优化,并完善原型系统的任务恢复能力。

参考文献:

- [1] 孙 栓,赵 敏,戴 维.微小卫星星载计算机存储容错技术研究[J].计算机技术与发展,2008,18(8):148-151.
- [2] 王 霆,长宁宁,王艳利.分布式高可靠性星载计算机系统研究与实现[J].科技信息,2008(10):57-58.
- [3] 历海燕,赵志国,李新明.分布式嵌入式环境下的容错软件设计[J].计算机应用研究,2006(7):142-143.
- [4] 罗国庆.嵌入式软件开发[M].北京:机械工业出版社,2003.
- [5] 黄启军,李晓光.基于 S3C44B0X 微处理器的 VxWorks BSP 开发与设计[J].计算机技术与发展,2006,16(7):45-47.
- [6] 周启平,张 杨,吴 琼.VxWorks 开发指南与 Tornado 实用手册[M].北京:中国电力出版社,2004:251-254.
- [7] 李小群,张文君,潘远明,等.基于 RTEMS 的软件容错系统设计[J].计算机应用研究,2009,26(3):911-913.
- [8] 张卫民.航天飞控软件的二维容错体系结构设计[J].计算机工程,2008,34(5):266-267.
- [9] 李宏亮,文 梅,张春元,等.高可用实时系统中故障检测及故障恢复技术的研究[J].计算机工程与科学,1999,21(6):81-84.
- [10] Shirvani P P, McCluskey E J. Fault-tolerant Systems in A Space Environment: The CRC ARGOS Project[R]. [s. l.]: [s. n.], 1998.
- [11] 苟冬荣,刘海清.双机容错计算机系统的设计与实现[J].计算机工程,2008,34(15):255-257.
- [12] Deconinck G. Software-implemented Fault Tolerance and Separate Recovery Strategies Enhance Maintainability[J]. IEEE transactions on reliability, 2002, 51(2): 158-165.
- [13] Huang Yennun, Kintala C. Software Fault Tolerance in the Application Layer[M]. [s. l.]: John Wiley & Sons Ltd., 1995: 237-247.
- [14] 鄢喜爱,杨金民,田 华.双机容错系统中最佳检查点间隔的分析[J].计算机工程,2007,33(5):283-284.