VxWorks 中任务恢复机制的设计与实现

胡延苏[1]，南秦博[2]，高　昂[1]，慕德俊[1]
(1.西北工业大学自动化学院,陕西西安,710072;2.空军工程大学训练部,陕西西安,710051)

摘要　软件容错技术是保证系统高可靠性及高可信性的有力工具。设计并实现了一种在 VxWorks 系统下基于检查点的任务恢复机制。通过对 VxWorks 系统下检查点文件内容的分析,采用 3 种方法来解决检查点的任务恢复问题:基于内存预先分配的主动内存管理,解决任务恢复时数据内存地址变化的问题;建立系统内核对象池,实现支持多任务之间同步和通信的内核对象的恢复;设计用户层任务恢复中间件,实现用户级检查点设置和任务恢复。最后设计基于 VxWorks 和 PowerPC 的计算平台原理样机,通过对单任务、多个单任务、多任务通讯、以及多任务协调工作 4 个测试用例的分析表明,所设计的基于检查点的任务恢复实现方案能正确保存任务关键信息及保证任务恢复的正确性和一致性。

关键词　VxWorks;软件容错;检查点;任务恢复
DOI　10.3969/j.issn.1009-3516.2013.05.012
中图分类号　TP311.52　　**文献标志码**　A　　**文章编号**　1009-3516(2013)05-0048-05

Design and Implementation of the Checkpointing and Task Recovery Mechanism Based on VxWorks

HU Yan-su[1],NAN Qin-bo[2],GAO Ang[1],MU De-jun[1]
(1.School of Automation, Northwestern Polytechnical University, Xi'an,710072,China;
2.Department of Training, Air Force Engineering University, Xi'an 710051,China)

Abstract:Software fault tolerant technique is a means used to guarantee the high reliability and high credibility in systems. This paper proposes a checkpointing-based task recovery mechanism on the VxWorks operating system. Three methods are adopted to solve the checkpointing-based task recovery problem through analyzing the contents of checkpoint files in VxWorks: the variation of memory address is solved based on the active management of pre-allocated memory; the system kernel object pools are built to realize the recovery of kernel object supporting the synchronization and communication between multi tasks; and the task recovery middleware in user layer is designed to achieve the checkpointing setup and task recovery. Finally, the prototype system based on VxWorks and PowerPC is designed and the results show that the use of the checkpointing-based task recovery mechanism can save the task key information correctly and guarantee the validity and consistency of the task recovery.

Key words:VxWorks; software fault tolerance; checkpoint; task recovery

收稿日期:2013-04-15
基金项目:国家自然科学基金资助项目(61203233)
作者简介:胡延苏(1985－),女,山东曹县人,博士(后),主要从事嵌入式开发、云计算资源调度等研究.
E-mail:huyansu@nwpu.edu.cn

VxWorks 是 Wind River 公司开发的一种嵌入式操作系统,它具有一个高性能的操作系统内核,主要特点有快速多任务切换、抢占式任务调度、任务间通讯手段多样化、任务切换时间短、中断时延小等。它以良好的可靠性和卓越的实时性被广泛应用在通信、军事、航空、航天等领域中[1-2]。

为保证系统的高可靠性及高可信性,除了采用硬件冗余容错处理外,往往还必须提供软件容错能力[3]。基于检查点的任务恢复机制是一种典型的软件容错技术[4]。它通过保存任务运行状态到持久存储器,当系统出现故障而导致某些任务终止时,它可利用备份的任务信息实现任务的恢复,从而能有效提高系统的可靠性。

目前,基于检查点的任务恢复机制的研究已有很多,先后有研究人员尝试在 Linux[5]、Unix[6]、Minix[7]、Solaris[8] 和 WinNT[9] 等操作系统中实现检查点机制,但在 VxWorks 环境下检查点机制的研究,就我们目前所知,还未有相关文献。因此,本文尝试在 VxWorks 操作系统环境下实现基于检查点的任务恢复机制。

1　VxWorks 系统检查点内容分析

在 VxWorks 系统中,基于检查点的任务恢复机制的设计需保证检查点文件具有一定的完备性。检查点内容一般包括:

1)TCB(Task Control Block):TCB 记录了任务运行的上下文,包括任务队列,任务当前内容,系统对象等。在 VxWorks 中,Wind- TCB 结构定义了 TCB 相关数据。

2)任务栈:VxWorks 中,每个任务都有独立的栈空间,栈用于任务的函数调用,分配局部变量和函数返回值等。

3)数据块:主要包括用户任务中使用的全局变量和动态分配的内存数据。

4)任务间通信的内核对象:VxWorks 支持多任务运行,提供了信号量、消息队列、共享内存和管道等手段来实现任务同步和通信。

5)活动文件:主要包括文件描述符,访问方式,文件大小,读写指针等相关信息。

6)活动外部 I/O:典型的外部 I/O 如串口、以太网络等。

2　基于检查点的任务恢复技术

基于检查点的任务恢复并不是简单的将需要的信息保存下来,在恢复时简单复制即可,需根据待恢复对象的不同对数据进行选择性的更新,存在较大的技术难度。为此,本文采用 3 种方法解决 VxWorks 系统下基于检查点的任务恢复问题。

2.1　基于内存预先分配的主动内存管理

采用基于内存预先分配的主动内存管理方法,解决任务恢复时,数据内存地址变化的问题。在 VxWorks 系统内核映像 bss 段[10] 开辟一块较大的内存,将任务恢复过程中内存地址可能变化的对象统一进行分配和释放。在设置检查点时,将主动内存管理中使用的数据空间复制到持久存储器(如板载 Flash),在任务恢复时再原样装载到系统内存。由于该块内存以全局数组的方式存在,其在内存中的地址是始终不变的[11]。这种内存预先分配的主动内存管理方法,主要负责 TCB、任务栈、全局变量和动态分配数据的内存管理。

2.1.1　TCB 的保存和恢复

在创建任务时,利用主动内存管理分配 680 byte,作为 TCB 块的存储区域。TCB 中存在一些动态变化的系统环境变量,在任务恢复时,需更新检查点文件中 TCB 的相关变量,保证任务运行环境的一致性。经过大量试验分析,表 1 总结了 TCB 数据结构中必须更新的变量。

表 1　任务恢复时,TCB 中必须更新的变量

Tab. 1　The updated variables in TCB when task recovers

WindTcb 结构中变量定义	含义
OBJ-CORE objCore	对象管理
Q-NODE qNode	就绪阻塞队列节点
Q-NODE tickNode	延迟队列节点
Int excCnt	异常计数
UINT status	任务状态
RTP-ID currentCtx	任务当前上下文
RTP-ID rptId	实时进程 ID
UINT safaCnt	安全删除计数器
Q-HEAD saftQHead	安全删除队列头
Structsel Context * SelectContext	任务 select 信息
void * pWdbInfo	Wdb 信息指针
void * pPwrMgmtPState	PWR 管理任务状态

2.1.2　任务栈的保存和恢复

在 VxWorks 中,任务栈空间包含执行栈和异常栈。在创建任务时,通过主动内存管理从预先分配的内存空间中分配任务的执行栈和异常栈。任务运行时分配的局部变量和函数调用返回值等信息始终保存在主动内存管理可控的内存区间,当任务在恢复时可直接复制检查点中的任务栈数据,而不用担心变量内存地址变化的问题。

2.1.3 全局变量的保存和恢复

为了实现全局变量的保存和恢复,提供用户接口 API 将全局变量统一管理,以链表的形式添加和删除全局变量信息节点。在设置检查点时,整个全局变量链表被完整复制到外部持久存储器,当任务恢复时再利用保存的信息恢复全局变量。整个全局变量链表存储空间的分配由主动内存管理完成。

2.1.4 动态分配数据的保存和恢复

采用 malloc 函数等动态分配的内存地址是由 VxWorks 控制的。在任务恢复时,重新分配的动态内存地址会发送变化。因此,动态分配数据也需采用主动内存管理。由于每次分配的内存大小不定,实际分配的内存增多 4 bytes,用于指明该块动态数据的大小。在设置检查点时,所有动态数据区被保存到外部持久存储器。当恢复任务时,重新获得原分配内存空间,并把检查点文件中数据存入相应的内存中。

2.2 基于系统内核对象池的任务间通信管理

在 VxWorks 系统中,提供了信号量、消息队列、共享内存等内核对象支持多任务之间同步和通信。这些内核对象由操作系统管理,在任务恢复时,难于处理。通过建立系统内核对象池,可实现这些内核对象的恢复。在用户任务运行前,系统预先建立一定数量的内核对象,当用户任务需使用内核对象时,直接向对象池申请。在设置检查点时,只需记录对象池中各个内核对象的使用状态信息。当任务恢复时,重建对象池,并重新定义各个对象的使用状态信息。可简便的实现系统内核对象的保存和恢复。本文主要分析了信号量和消息队列的恢复。

2.2.1 信号量的保存和恢复

在 VxWorks 系统中,提供了 3 种信号量:二进制信号量、计数信号量和互斥信号量。二进制信号量是 VxWorks 中最高效的,使用最广泛的信号量,主要用于同步互斥。以二进制信号量为例,说明信号量的保存和恢复。

当需要使用信号量时,任务从对象池中申请信号量,获得使用信号量的标识符。任务利用用户接口 API 来设置信号量状态。在设置检查点时,记录信号量对象池中被申请的信号量标识符、打开参数、信号量值(空或满),多个信号量以链接的形式进行管理,存储到外部持久存储器。任务恢复时,系统重建信号量对象池,根据保存的信号量信息,重新更新信号量状态,保证任务恢复后使用的信号量状态与设置检查点时刻一致。

2.2.2 消息队列的恢复

在 VxWorks 系统中,消息队列是任务间通信的常用机制。类似于信号量的保存和恢复,建立消息队列对象池供用户任务使用,任务通过消息队列标识符来引用消息队列,通过提供的用户接口 API 来使用消息队列。在设置检查点时,如果还有很多消息没有接收,那么在任务恢复时这些消息就会丢失。因此,在设置检查点时需选择恰当的时机,并保证任务之间检查点信息的一致性。

2.3 基于检查点的任务恢复中间件

由于 VxWorks 不是开源软件,获取其内核源代码,需支付昂贵的版权费。另外,VxWorks 具有实时内核结构,能满足强实时性要求,如果直接修改 VxWorks 内核,可能会影响系统的实时调度,也可能引入新的未知错。因此,采用在用户级实现检查点设置和任务恢复,并不直接修改 VxWorks 内核代码,而是提供一组用户接口 API 函数库,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。根据该函数库所处的系统层次,称之为用户层任务恢复中间件,见图 1。

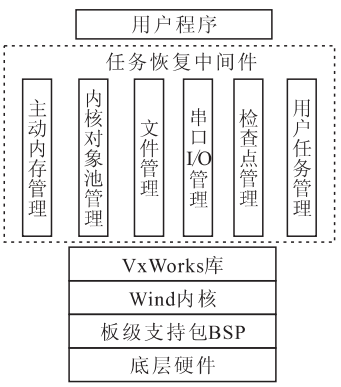

图 1 任务恢复中间件所处系统层次

Fig. 1 The system layer of task recovery middleware

从图 1 可见,任务恢复中间件在 VxWorks 库的基础上进行封装,为用户程序提供基于检查点的任务恢复能力。目前,任务恢复中间件包含 6 个模块:主动内存管理、系统内核对象池管理、文件管理、串口 I/O 管理、检查点管理、任务管理。前两个已经介绍,下面主要介绍后 4 个模块的实现原理。

1) 文件管理。通过提供文件管理用户接口 API,实现文件的检查点设置和恢复。任务通过文件标识符来引用文件对象,利用提供的文件管理 API 进行文件操作。文件管理 API 记录了使用的文件对象,包括文件名、打开参数,以及读写指针位置等,多个文件对象以链表形式保存。在设置检查点时,将文件对象链表保存到外部持久存储器。当任务恢复时,根据链表保存信息重新打开文件,并定位读写指针,恢复后的任务即可通过文件标识符继续进行文件操作。

2) 串口 I/O 管理。串口的检查点设置和恢复原理与文件管理类似。任务通过串口标识符进行串口通信。在检查点设置时,串口管理记录串口协议即可。当任务恢复时,根据串口协议重新打开串口即可。串口管理存在数据一致性问题,当本地系统重启时,如果外部计算机不知道本地系统已经出错,而继续发送串口数据,这些数据将会丢失。这一问题需通过多系统同步来解决[10]。

3) 检查点管理。检查点管理主要负责检查点文件的管理。任务恢复中间件建立了一个后台检查点设置任务。该任务维护了当前系统运行用户任务的信息链表,根据用户任务的运行情况选择检查点的设置时机。检查点管理模块也提供了用户接口 API,由用户程序发起检查点的设置请求。检查点设置主要解决 2 个问题:选择合适的时机和压缩检查点文件的大小。选择合适的时机以保证任务恢复前后数据的一致性;压缩检查点文件的大小以减少检查点设置的时间,提高检查点设置的效率。目前,采用定期设置检查点文件的方式,只复制被利用的主动内存区域,并对检查点文件进行压缩处理。

4) 任务管理。通过提供用户接口 API 实现任务的管理,包括任务的创建和删除。任务的创建利用了 taskLib 库中的 taskInitExcStk()函数。该函数可自定义任务 TCB、任务执行栈和异常栈在内存中的位置,可用于任务的恢复重建。任务创建以后,相关信息保存到任务的信息链表,由后台检查点设置任务维护。

任务终止有 2 种方式:一是用户调用 taskDelete()或 taskDeleteForce()删除任务;二是任务运行结束,VxWorks 系统调用 exit()函数隐性的删除任务。利用 taskHookLib 库中的 deleteHook()钩子函数时,不管哪种方式删除任务,任务都会先调用 deleteHook()函数,再调用任务删除函数。在 deleteHook()钩子函数中完成与该任务相关的清理工作。

3　VxWorks 下任务恢复机制验证

目前,VxWorks 下基于检查点的任务恢复原型系统基本实现。利用一块 PowerPC 开发板进行了验证实验。利用板载 Flash 作为外部持久存储器保存检查点信息。测控主机串口与开发板相连,利用串口向开发板发送命令,根据串口中输出的数据判断测试结果的正确性。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,检查任务是否在检查点设置时刻继续执行。测试环境与配置如下:

主机环境:

操作系统:Windows Xp;应用软件:Wind River Workbench 3.0

开发板环境:

开发板:MPC8313 EVB v11;操作系统:VxWorks 6.1

DDR SDRAM:256M;BOOT FLASH:512 KB

Main FLASH:16 MB

共采用了 4 个测试用例,见表 2。通过测试,任务能利用检查点文件进行恢复,继续执行。目前,VxWorks 提供的任务间通信机制有共享内存、信号量、消息队列、管道、信号和 Socket。在上面的机制中只实现了信号量和消息队列的恢复,共享内存的恢复可以借助信号量的恢复来实现,所以对使用管道、信号和 Socket 的任务还不能恢复。若要在任务中使用这些机制,还得做相应的机制恢复工作,这是一个缺陷。同时,在做检查点设置时,需要挂起任务,并对多个文件进行写操作,这样对任务的运行有一定的延迟,这方面还需改进。

表 2　原型系统的测试用例

Tab. 2　The test cases of the prototype

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,3 个任务之间利用消息队列传递加解密数据。

4　结语

本文尝试在 VxWorks 操作系统环境下实现基于检查点的任务恢复原型系统。通过测试表明,该原型系统具有基本的任务恢复能力。对在 VxWorks 系统下,利用基于检查点的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过程进行优化,并完善原型系统的任务恢复能力。

参考文献(References):

[1]　李祁,郭天杰,邢翠芳. 基于 VxWorks 的 1553B 总线

软件设计及实现[J]. 火力与指挥控制,2010,35(5):177-179.
LI Qi,GUO Tianjie, XING Cuifang. Design and realization of software for 1553B bus based on VxWorks [J]. Fire control & command control, 2010, 35(5): 177-179. (in Chinese)

[2] Zhang Chengye,Deng Shenglan,Ning Hong. A local checkpoint mechanism for on-board computing[C]// 2012 international conference on information science & technology. [S. l.]:IEEE press,2012: 520-526.

[3] De Assis,F H,Takase F K,Maruyama N,et al. Developing an ROV software control architecture: a formal specification approach[C]//38th annual conference on IEEE industrial electronics society. [S. l.]:IEEE press,2012: 3107-3112.

[4] 陆阳,王强,张本宏,等. 计算机系统容错技术研究[J]. 计算机工程,2010,36(13):230-235.
LU Yang, WANG Qiang, ZHANG Benhong, et al. Research on fault-tolerant technology for computer system [J]. Computer engineering, 2010, 36(13): 230-235. (in Chinese)

[5] 杨晖,陈闳中. 支持文件迁移的 Linux 检查点机制的实现[J]. 计算机工程,2010,36(3):266-268.
YANG Hui, CHEN Hongzhong. Implementation of Linux checkpoint mechanism supporting file migration [J]. Computer engineering, 2010, 36(3): 266-268. (in Chinese)

[6] 王春露,汪东升. Unix 进程检查点设置关键技术[J]. 计算机工程与应用,2002,38(1):90-93.
WANG Chunlu, WANG Dongsheng. Key techniques for Unix process checkpointing [J]. Computer engineering and applications, 2002, 38(1): 90-93. (in Chinese)

[7] 李毅,周明天. Minix 进程检查点机制的实现[J]. 计算机应用,2003,23(1):13-17.
LI Yi, ZHOU Mingtian. Process checkpoint implementation based on Minix[J]. Computer application, 2003,23(1):13-17. (in Chinese)

[8] 张悠慧,汪东升,郑纬民. Solaris 系统多线程检查点设置与卷回恢复[J]. 计算机工程与应用,2000,36(8):45-47.
ZHANG Youhui, WANG Dongsheng, ZHEN Weimin. A transparent checkpointing and rollback recovery for multithreaded programs under Solaris [J]. Computer engineering and applications, 2000, 36(8): 45-47. (in Chinese)

[9] 张悠慧,汪东升,郑纬民. Windows NT 环境下的进程检查点设置与回卷恢复[J]. 计算机研究与发展,2001,38(1):50-55.
ZHANG Youhui, WANG Donghsheng, ZHENG Weimin. Checkpointing and rollback recovery for Windows NT applications [J]. Journal of computer research and development, 2001, 38(1): 50-55. (in Chinese)

[10] 张杰智,任国林. 一种基于信道不可靠环境的协调式检查点协议[J]. 计算机技术与发展,2008,18(2):55-58.
ZHANG Zhijie, REN Guolin. A coordinated checkpointing protocol based on unreliable channels [J]. Computer technology and development, 2008, 18(2): 55-58. (in Chinese)

[11] Wind river systems, Inc. VxWorks kernel programmer's guide 6.1[S],2005.

(编辑:徐楠楠)

(上接第 4 页)

[3] 尤晋闽,陈天宁. 结合面静态接触参数的统计模型研究[J]. 振动与冲击,2010,29(11):47-50.
YOU Jinmin, CHEN Tianning. Statistical model for static contact parameters of joint surfaces[J]. Journal of vibration and shock, 2010, 29(11): 47-50. (in Chinese)

[4] ZapicoValle. A new method for finite element model updating in structural dynamics[J]. Mechanical systems and signal processing, 2010, 24: 2137-2159.

[5] 李德益,孟海军,史雪梅. 隶属云和隶属云发生器[J]. 计算机研究与发展,1995,32(6):15-20.
LI Deyi, MENG Haijun, SHI Xuemei. Membership clouds and membership cloud generators[J]. Journal of Computer Research and Development, 1995, 32(6): 15-20. (in Chinese)

[6] 李德益,刘常昱. 论正态云模型的普适性[J]. 中国工程科学,2004,6(8):28-34.
LI Deyi, LIU Changyu. Study on the universality of the normal cloud model[J]. China Engineering Science, 2004, 6(8): 28-34. (in Chinese)

[7] Kennedy J, Eberhart R C. Particle swarm optimization [C]//IEEE international conference on neural networks. Perth Piscataway: IEEE service center, 1995: 1942-1948.

[8] Shi Y, Eberhart R C. Fuzzy adaptive particle swarm optimization[C]//Proc congress on evolutionary computation. Seoul, Korea: 2001: 1103-1108.

[9] 黄仁全,靳聪,贺筱军,等. 自适应局部增强微分进化改进算法[J]. 空军工程大学学报:自然科学版,2011,12(3):84-89.
HUANG Renquan, JIN Cong, HE Xiaojun. A modified differential evolution algorithm with adaptive and local enhanced operator[J]. Journal of air force engineering university: natural science edition, 2011, 12(3): 84-89. (in Chinese)

(编辑:徐敏)