第34卷 第17期 计 算 机 工 程 2008年9月
Vol.34 No.17 Computer Engineering September 2008
·开发研究与设计技术· 文章编号：1000—3428(2008)17—0281—03 文献标识码：A 中图分类号：TP302.8

基于μC/OS-II的多MCU容错设计与应用

许 强，徐 凯
(重庆交通大学计算机与信息学院，重庆 400074)

摘 要：在分析μC/OS-II系统的基础上增加多MCU容错实时操作功能，可以对当前任务运行状态实时进行一致性操作，使多MCU中的一个节点在出现死节而崩溃时，其他节点可以实现死节的决策，并恢复其坏死节点的任务状态使其继续运行。通过三MCU方式结构及运行验证，改进的系统表现出稳定可靠的容错能力。

关键词：容错；μC/OS-II系统；多MCU系统；通信协议

Design and Application of Multi-MCU Fault-tolerance Based on μC/OS-II

XU Qiang, XU Kai
(College of Computer & Information, Chongqing Jiaotong University, Chongqing 400074)

【Abstract】 The paper analyzes μC/OS-II system, and adds some fault-tolerant operating modules based on multi-MCU modules. The modules achieve the coherent real-time operation for the current state of the running tasks. If one of the multi-MCU modules breakdowns, other MCU modules can judge the fault and resume bad system state to run on decision-making. The fault-tolerant real-time operating system, which shows a stable dependable fault-tolerant ability, has been tested on three MCUs.

【Key words】 fault-tolerance; μC/OS-II system; mutil-MCU system; communication protocol

1 概述

在容灾容错系统中，采用数据远程镜像结构[1]只能做时间不可确定性的恢复，出错系统本身不能由其他系统实时在线代替。在高可靠性要求的嵌入式应用系统设计中，更需要体现系统本身的容灾容错性，含嵌入式实时操作系统的单一MCU嵌入式系统技术已非常成熟，但其片内的Watch Dog仅能将程序的非正常运行重新复位从启动开始运行，仅掉电保护可以从断点处恢复，对于单MCU嵌入式系统的整体崩溃，不可能实现在线恢复运行。为解决整体崩溃或故障的实时容灾容错问题，本文采用多MCU的容灾容错结构，其关键是对单时实操作系统的改造，构造一致同步跟踪功能。由于是对μC/OS-II实时操作系统进行改造，完成同步决策，并实施恢复运行，因此其操作过程一方面在多实时操作系统之间建立消息跟踪模块，决策节点是否崩溃，另一方面实时同步保存与任务运行状态相对应的镜像任务，并通过镜像任务的激活对崩溃节点的运行状态进行有效恢复。

本文采用μC/OS-II实时操作系统进行改造是因为它是一个完整的、可移植、可固化、可剪裁的抢先式实时多任务内核。其原代码用ANSI C编写，占存储空间小，可适应8 bit~64 bit处理机的管理操作，其稳定性与可靠性已获得RTCADO-178B标准认证。

2 多MCU容灾容错结构

多MCU容灾容错可用如下几种方式实现容错功能：

(1)基于多嵌套系统的同工操作，即几套单嵌入式系统的结构、执行程序及节拍相同，对执行结果进行表决，若有一个节点出错，根据少数服从多数投票原则，选择正确结果[2]，如图1所示。

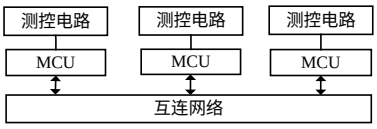

图1 同构多机冗余模型

(2)基于多嵌套系统的异步数据的实时保护操作。这一方法仅对运行状态及数据进行保护，原系统的再运行必须等到原系统恢复正常运行后，才能恢复程序断点处执行，并复原相关数据，如图2所示。

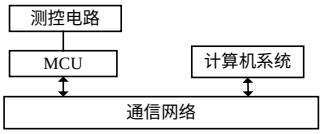

图2 异步数据实时保护模型

(3)基于多嵌套系统的异步实时操作系统的任务级在线替换无缝恢复，如图3所示。

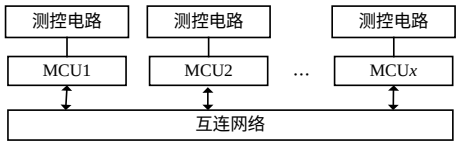

图3 测控电路冗余模型

基金项目：国家部委预研基金资助项目
作者简介：许 强(1964－)，男，高级工程师、硕士，主研方向：计算机测量与控制，多智能体系结构；徐 凯，高级工程师
收稿日期：2008-04-10 **E-mail：**xuqiang11821@163.com

从上述几种容错结构来看，第(1)种是传统的硬件及软件都同构的冗余结构，其利用率和效率很低；第(2)种仅对数据本身有一定的实时保护，不能在线恢复，这对控制实时不间断的系统显然不合适；第(3)种就是本文提出的方法，它不同于多任务及运行状态的单纯冗余备份[3]，是物理冗余但运行不冗余的容错系统，比如有多冗余单嵌入系统，每个单嵌入系统运行的任务可以不相同，以实现各自的功能，因此，效率大大提高，并且如果其中几个发生故障或崩溃，其他正常运行的单嵌入系统可实时替换恢复运行。当MCU同构时，可做到断点与应用数据的容错无缝运行，当MCU异构时，可做到操作状态与应用数据的容错运行。

3 实时容错操作系统结构设计

3.1 实时容错操作系统结构

由于采用了操作系统级的容灾容错，所恢复的级别为任务级，因此必须完成如下步骤：

(1)实时抽取有效恢复任务级的数据信息，如仅本机调用的任务控制块、事件控制块及任务堆栈数据，并对该数据信息压包，按广播方式传送到其他 MCU 系统上。

(2)接收消息包，解包并绑定修改服务程序进行实时修改，修改的内容主要是本机的镜像任务控制表、事件控制表及任务堆栈。

(3)对接收消息包定时监控，若超过一定时限，由多机联合决策，判断是否出错或崩溃。

(4)若出错或崩溃，则由其他多机根据负载平衡量竞争获得替换权。若仅为出错，则替换恢复，待出错系统恢复正常后，接收到消息，并确认，再抽取替换机器的实时信息复制到出错机器上恢复运行；若为崩溃，则替换恢复，并永久替换运行。

系统结构如图 4 所示。

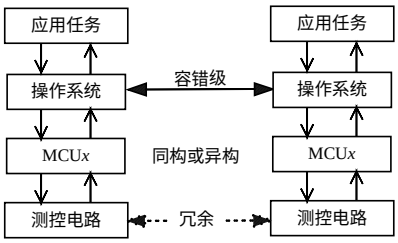

图 4 实时操作系统容错层次

3.2 μC/OS-II 操作系统多 MCU 容错操作设计

(1)系统初始化、多机冗余分配及调度

由于μC/OS-II现支持 64 个任务，除了系统可套用的 8 个任务外，最多可支持 56 个用户任务[4]，在多任务多机执行时要提前对多任务静态功能分类，并按功能分类分配到不同的机器，并在每个OSInit()程序中加入建立分配表部分，为了确保运行的独立性，不同类的任务在分配时无相关性。

在装载到不同 MCU 中的文件 OS_CFG.H 内，添加全局变量 INT8U OSMCUID，并命名本机的编号值。同时将任务创建函数变为：OSTaskCreate(void(*task)(void *pd), void *pdata,OS_STK *ptos,INT8U prio,INT8U McuID)或 OSTaskCreateExt(…, INT8U McuID)，参数项 INT8U McuID 为机器编号，实际上所有任务在每个机器都被创建，仅将静态分配了本机编号的任务作为可调度的激活任务，而其他任务可标识其他机器编号。在创建任务中调用 OS_TCBInit(…, INT8U McuID)，其中，INT8U McuID 为添加 MCU 编号参数，表明本任务所属 MCU 属性，并将函数对就绪表操作语句修改为

```
if (ptcb->MCUID=OSMCUID){
OSRdyGrp |=ptcb->OSTCBBitY
OSRdyTbl[ptcb->OSTCBY]= ptcb->OSTCBBitY}
else{
OSRdyGrpM |=ptcb->OSTCBBitY
OSRdyTblM[ptcb->OSTCBY]= ptcb->OSTCBBitY}
```

OSRdyGrpM 和 OSRdyTblM[]控制冗余任务表格，它的变化受其他 MCU 发来的运行任务状态消息的影响。

初始化创建 OS_EVENT 事件，其创建过程必须加入查找编码，因为对于不同机器的相同事件，事件指针并不同，所以必须加入 OSEVENTList 链接指针，其结构 EventList 为

```
typedef struct{
    Void *    OSEVENTNextPtr; //指针
    INT16U    OSEVENTID; //状态数据缓冲区
    OS_EVENT * pevent; //事件指针
    }EventList
```

目的是在修改镜像事件时，通过查找事件编码方便地找到事件指针，并修改镜像事件控制块的任务等待表。因此，在创建事件时必须加入事件编码参数，如将信号事件创建函数改为 OSSemCreate(INT16U cnt,INT16U EVENTCardID)，并在函数内向 OSEVENTList 链表添加查询事件块。

添加屏蔽表 OSMaskTbl[]，如同就绪表，创建任务时置位，1 为本地任务，0 为非本机任务。并初始化广播到不同的机器，作用于镜像任务与本地任务合并操作，实现容错功能。

(2)通信功能的设计

由于μC/OS-II 调度是时间间隔中断抢占式实时操作，因此定时调用 TickTask()，并调用 OSSched()进行任务调度。在调度函数 OSSched()中添加对就绪表、事件控制块和任务堆栈数据进行一致性镜像映射，其映射函数为 OSMirrorCopy(INT8U Mcommand)，其中，INT8U Mcommand 是对状态的分类，表示当前操作系统运行的状态，如等待状态、就绪状态或睡眠状态消息。不同类别在镜像映射操作时绑定的服务对象是不一样的，如运行态变为就绪状态，就传送变化的就绪表及任务的堆栈数据；运行态变为等待状态，则传送等待消息的任务控制表及任务堆栈数据。

定时将当前运行任务及转化的状态数据加入发送缓冲池，并按机器号广播发送，发送数据模块结构 MCOMSMessage 为

```
typedef struct{
INT8U       OSCommand; //消息类型
Void *      OSstateDat; //状态数据缓冲区
INT32U      OSstateDatSize; //缓冲区大小
Void *      OSTaskDat; //未传送的应用数据
INT32U    OSTaskDatSize; //未传送的应用数据大小
}MCOMMessage
```

当通信接口接到消息数据时，系统产生中断服务，并执行服务程序 OSMirrorRecept(INT8U Mcommand)处理，其中，INT8U Mcommand 对状态进行分类处理、修改冗余任务的属性及状态保存表，例如 OSRdyGrpM、OSRdyTblM[]及任务堆栈信息。

(3)灾错判断及容灾容错功能设计

当一个机器崩溃，其应答消息将不能到达其他机器，其他机器的每个个体为接收到的消息辨识机器编号，并构成可通信集合，设 x 消息的发送为崩溃机器，原集合$\{x,y,z,\cdots\}$将

分裂为{x},{y,z,…}，并且{y,z,…}在每一个未崩溃机器中都有一份，通过全集{x,y,z,…}-{y,z,…}={x}判断发 x 消息的机器崩溃或故障，并由发{y,z,…}的机器进行替换竞争，竞争原则是根据负载平衡 CPU 利用率的统计值 OSCPUsage 进行判断，假如发 y 消息机的负载最轻，则由 y 机替换，否则由其他负载最轻的机器替换恢复。替换恢复时，将被恢复任务的控制标志改为本地机编号，完成恢复替换运行。

当前级测控故障可通过本机报出故障，发送消息请求替换恢复。

当并非硬件崩溃而仅是软件程序出错时，待系统恢复正常运行后，可将替换机器运行的追加任务重新返回到原机器中运行。

复制内容包括镜像任务的任务堆栈、事件控制块、任务控制块、任务处理应用更新信息等反映当前状态和应用数据的信息。

修改就绪表时，将镜像任务保存的 OSRdyGrpM 和 OSRdyTblM[]与屏蔽表 OSMaskTbl[]相与，其结果与就绪表 OSRdyTbl[]相或得到新的就绪调度表 OSRdyTbl[]，即 OSRdyTbl[]=OSRdyTblM[] & OSMaskTbl[]。

修改事件控制表时，查找 OSEVENTList 的链表，获得相同编码的事件指针，从而准确地修改事件等待表。若镜像任务与本地任务合并，则事件控制自然进行操作进程。

4 实例设计及系统可靠性评估

实际设计时考虑三 MCU 体系结构，如图 5 所示，因为三 MCU 可充分支持表决功能。其无缝恢复的效率及可靠性取决于通信总线的效率和可靠性，通信总线采用片上高可靠性的双 SPI，实现三 MCU 全互连结构。当有意损坏一个 MCU 节点系统时，其表决功能起作用，由另一节点抢占替代其运行功能，并继续无缝运行；当有意让程序出错，即不发消息时，同样由镜像任务替代运行；当收到系统恢复消息，则在下一个通信节拍后无缝恢复运行。

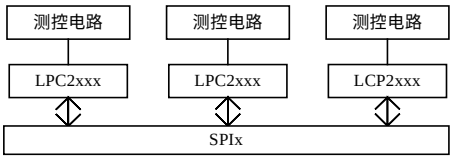

图 5　三 MCU 结构

本系统已在某火箭卫星发射基地得到应用，在恶劣环境(如强电磁及机器碰撞损毁)下运行的有效性已得到验证。

5 结束语

本文从独立系统本身的容灾容错出发，建立多机同步决策，实施节点故障及崩溃的在线补救与恢复，并对其作出容错改造。设计的三 MCU 嵌入式系统在检测控制上已得到可靠性运用，由于运行系统是各自独立的系统，因此，利用效率比单纯的重复高得多，其容错容灾思路对其他复杂恶劣环境下的系统设计具有借鉴作用。

参考文献

[1] Ji Minwei, Veitch V, Wilkes J. Seneca: Remote Mirroring Done Write[C]//Proceedings of the 2003 USENIX Annual Technical Conference. San Antonio, USA: [s. n.], 2003: 253-268.

[2] 胡昌华, 许华龙. 控制系统故障诊断与容错分析和设计[M]. 北京: 国防工业出版社, 2000.

[3] Krishna C M, Shin K G. On Scheduling Tasks with a Quick Recovery from Failure[J]. IEEE Trans. on Computer, 1986, 35(5): 448-454.

[4] Labrosse J J. 嵌入式实时操作系统 uC/OS-II[M]. 2 版. 邵贝贝, 译. 北京: 北京航空航天大学出版社, 2003.

（上接第 280 页）

(2)客户端缓存

客户端的缓存机制是将浏览过的幅面缓存在客户端，再次拖动回该区域时，就像在本地操作一样，无需跟服务器打交道。当用户缩放地图时，就清空客户端缓存。

客户端缓存技术在各类应用条件下都可以使用，对于频繁往回拖动的情况有很好的性能体验。但是，当需要填充拖出的空白区域时，往往要从服务器端下载 2 幅~3 幅图片来填充。比如往右下拖动时，需要下载原来图片的左、左上、上方 3 幅图片来填充空白，增加了用户等待时间。另外，随着缓存数据的增加，客户端内存的占用也十分明显，对配置不高的客户端机器造成了负担。

4 结束语

在开发Web GIS软件项目的过程中，制约Web GIS软件发展的一个重要问题就是空间操作的响应速度过慢。本文利用ESRI公司的ArcServer，测试了影响Web GIS空间操作性能的各项主要因素及各项优化方案对系统性能带来的影响，对开发同类型的地理信息系统的可用性有重要的指导作用。为了更高程度地提升Web GIS系统的响应速度水平，下一步将从系统框架层面，用面向Agent的框架结构[5]并加入智能预测模型来改进Web GIS的性能。

参考文献

[1] 赵村民, 宋利好, 赵晓民. 基于 Oracle 与 ArcSDE 的空间信息访问优化[J]. 吉林大学学报, 2004, 22(3): 283-288.

[2] Baptista S, Nunes C, Sousa C P. On Performance Evaluation of Web GIS Applications[C]//Proc. of the 16th International Workshop on Database and Expert Systems Applications. Copenhagen, Denmark: IEEE Computer Society, 2005.

[3] 吴华意, 章汉武. 地理信息服务质量(QoGIS): 概念和研究框架[J]. 武汉大学学报: 信息科学版, 2007, 32(5): 385-388.

[4] Kim S H, Yoo C Y, Lee Y G, et, al. A Jigsaw Query Processing Technique for Web GIS[C]//Proceedings of the IEEE International Conference on Information Technology: Coding and Computing. Las Vegas, NV, USA: IEEE Computer Society, 2000.

[5] 罗英伟, 汪小林, 许卓群. 分布式 GIS 的多 Agent 系统建模与实现[J]. 计算机辅助设计与图形学学报, 2004, 16(12): 1730-1737.