第33卷 第17期 计 算 机 工 程 2007年9月
Vol.33 No.17 Computer Engineering September 2007
·软件技术与数据库· 文章编号：1000—3428(2007)17—0071—03 文献标识码：A 中图分类号：TP316

嵌入式操作系统RTEMS基于多处理器的优化

宋 伟，杨学军
(国防科技大学计算机学院，长沙 410073)

摘 要：随着多处理器嵌入式系统应用的普及，通用嵌入式操作系统开始引入对并行化的支持，作为优秀的开源嵌入式操作系统 RTEMS亦是如此。但是RTEMS对多处理器的支持机制上存在可能会使整个多机系统崩溃的死锁问题，此外在数据处理系统中做数据分发时对硬件性能损失十分严重。该文结合某型号星载计算机的研发工作，对RTEMS的多处理器支持机制加以完善，并对RTEMS存在的数据分发时的性能瓶颈给出一种优化方案。

关键词：多处理器；嵌入式；RTEMS

Optimization of Embedded Operating System RTEMS Based on Multiprocessor

SONG Wei, YANG Xue-jun
(Computer Institute, National University of Defence and Technology, Changsha 410073)

【Abstract】With the popularity of multiprocessor embedded system, like many other embedded operating systems, RTEMS, as a great open source embedded operating system, also supports parallelization. However, RTEMS's multiprocessor support may cause deadlock on multiprocessor system. And it may cause severe performance damage to the hardware when distributing data in data processing system. Based on an onboard computer research, this article improves RTEMS's multiprocessor support and will work out an optimization for RTEMS's performance bottleneck that may occur in data distribution.

【Key words】multiprocessor; embedded; RTEMS

1 概述

随着嵌入式系统应用领域的不断拓展，许多应用对嵌入式系统计算能力和实时性都有着较高的要求，致使普通的单处理器系统已经无法满足需求。另一方面，随着处理芯片的廉价化，采用多处理器协同工作来提高嵌入式系统的计算能力，降低响应时间已成为一种必然趋势。嵌入式系统的并行化直接要求嵌入式操作系统对多处理器的支持，因而对嵌入式操作系统并行化支持的研究是有意义的，尤其是对优秀的通用嵌入式操作系统的多处理器支持的完善和优化更是具有实用价值。

RTEMS(real-time executive for multiprocessor systems)实时操作系统最初是美国军方为了实时导弹系统而开发的。RTEMS从1988年开始开发，并于1999年开始对外开放源码，并由OAR公司进行维护和升级。经过近20年的发展，RTEMS已成为一种可以与诸如VxWorks等优秀的商业实时操作系统相媲美的开源嵌入式实时操作系统。如今RTEMS具有支持大多数处理器芯片；支持多任务；支持同构或异构多处理器系统；支持事件驱动、基于优先级的实时调度算法；具有可选的实时调度算法；支持任务间的通信和同步；支持Priority Inheritance 算法；快速响应的中断管理；支持动态存储器分配等优点。

RTEMS 本身已经是一种比较成熟的嵌入式实时操作系统，而且有着比较广泛的应用，对其进行多处理器支持上的完善和优化具有比较大的实用性。本文结合某型号星载计算机的研发工作，针对RTEMS在多处理器环境支持机制中存在的严重缺陷给予修正，对RTEMS的多处理器支持机制予以完善，并对RTEMS应用于大规模数据处理系统中在数据分发时存在的性能瓶颈给出一种优化方案。希望在完善RTEMS 系统的同时也可以对其他嵌入式实时操作系统的开发起到一定的指导和借鉴意义。

2 RTEMS多处理器支持机制的完善

2.1 RTEMS多机支持机制的缺陷解析

RTEMS 通过在用户层和核心层之间的多处理器支持层(MPSL)来实现对多处理器的支持。RTEMS中MPSL是一种基于共享内存的硬件支持，采用消息传递的通信方式的结构。MPSL是一个层次化的结构，其由MPCI接口层、共享内存实现层与硬件相关实现层3部分组成。MPCI接口层为上层应用服务组件提供一套基于包的消息传递接口，而共享内存实现层则是基于共享内存的MPCI接口的实现模块。为实现MPCI接口层的包交换语义，RTEMS共享内存的组织也采用包(packet)作为基本单位，将packet封装后按结点组织成多条通信队列来分配和管理共享内存，封装后的 packet 称之为envelope。此外还有一条空Envelope队列用于管理整个共享内存中尚未使用的Envelope包。

假设一段起始于地址base，length大小的共享内存区域，图1即为该段共享内存的逻辑组织形式。可见，对于一个有N个结点的多处理器系统共享内存中需要组织N+1个逻辑队列用以分别管理各结点所接收到的未处理的Envelope包和系统内尚未使用的Envelope包。

作者简介：宋 伟(1981－)，男，硕士研究生，主研方向：并行操作系统；杨学军，教授、博士生导师

收稿日期：2006-09-25 **E-mail：**frank1997@gmail.com

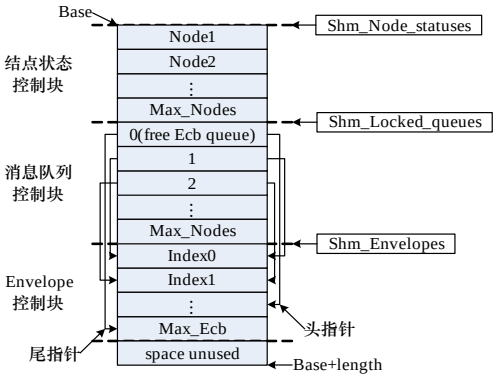

图 1　共享内存逻辑组织

此外，为实现各结点间通信的及时性，在 RTEMS 中每个结点再启动时都会创建一个服务器线程 MPCI server(MS)。系统在最初创建 MS 线程时还会定义一个系统信号量(MPCI Semaphore)用于指示 MS 的调度时机。最初信号量值为 0，MS 阻塞在此信号量上。当结点 A 向结点 B 发送 Envelope 包时，结点 A 会触发结点 B 的机间中断。结点 B 的中断处理函数会将信号量 MPCI Semaphore 的值加 1，唤醒结点 B 上的 MS 服务器。MS 服务器到本结点在共享内存内的队列中收取 Envelope 包并对 Envelope 包进行解析，执行相应被请求的操作。

在上面的论述中可以看到 RTEMS 对远程请求的执行是由 MS 服务器代理远地进程完成的，显而易见，如果远程请求的操作被阻塞那么 MS 服务器就会被阻塞(在这一点上 RTEMS 4.6.2 的用户文档《RTEMS C User's Guide》的描述与代码实现有些出入，这里以源代码实际实现的功能为准)。这种机制对信号、事件等对象的支持是完好的，但是在频繁的消息通信环境中，或者是在对临界资源的激烈竞争时这种机制就会显露其致命的缺陷。通过对 RTEMS 所提供的用户接口 API 在多机环境中的实测，发现在对信号量的支持以及消息队列的支持中都会暴露 RTEMS 多机支持机制的这一缺陷。下面以 RTEMS 对信号量的支持为例通过对一个执行流程的分析说明这种缺陷的发生以及影响。

首先假设这样一个情况，一个嵌入式系统中有 A、B、C 3 个结点，结点 A 创建一个全局信号量 S，在某一时刻结点 B、C 近乎同时发出对 S 的 obtain 请求，假设 B 的 Envelope 请求包略先于 C 的 Envelope 请求包到达结点 A，也就是说在 A 的共享内存中组织的消息队列中 B 的 Envelope 请求包排列在 C 的前面。此时结点 A 的共享内存中的消息队列的组织可由图 2 中时刻 1 示意。

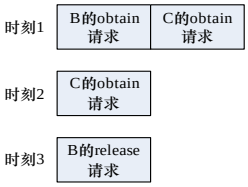

图 2　结点 A 共享内存消息队列示意

来自 B 的机间中断到达后，A 结点上的 MS 服务器被中断处理函数唤醒，接收 B 的 obtain 请求包，此时结点 A 的共享内存中的消息队列的组织可由图 2 中时刻 2 示意。A 结点的 MS 服务器解析 B 的 obtain 请求包，并将 S 的使用权交给 B。然后 MS 服务器再从消息队列中接收 C 的 obtain 请求包，并解析请求包。但是由于 S 的使用权已经交给 B，因此 MS 服务器只能阻塞直到 B 释放 S 的使用权。

当 B 完成对临界资源的访问后即向 A 发送对 S 的释放请求，此时结点 A 的共享内存中的消息队列的组织可由图 2 中时刻 3 示意。于是 A 再度收到来自 B 的中断，中断处理函数将 MPCI Semaphore 的值加 1 以唤醒 MS 服务器，但是由于 MS 服务器因为 C 的 obtain 请求而阻塞，因此不可能到共享内存的消息队列中做收取 Envelope 包的操作，所以 B 的 release 请求也就得不到响应。

这时整个系统内出现这样一个局面，A 的 MS 服务器等待 B 释放信号量 S 以满足 C 的 obtain 请求并解除自己的阻塞状态去接收下一个请求，而 B 等待 A 的 MS 服务器完成 C 的 obtain 请求后以响应自己的 release 请求。显然 A、B、C 3 个结点都进入了死等的状态，整个系统的多处理器通信机制也因此而崩溃。

2.2　RTEMS 多机支持机制的完善

显然 RTEMS 多处理器支持机制的这种缺陷对于整个多机系统的正常运行而言是致命的。导致这个致命缺陷的原因是 MS 服务器由于亲自去执行远地结点的请求而有机会被阻塞，因此解决这个致命缺陷的最直接的方法就是让 MS 服务器不亲自执行请求而是为每个请求开一个代理服务器，让代理服务器代表远地结点在本地执行请求的操作，而 MS 服务器只负责接收 Envelope 包，创建代理。这种方法虽然可以解决 RTEMS 的缺陷，但是考虑到嵌入式系统的背景，为每个远地结点的请求开启代理服务器势必大量增加系统资源的开销，而嵌入式系统往往内存等资源很紧张，因而这种开销显然是不被接受的。

在某型号星载计算机的研发过程中通过对信号、事件、信号量、消息队列等 4 种主要对象的 API 接口函数的测试，发现 RTEMS 的这种缺陷对于信号和事件两种对象几乎没有影响，也就是说为每种对象的远地请求开启代理执行是不必要的。

经过分析，发现事件和信号的接口 API 实际上都不存在阻塞的问题，其请求是可以被顺利执行完毕的。所以实际上只需要对可能会存在阻塞问题的 API 函数进行修改，使其对这些函数的执行采用代理即可。图 3 即为改进后的 MS 服务器的执行流程示意图。

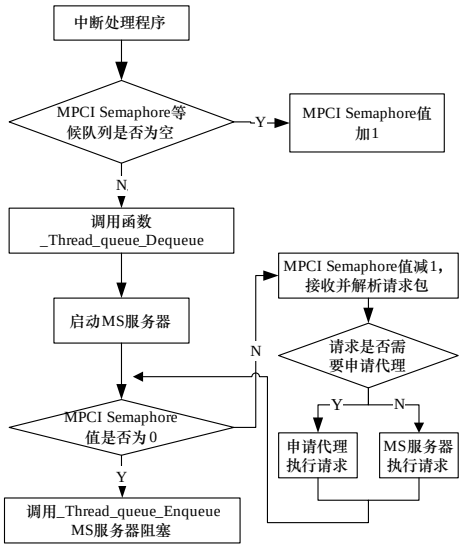

图 3　改进后的 MS 服务器的执行流程示意图

通过对信号、事件、信号量、消息队列等 4 种主要对象

的 API 接口函数的分析，发现可能会存在阻塞问题的其实只有信号量的获取函数和消息队列的接收函数，所以 MS 服务器只要在处理这两种远程请求时创建代理服务器就可以解决 RTEMS 对多处理器支持上的缺陷，而且这种修改方法并不需要耗费大量的系统资源，并通过在某型号星载计算机的研发过程中的实际测试证明这种修改方法是完全可行且有效的。

3 RTEMS 多处理器支持的性能优化

在前文中提到过 RTEMS 的 MPSL 实现是基于包的，在 RTEMS 的默认配置中，Envelope 的大小是 416 个字节，去掉封装信息后的 packet 的大小是 384 个字节，也就是说通过 MPSL 一次可传输的内容不大于 384 个字节。这个大小对于信号量、事件、信号等普通对象的使用是足够的，但是在实际应用中有时会需要大量的传送数据，比如在星载计算机中如果要对图像进行处理，往往是一个结点从外设接收图像数据并将其分发给其余结点作并行处理，若要保证处理的及时性，对图像分发的速度要求就会比较高。

假设一个 3MB 的图像文件需要通过 RTEMS 的消息队列分发给各结点，在使用 RTEMS 默认的包大小配置时至少需要传递 8 192 次包。也就是说系统要至少进行 8 192 次封装包、传输、解析包等操作。像这样一次分发操作需要进行如此大量的传输 Envelope 包的工作，其软件行为的开销是十分巨大的，这使得整个分发的传输开销变得很大。而嵌入式系统中采用多处理器的并行化计算是为了获得更好的实时性，而如果传输的开销太大，则会严重影响系统的实时性能。

考虑到这种开销主要是源于需要传输的 Envelope 包的数量太多，想要降低软件在这方面的开销应该尽可能减少需要传输的包的数量，由于包的数量和包的大小成反比，很自然地想到要增大包的大小。但是实际上简单地增大 Envelope 包大小的方法是不甚合理的。

RTEMS 对于共享内存的划分是基于 Envelope 包的，包越大，共享内存内可存在 Envelope 包就越少，这样导致的结果是系统内可同时存在的通信消息就越少。对一个频繁交互的系统而言这显然是不可取的。而且诸如信号量、事件、信号以及大量消息队列的消息等应用，其包含有用信息的包的大小都是很小的，对于这些应用而言 384 个字节的包是足够大的，而增加包大小对于这些应用而言不但是一种浪费，而且还会增加拷贝包时的开销。而这些应用往往在多机系统通信中占主要部分，因而简单地增大包的大小的方法得不偿失，是不可取的。

可见有效的方法应该是在尽量保证系统内有足够的可用 Envelope 包的前提下，尽量减少数据分发时的包传输次数。也就是说一方面要尽量不增大 Envelope 包大小，另一方面要尽量增大分发数据时使用的包的大小。

显然如果使用 Envelope 包来实现数据的分发必然会陷入包大小和包数量这对矛盾之中。所以最直接的解决方法就是改变共享内存的组织形式，将两种类型的应用区别对待。

在一个 N 结点的多处理器系统中，将共享内存划分为两个区域，一个区域用于实现 RTEMS 原有的 MPSL 对共享内存的组织方式，而另一个区域划分为 N 个部分，每个部分用于接收分发给该结点的计算数据。这样共享内存的两个区域一个用于进行诸如信号量等小规模数据传输的通信，保证系统内有足够的 Envelope 包供通信使用，另一个则专门用于大量数据的分发，确保可以尽量减少一次分发数据所需要的包传输的次数。图 4 即为优化后的共享内存的逻辑组织形式。

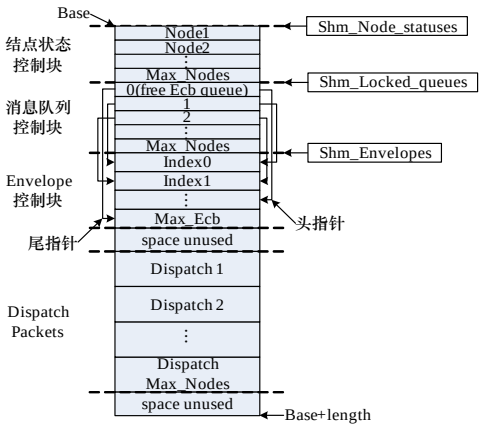

图 4　优化后的共享内存逻辑组织

这种共享内存的组织方式虽然在一定程度上破坏了 RTEMS 原有的 MPSL 的分层结构，但是只要在共享内存实现层对数据分发的发包过程进行封装，对上层用户提供统一的接口函数，一样可以达到对用户隐藏具体硬件实现的目的。

4 优化后的 RTEMS 的多处理器支持的性能测试

在某型号星载计算机的研发工作中笔者对前面所讨论的性能优化方法进行了实现，并对优化前后作了性能对比测试，证明了优化方法的有效性。测试是在一个由 4 个主频为 20MHz 的处理器搭建成的多处理器系统环境下进行的。该系统的共享内存大小为 64KB(优化后被逻辑上分成两个 32KB 的区域)，共享内存的硬件访问速度大约为 6MB/s。

让结点 1 将一个大小为 336 948B 的数据分别发送给其他 3 个结点，即整个共享内存内的数据通信量是 1 010 844B。表 1 记录了优化前后的传输时间和传输速度等项目的比较数值。

表 1　性能测试实验结果

	优化前	优化后
一次传输大小/B	256	8 188
传输次数	3 951	126
传输时间/μs	3 702 945	302 353
传输速度/KB · s^{-1}	266.6	3 264.9

可见经过优化后速度的提高是很明显的，可以提高 10 倍左右的速度。另一方面虽然用于组织 Envelope 包的共享内存在优化后减半只有 32KB，但依旧可以组织 78 个左右的 Envelope 包，这对于多处理器的正常通信是足够的。另外由于测试环境的限制，优化方法的实现中，用于分发数据的那部分共享内存采用的是轮询的方式，极大地增加了共享内存的访问负担，如果也改成中断的方式应该可以得到更高的速度。但由于实验条件的限制并未对此做出实现验证。

5 小结

RTEMS 作为一个功能齐备性能优良的开源嵌入式操作系统有着广泛的应用，对其进行完善和优化具有一定的实用价值。本文针对 RTEMS 对多处理器的支持机制上存在的可能会使整个多机系统死锁崩溃的问题进行了修改完善，并对 RTEMS 在数据分发时所存在的性能瓶颈提出了一种优化方法，同时结合某型号星载计算机的研发工作给予了实际的测试验证，证明了方法的可行性和有效性。这对于基于 RTEMS 的多处理器嵌入式系统的开发乃至其他多处理器嵌入式操作系统的开发都将具有一定的指导和借鉴意义。

（下转第 76 页）