第 35 卷　第 20 期　　计　算　机　工　程　　2009 年 10 月
Vol.35　No.20　　Computer Engineering　　October 2009
·软件技术与数据库·　文章编号：1000—3428(2009)20—0091—03　文献标识码：A　中图分类号：TP316.2

多任务并发调度在 SDH 网元中的应用

文俊浩[1]，张雅洁[1]，吴　涛[2]
(1. 重庆大学软件学院，重庆 400030；2. 中兴通讯股份有限公司深圳研究所，深圳 518055)

摘　要：为提高 EMS/Agent 集中式管理方法中 Agent 的并发处理性能及响应时间，对网元 Agent 中的多任务并发调度机制进行研究，提出采用轻量级 Job 取代任务作为最小调度单位，结合多线程异步并发机制构建网元 Agent 消息处理调度框架。该技术已应用于中兴通讯的光传输设备中，有效提高了系统的并行处理能力，并改善了交互响应时间。

关键词：嵌入式；多任务并发调度；SDH 光传输设备；网元 Agent

Application of Multitask Concurrency Schedule in SDH Network Element

WEN Jun-hao[1], ZHANG Ya-jie[1], WU Tao[2]
(1. College of Software, Chongqing University, Chongqing 400030;
2. Zhongxing Telecommunication Equipment Co. Ltd. Shenzhen Institute, Shenzhen 518055)

【Abstract】 In order to improve the concurrency processing performance and response time of Agent in EMS/Agent centralized management, this paper proposes an efficient NE Agent architecture. It uses lightweight Job to replace task as the smallest units of system schedule, and combines with multi-threaded asynchronous mechanism to improving parallelism. At present, the technology has been successfully applied in ZTE's optical transmission equipment and effectively improved the parallel processing capacity and interactive response time of the system.

【Key words】 embedded; multitask concurrency schedule; SDH optical equipment; network element Agent

1　概述

随着光纤数字通信的迅猛发展，PDH 设备已逐步退居为局域性的传输辅助设备，同步数字体系(SDH)凭借其灵活可靠和高效的网络运行与维护、强大的故障恢复和保护功能等优点，成为应用最为广泛的传输技术。

SDH 作为一种新的传输体制，采用 EMS/Agent 的集中式管理结构，如图 1 所示。

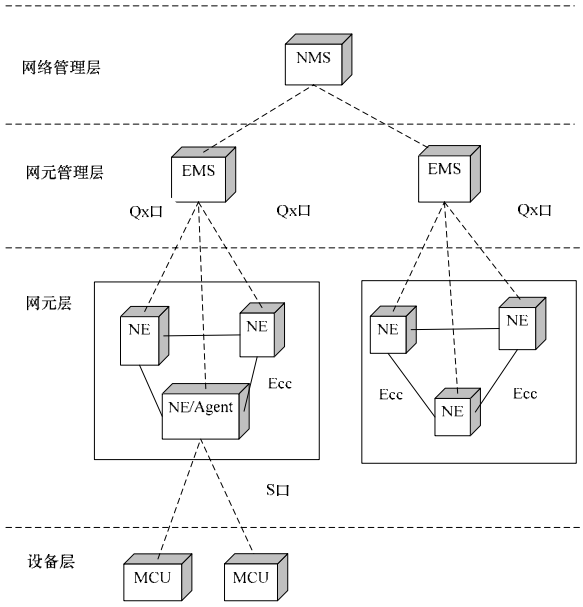

图 1　SDH 网络管理系统结构

基于多个网络管理站(ADM、中继机设备等)和若干个网络元素(NE)组成的 SDH 的光传输网络，主要可以划分为 4 个管理层次：网络管理层，网元管理层，网元层和设备层。整个网络通信主要由 3 个部分组成：网管(EMS)，网元代理器(Agent)和处理传输业务的单板(MCU)[1]。

网元 AGENT 在传输网络管理组织结构中处于网元层的位置，在 SDH 设备中起着承上启下的中心枢纽作用：一方面接收和处理 EMS 下发的命令，另一方面上报网元设备的告警，性能等信息。一个 Agent 可以被一个或多个网管管理，但只有一个具有写权限。Agent 可以通过 TCP/IP 的 Qx 接口与 EMS 连接，也可通过 S 接口，采用 HDLC 通信机制与其他单板进行一对多的通信，还可以通过 SDH 的嵌入式通道 ECC 与网络上其他站点的 Agent 通讯。因此 Agent 的性能决定了整个 SDH 系统的性能，它必须具有强大的控制处理能力、高可靠性和稳定性。

为提高 Agent 的并发处理性能及响应时间，结合网元 Agent 的实际环境，本文提出在嵌入式操作系统的任务调度基础上再进行轻量级的 Job 调度，然后结合多线程并发机制来构建网元 Agent 的体系结构，对 EMS 的设置查询命令和 MCU 上报的命令进行统一的调度和分发。

基金项目：国家“863”计划基金资助项目“面向通信行业的嵌入式软件开发平台”(2002AA1Z2306)
作者简介：文俊浩(1969－)，男，教授、博士，主研方向：数据挖掘，软件工程；张雅洁，硕士研究生；吴　涛，工程师
收稿日期：2009-02-01　E-mail：zhangyajie_koy@126.com

2 嵌入式轻量级多任务调度机制

对于大部分嵌入式操作系统(如 vxWorks)最基本的调度单位是任务，如果对每类命令处理都创建一个任务的话，将会极大地消耗资源；同时任务间的切换也会对系统性能产生影响，任务间的同步互斥等问题也是一个不可忽视的问题[2]。因此，须将任务轻量级化，即在每个操作系统任务下管理多个子任务(Job)。任务的调度是由操作系统完成，而 Job 调度是由任务实现的，每个 Job 有自己的堆栈区和数据区，但没有自己的运行空间，类似 Windows 下的线程，但不同的是 Job 是在任务中统一管理的。每个 Job 都有一个入口函数，完成某个业务处理，这样通过创建轻量级的 Job 来取代任务，采用合理的 Job 并发机制可以显著的提高 CPU 利用率，减少内存消耗和提高用户响应速度。

2.1 Job 调度机制

为达到使用轻量级的 Job 来取代任务的目的，系统中创建的任务更像是一个容器，真正完成业务处理的是容器中待调度的 Job。每个任务具有不同的优先级，操作系统按优先级抢占的方式对其进行调度。每个任务管理下的 Job 具有跟该任务同等的优先级，Job 之间的调度按照 FIFO 的轮转原理进行调度[3]。每个任务都是一个大循环，完成对 Job 的调度。任务本身有一个存储消息的队列，除此外还维护 2 个消息队列，*ready* 和 *block* 队列。如果一个 Job 对应的队列中有消息，则将该 Job 插入到 *ready* 队列尾，任务依次从 *ready* 队列中取出队列头的 Job，并从该 Job 的队列中取出消息，调用 Job 的处理函数进行消息处理，该 Job 返回后，如果队列中还有未处理的消息，则将其插入 *ready* 队列尾，否则将其插入 *block* 队列尾，一个 Job 不能连续被执行，即使其队列中有消息需要处理，这种轮转调度可以使具有同等优先级的 Job 都能得到执行。当任务的 *Ready* 队列中没有等待调度的 Job 时，任务从自身队列中取出消息派发到响应的 Job 队列中，如果队列中没有消息则该任务会交出控制权进入 *block* 状态，其他任务可以得到调度。Job 调度流程如图 2 所示。

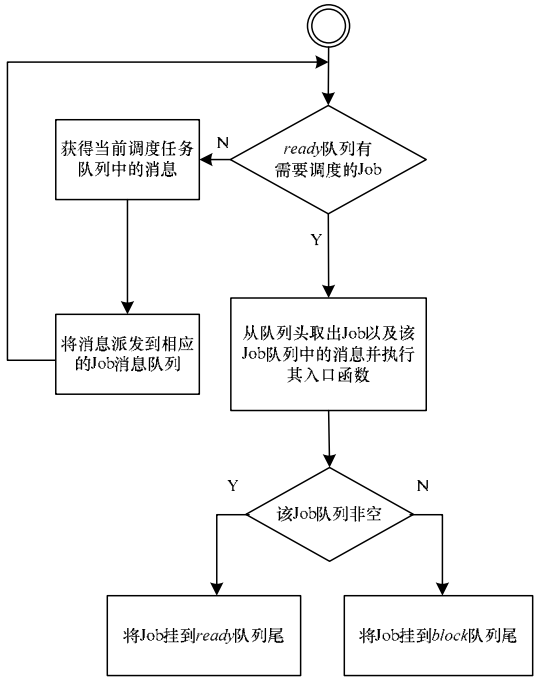

图 2 Job 调度流程

2.2 Job 间的通信

对于通信领域而言，应用是消息驱动的，每个 Job 主要是完成某类消息的处理，Job 之间通过消息队列进行信息交互，所有的消息传递通过消息指针实现，无需内容拷贝，从而大大提高了效率，减少了内存碎片。

Job 之间的消息发送可以分为任务内和任务之间。同任务内通信将消息发送到 Job 的队列中，任务间通信将消息发送到任务的队列中，进而由任务将消息派发到相应的 Job 队列中。

这种轻量级的 Job 调度，具有如下几个优点：Job 占用的内存空间比任务少很多；同一任务下的 Job 是顺序执行的，使得 Job 之间不存在互斥的问题；各 Job 由于有自己独立的堆栈空间，当某个 Job 被阻塞的时候不会影响其他的 Job 运行，Job 级别的并发程度可以达到很高的水平。因此，使用该调度机制可以支持大量的任务/Job 并发执行，有效解决通信嵌入式设备系统资源有限和高并发度需求之间的矛盾。

3 网元 Agent 的多任务并发调度

网元 Agent 主要实现 ISO 组织在其提出的开放系统互联(Open System Interconnection, OSI)，网络模型中定义的几个管理功能域。包括配置管理、维护管理、故障管理、性能管理、安全管理等。除了 OSI 定义的管理功能域外，通常还需要实现系统管理、日志管理、版本管理等。

Agent 需要接收来自 EMS 的配置、维护等命令处理后下发给 MCU，同时又需要将来自 MCU 上报的告警、性能信息处理后发送给 EMS。因此，Agent 处理通常包括以下几个过程：建立连接并接收数据、分析/处理数据、发送数据。一个 Agent 可以与多个 EMS 和上百个 MCU 通信，因此，Agent 处理数据的机制对设备处理并行性以及整个网络的效率都有着至关重要的影响。

采用细粒度的多线程并行技术可以提高系统的并发处理能力。本文提出的轻量级 Job 调度机制，具有类似多线程切换代价小，占用资源少，响应速度快的优点，本文结合多线程并发机制来设计网元 Agent 的多任务调度。

3.1 多线程并发结构

多线程机制可以提高系统的并行处理能力，有效的改善交互相应时间。目前多线程并发结构主要有以下几种：多线程流水线机制和异步多线程并发机制[4]。

多线程流水线使用队列存放待处理的消息，整个处理过程被分解成几个相互独立的处理阶段，每个阶段是一个预先创建的线程，各线程之间也使用消息队列来传递数据和缓存消息，当前线程从队列中取出消息并处理完毕后，将处理后的消息放入下一个阶段处理线程的队列中，这样多个线程采用流水线作业的方式完成对队列中的消息处理。

多线程并发机制是根据完成任务或接收事件类型的不同需要创建多个事件处理线程，每个线程完成某个特定的任务，会有一个事件分配线程负责从消息队列中取出消息，根据消息/事件的类型，将消息放入相应的事件处理线程的消息队列；事件处理线程从队列中取出消息进行处理和分析，然后发送数据。处理中通常采用异步非阻塞操作来提高系统性能。

具体采用何种机制主要依赖于应用的场景。两种机制中都有处理消息的线程，当要处理的消息优先级不一样时，优先级高的消息需要优先处理响应，流水线机制决定了该结构只能按照 FIFO 的顺序来处理，对于网元 AGENT 来说，配置、维护的命令的优先级要高于故障、性能命令，因此多任务流

水线机制并不能很好的满足要求。

3.2 Agent 调度机制

本文结合网元 Agent 的实际场景需要,创建轻量级的 Job 来取代嵌入式操作系统的任务，并结合多线程并发机制来设计 Agent 的体系结构，以提高系统的并行处理能力和交互响应时间，保证传输网的可靠、高效运行。

网元 Agent 是通信方式驱动，以命令处理为中心的代理软件，采用如图 3 所示的结构进行命令的调度分发。

根据本文提出的采用轻量级 Job 取代任务的并发调度机制，首先根据 Agent 要完成的功能，需要划分出如下几个轻量级 Job：Qx 口接收 Job，S 口接收 Job，调度 Job，配置管理 Job，维护管理 Job，告警管理 Job，性能管理 Job 等。按各 Job 优先级不同分组，将优先级相同的分为一组，即需要同一个任务进行调度。分组情况见图 4。

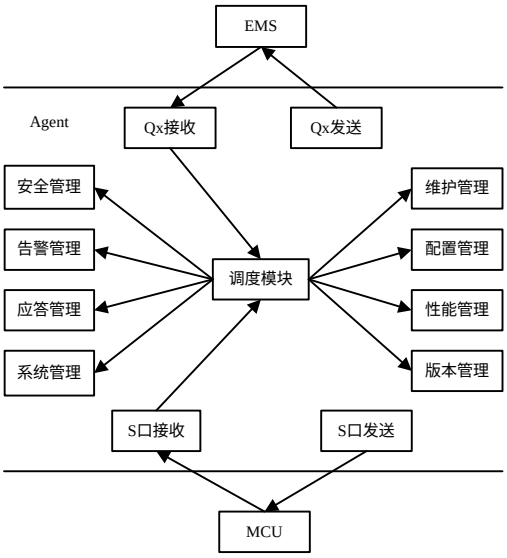

图 3　Agent 调度处理流程

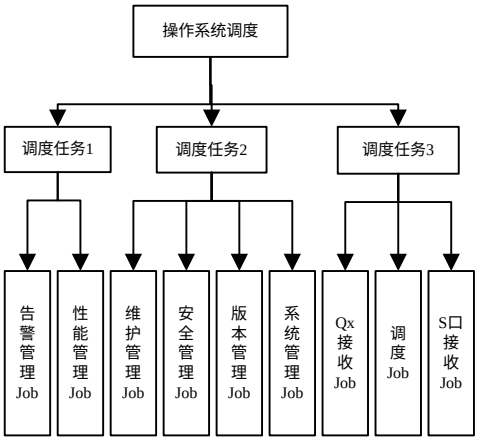

图 4　Agent 分级调度

系统需要一张静态 Job 注册表来维护所有的 Job 实例，每个 Job 主要包括如下几个表项：

```
typedef struct tagT_JobItem
{
WORD32    dwTno;              调度该 Job 的任务号
WORD32    dwJid;                    Job 标识
VOID      *pEntry;                Job 入口函数
WORD32    dwStackSize;            Job 堆栈大小
  WORD32  dwDataRegionSize;   Job 数据区大小
}T_JobItem;
```

该表包含了系统中所有 Job 的信息，需要由哪个任务调度，入口函数，堆栈大小等。该表是在操作系统创建调度任务前就确定的。同时每个任务也会维护一张 Job 描述表，放在本任务空间中，该表记录了需要本任务调度的 Job 信息，当操作系统创建任务后，任务就会从系统注册表中读取信息写入自己维护的 Job 描述表中。

Job“创建”后，Agent 可以采用多线程并发机制来进行命令调度了，主要包括以下几个过程。

(1)建链阶段：Qx 口接收 Job 和 S 口接收 Job 接收来自 EMS 和 MCU 的信息，并将消息发送到调度 Job 的消息队列；

(2)分析/处理阶段：调度 Job 从消息队列取出消息，按照预先定义好的报文格式进行报文解析，根据报文头中的命令编码字段，找到正确的业务处理 Job，将异步消息发送到该线程的消息队列中；

(3)发送阶段：业务处理 Job 将应答消息发送给 Qx 口发送 Job 或 S 口发送 Job，从而将应答消息发送给 MCU/EMS。

4　结束语

本文采用异步多线程机制，增加调度 Job 可以有效地提高系统的并发处理能力，同时采用轻量级的 Job 代替任务，节省了内存资源，降低了任务间切换的开销，提高了多任务并发的系统效率和响应速度。当某个 Job 被阻塞时不会影响其他 Job 运行，从而提高了系统的健壮性。

目前该机制已经应用在中兴通信的光传输设备中，经过系统运行试验，Agent 系统工作稳定，响应实时，功能完善，满足了 SDH 网络系统的要求，保障了 SDH 传输网可靠、高效的运行。

参考文献

[1] 刘　芳. 一种 SDH 网元管理系统设计考虑[J]. 山东通信技术, 2002, 22(3): 17-20.

[2] 夏旭丰, 丁文杰, 朱善君, 等. 实时多任务嵌入系统的实现[J]. 计算机应用研究, 2003, 20(9): 120-122.

[3] 刘　飞, 罗克露, 黄烨明, 等. 通信领域嵌入式系统调度管理的设计[J]. 计算机应用研究, 2004, 21(7): 186-188.

[4] 李　刚, 金蓓弘. 基于线程的并发控制技术研究与应用[J]. 计算机工程, 2007, 33(14): 43-45.

编辑　金胡考