分布式实时系统的自适应资源管理中间件

廖 勇[1]，陈旭东[1]，桑 楠[1]，胡林辉[2]，熊光泽[1]，朱清新[1]

(1. 电子科技大学计算机科学与工程学院 成都 610054; 2. 四川石油管理局地球物理勘探公司 成都 610213)

【摘要】提出了一种基于实时CORBA的自适应资源管理中间件服务及基于最优收益的QoS自适应机制，向开发人员提供了一套面向实时任务的QoS编程接口，并通过资源自感知和QoS自适应使应用级资源分配独立于底层平台。自适应资源管理中间件降低了分布式实时系统中应用程序开发成本和复杂度，并使系统在不确定环境下能动态优化资源分配以适应外界变化。

关 键 词 自适应资源管理; 中间件; 实时系统; 调度; 实时性可移植

中图分类号 TP316 文献标识码 A

Adaptive Resource Management Middleware in Distributed Real-Time Systems

LIAO Yong[1]，CHEN Xu-dong[1]，SANG Nan[1]，HU Lin-hui[2]，XIONG Guang-ze[1]，ZHU Qing-xin[1]

(1. School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054;

2. Geophysical Company, Sichuan Petroleum Administration Chengdu 610213)

Abstract The platform-specific code in the distributed real-time systems (DRTS) is a main contributer to the complexity and cost of application software development. It reduces the maintainability and transplantability, but is not capable of ensuring the performance of real-time tasks. In this paper, an adaptive resource management middleware (ARMM) service based on the real-time CORBA and a QoS (quality of service) adaptation mechanism based on the optimal reward are proposed. A set of QoS APIs that is orient to real-time tasks is provided. The user-level resource management and software development are separated from the hardware platform by resource self-profiling and QoS adaptation. Thus the cost and complexity of application software development of DRTS is reduced, and the resource allocation under unpredictable environment can be dynamically optimized according to the outside variation.

Key words adaptive resource management; middleware; real-time systems; scheduling; transplantability of real-time

传统实时系统软件的开发需要对实时任务的参数进行精确的、定量化的描述，如执行时间、周期等，并通过可调度分析向系统提供实时性保证，而这部分代码的实现需了解底层平台的处理能力，是造成实时软件开发高复杂性和高成本的重要因素[1]。随着应用的推动和芯片制造技术的不断革新，实时系统变得日趋复杂化。新一代实时系统软件的开发具有以下特征：(1) 底层平台因硬件不断升级而使得系统资源容量与计算速度不断提升；(2) 系统因外界环境变化而表现出不确定性[2-5]。如何在不同底层平台、在动态与不确定条件下开发出低成本的、并能确保应用程序性能的实时软件系统成为了新的挑战[6-9]。此外，对于性能敏感的分布式实时系统与平台相关的代码增大了应用软件开发的工作量，降低了可维护性与可移植性，并且不能动态确保任务的实时性[1]。为了避免上述问题，需要一种能自适应不同硬件平台与外界环境变化的中间件，以向系统提供新编程接口，向应用系统提供动态性能保证，从而使新一代实时软件开发具有平台无关性。

1 基本概念

1.1 实时性可移植

由于应用系统的实时性与底层平台处理能力(如CUP速度)相关，使实时性在某一平台上得以确保的应用系统在新平台上未必能得以满足。因此，开发人员需要花大量时间与成本调整并测试原有应用程序，使之能适应新平台。虽然现有对象请求代理人(object request broker，ORB)能实现应用系统功能

收稿日期：2006－06－04；修回日期：2006－08－18

作者简介：廖 勇(1976－)，男，博士生，主要从事分布式实时系统、实时高度方面的研究.

可移植，但在实时性方面却不可移植。因此，必须开发出新的中间件服务以实现应用系统的实时性可移植。

1.2 资源自感知

资源自感知能感知并衡量不同底层平台的资源性能(如CPU、内存等)的能力，是实现实时性可移植的基础和依据。

2 面向实时任务的QoS

为了向分布式实时系统(DRTS)中的任务提供动态实时性确保，并且使任务之间的性能互不影响、相互隔离，本文将QoS机制引入实时任务中，提出了一种自适应资源管理中间件(adaptive resource management middleware，ARMM)。ARMM根据应用系统的QoS需求为任务动态分配资源，并提供QoS保障。另外，当前大部分实时应用在资源消耗上均具有一定灵活性[1-2]，为此本文将应用系统提出的QoS需求用QoS表来描述，定义为：$L_i(Q_i^d, Q_i^{\min}, V_i[k], R_i[k], P_i)$。其中，$L_i$为第$i$个任务的QoS表；$Q_i^d$为所期望的QoS级别；$Q_i^{\min}$为最小可接收的QoS级别；$V_i[k]$为级别$Q_i[k]$所对应的值(如执行时间、周期等参数，$Q_i[k] \in [Q_i^{\min}, Q_i^d]$)；$R_i[k]$为级别$Q_i[k]$给系统带来的收益；$P_i[k]$为违背$L_i$给系统造成的损失。这样在底层资源过载时，ARMM可协调任务QoS以获得最佳系统收益。一个QoS表可由$[Q_i^{\min}, Q_i^d]$内的n个级别构成，而级别数以及每个级别的参数由应用系统的性能要求决定。

根据QoS表中的参数，ARMM将应用系统QoS映射成相应资源请求量。这种映射依赖于底层平台性能，当底层平台发生变化，ARMM能感知底层资源状况，并根据QoS表自适应地为任务选择最优QoS级别，保证任务的实时性要求。

3 ARMM体系结构

ARMM体系结构如图1所示。图中ARMM设计为两层：上层为QoS层，与硬件和实时操作系统(real-time operating systems，RTOS)独立，在下层基础上向用户提供有关QoS的编程接口；下层为自适应层，将底层硬件与RTOS抽象成一个实时虚拟环境，向上层提供自适应任务调度服务。

3.1 QoS层

QoS机制需要新的QoS API，本文在原有的API基础上提供了新的QoS扩展，可使ARMM具有更好的可维护性，易于代码升级。QoS层向用户提供6个API：A1用于为新任务τ_i创建QoS表L_i；A2用于为新建QoS表注册参数；A3用于将定制好的QoS表提交给下层许可控制器；以让自适应层根据该表确定任务QoS级别；A4用于返回任务当前QoS级别；A5用于删除QoS表；当新任务τ_i被接收后，A6用于将任务τ_i与其对应QoS表L_i绑定。

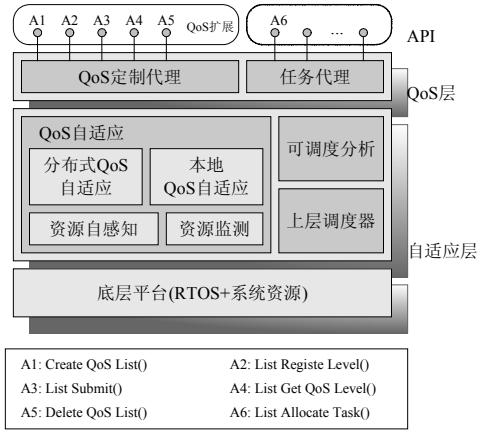

图1 ARMM的体系结构

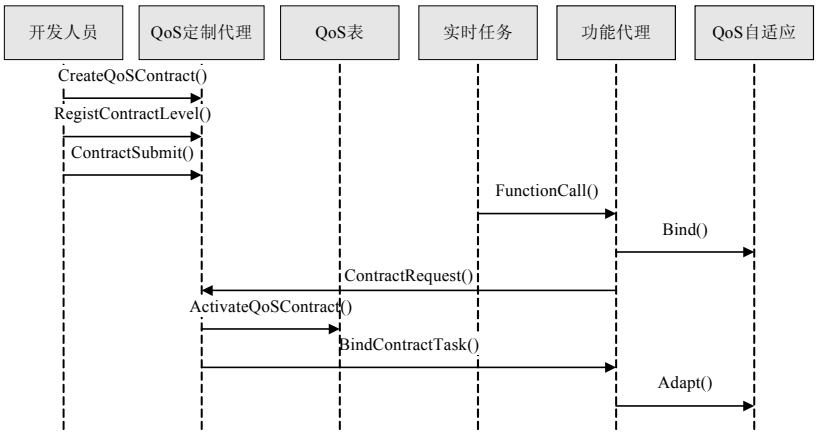

图2 QoS表的定制

QoS表的定制如图2所示，其定制与执行在逻辑上是分离的。QoS表的定制表一旦被定制，就存放

在定制代理中；当新任务到达系统，任务代理将激活其对应QoS表，并提交给自适应层，由QoS自适应模块根据系统资源状况为任务分配资源。QoS表的执行则是在自适应层内部进行，当ARMM检测到外界环境变化，将触发QoS自适应，在全局范围内动态调整资源分配，但该过程必须遵循任务QoS表。

3.2　自适应层

3.2.1　结构及流程

QoS自适应模块用于动态调整实时任务QoS级别，使之适应当前系统资源状况，并保持系统收益最优化，包括本地QoS自适应和分布式QoS自适应。资源监测模块用于监测DRTS范围内的CPU资源消耗情况。为实现应用系统的实时性可移植，ARMM设计了资源自感知模块以动态地确定底层平台的处理能力，而不必每次因平台升级进行的人工测量与估计。资源自感知模块周期性更新实时任务实际执行时间 E_i^A 与ARMM为之分配的执行时间 E_i^B 的比率 $r = E_i^A / E_i^B$ 。当 $r<1$ 时，表示自适应模块为任务分配的CPU资源大于它当前实际资源需求量，造成CPU资源浪费，自适应模块可提高任务QoS级别；当 $r>1$ 时，表示任务分配的CPU资源小于它当前实际资源需求量，造成CPU过载，自适应模块必须降低任务QoS级别以满足其可调度性；而 $r=1$ 时，表示任务分配的CPU资源恰好等于它当前实际资源需求量，自适应模块无需做任何调整。这样可使应用级别CPU资源预算与软件开发独立于底层硬件。当改变底层平台时，通过修正比率r以及QoS自适应调整可确定出新的QoS分配方案。可调度分析模块用于判定当前CPU资源是否能满足任务实时可调度性，它与QoS自适应模块协同工作对新到任务进行许可控制。调度器一方面用于更新经自适应调整后的任务参数，并传递给底层RTOS，由内核负责具体的任务调度；另一方面，当一个新任务到达系统和外界环境发生变化时，将触发任务代理向自适应层发出一个QoSAdapte()操作，进行资源动态调整，如图3所示。

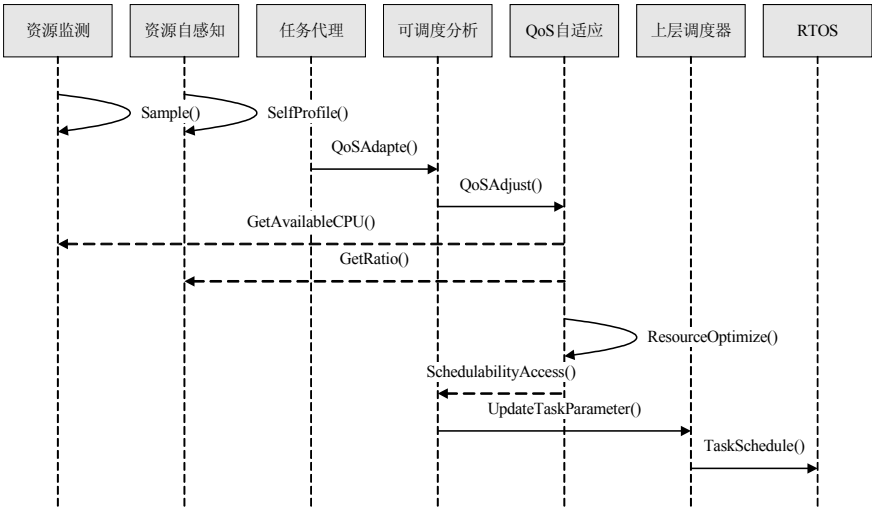

图3　QoS自适应

3.2.2　QoS自适应

如上所述，自适应层负责全局范围内的CPU资源管理。对于自适应策略，学术界提出了许多方法，如反馈控制法[2]，该方法将反馈控制理论结合到实时调度中，解决开放系统中自适应CPU分配问题以实现过载保护。但这种动态调度要求设计一种反馈调度器，并且必须从理论上对调度的稳定性进行分析。本文采用基于最优收益的在线调整法，考虑周期性实时任务，任务之间彼此独立，任务的QoS定义为周期，任务τ_i表示为$\tau_i(E_i, P_{id}, P_{i\min}, P_{i\max}, \omega_i, R_i(P_i))$。其中，$E_i$为执行时间；$P_{id}$为期望的周期；$P_{i\min}$、$P_{i\max}$分别为最小周期和最大周期；$\omega_i$为权重，代表任务的重要程度或者能够调整参数($P_i$)来适应动态环境的幅度；$P_i$为任务实际分配到的周期($P_i \in [P_{i\min}, P_{i\max}]$)；$R_i(P_i)$为周期$P_i$给系统带来的收益。任务的累计利用率为 $U_P = \sum_{i=1}^{N} u_i$ ，其中，u_i为任务的实际利用率， $u_i = E_i / P_i$ 。$U_{\text{lub}}^{\text{A}}$ 是系统在给定底层调度算法A时的可调度上限，对于RM(Rate Monoton)算法，$U_{\text{lub}}^{\text{RM}} = N(2^{1/N} - 1)$，其中$N$为当前任务数。给每个任务$\tau_i$赋予一个收益函数$R_i(P_i)$，该函数代表当$\tau_i$分配到的周期为$P_i$时给系统带来的收益。因为 $u_i = E_i / P_i$，所以，τ_i的收益是其分配到的利用率u_i的函数 $R_i = R_i(u_i)$ 。这样对于一个可调度的系统，其带权收

益为 $R_w=\sum_{i=1}^{N}w_iR_i$ 。再将系统中的任务分为关键任务和非关键任务两类。关键任务在整个运行过程中不允许改变其周期P_i，收益函数表示为：

$$R_i=\begin{cases}R_i(u_{i0}) & u_i=u_{i0}\\ 0 & u_i\neq u_{i0}\end{cases} \quad (1)$$

对于非关键任务，其周期P_i却可以随外界环境的变化而改变，收益函数表示为：

$$R_i=\begin{cases}0 & u_i<u_{i\min}\\ R_i(u_i) & u_{i\min}\leqslant u_i<u_{i0}\\ R_i(u_{i0}) & u_i\geqslant u_{i0}\end{cases} \quad (2)$$

如果一个可行的调度使得系统的带权收益最大，那么该调度是一个最优调度[3]，则QoS自适应的目标就是找到一个CPU资源分配方案使得系统的带权收益 R_w 最大。有关该在线调整算法的细节参见文献[3]。

4 ARMM的实现

ARMM以实时CORBA的中间件服务TAO的Kokyu调度器[10]来实现。ARMM与TAO的关系如图4所示。ARMM位于TAO的Kokyu调度器之上、应用系统之下，具有较好的灵活性与可移植性。ARMM用C++开发，作为一个高优先级的任务运行与RTOS之上，向应用系统提供符合CORBA标准的接口。与TAO相比，ARMM支持应用系统的实时性可移植，动态调整CPU资源分配以适应外界环境变化。

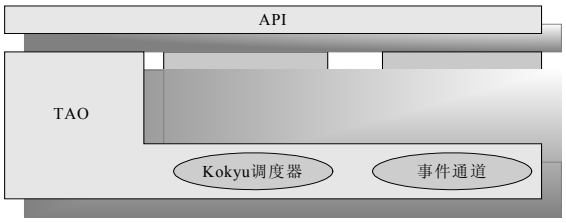

图4 ARMM与TAO的关系

5 结 论

为了降低分布式实时系统中应用程序开发的复杂性，并使系统能在动态、不确定环境下实现自适应CPU资源管理，本文将QoS机制引入到实时任务中，提出了一种基于实时CORBA的中间件服务ARMM。当底层平台发生改变时，ARMM能自感知CPU的处理能力，并通过QoS自适应来确保应用系统的实时性；当外界环境发生变化时，ARMM能自适应地调整任务QoS，在确保系统可调度性的前提下使系统总收益最大化。ARMM实现了应用系统的实时性可移植以及分布式实时系统自适应CPU资源管理。

参 考 文 献

[1] ABDELZAHER T F. QoS adaptation in real-time systems [D]. Michigan: Michigan University, 1999.
[2] LUI S, TAREK A, KARL E A, et al. Real time scheduling theory: a historical perspective[J]. Journal of Real-Time Systems, 2004, 28: 101-155.
[3] LIAO Yong, CHEN Xu-dong, SANG Nan, et al. Optimal reward based adaptive CPU resource allocation for computing devices in pervasive environment[J]. Journal of Information and Computational Science, 2005, 2(1): 75-80.
[4] LU Chen-yang, STANKOVIC J A, et al. Feedback Control Real-Time Scheduling: Framework, Modeling, and Algorithms[J]. Real-Time Systems Journal, 2002, 23(1-2): 85-126.
[5] LU Chen-yang. Feedback control real-time scheduling [D]. Virginia: Virginia University, 2001.
[6]LIAO Yong, CHEN Xu-dong, SANG Nan, et al. Optimal reward based adaptive CPU resource allocation for computing devices in pervasive environment[J]. Journal of Information and Computational Science, 2005, 2(1): 75-80.
[7] 李 允, 熊光泽, 罗 蕾, 等. 普及计算终端的自适应性技术研究[J]. 电子学报, 2002, 30(8): 1121-1126.
[8] ABDELZAHER T F, STANKOVIC J A, LU Chen-yang, et al. Feedback performance control in software services [J]. IEEE Control Systems Magazine, 2003, 23(3): 74-90.
[9] ABDELZAHER T F, SHARMA V, LU Chen-yang. A utilization bound for aperiodic tasks and priority driven scheduling[J]. IEEE Transactions on Computers, 2004, 53(3): 334-350.
[10] GILL C, SCHMIDT D, CYTRON R. Multi-paradigm scheduling for distributed real-time embedded computing [C]//IEEE Proceeding, Special Issue on Modeling and Design of Embedded Software. [S.l.]: [s.n.], 2003.

编 辑 税 红