·计算机工程与应用·

功能性能分离的可信服务构件模型

桑 楠，苏 芮，廖 勇，黄小红，侯 炜，李 波

(电子科技大学计算机科学与工程学院 成都 610054)

【摘要】为了向分布式实时嵌入式系统(DRES)的运行过程提供可信保障服务，方便上层应用程序的开发，提出了一种针对DRES的可信构件服务模型QUOCCM。该模型将系统功能实现与性能确保分离开，把应用程序的性能确保模块抽象成通用的服务确保构件，为运行于动态环境下的应用程序提供自适应的可信保障服务。该文以一个模拟实现的飞行控制系统验证了QUOCCM模型的可行性和灵活性。

关 键 词 自适应实时系统； 构件模型； 可信服务； QoS管理

中图分类号 TP302.1 文献标识码 A doi:10.3969/j.issn.1001-0548.2010.02.018

Confidence Component Model for Separating Function and Capability

SANG Nan, SU Rui, LIAO Yong, HUANG Xiao-hong, HOU Wei, and LI Bo

(School of Computer Science and Engineering, University of Electronic Science and Technology of China Chengdu 610054)

Abstract To provide credible services and facilitate the development of applications for the distributed real-time embedded system (DRES), a credible service component model, QUOCCM, is proposed. By QUOCCM, a DRES application is separated into two parts: function realization and performance guarantee. The latter is encapsulated into some components to provide common services for the former. Specially, QUOCCM can provide an adaptive service support about application performance running in the dynamic environment. Finally, a flight control system in DRES has been used to confirm the feasibility and flexibility of the model.

Key words adaptive real-time system; component model; credible service; QoS management

当前，面向对象技术在分布式应用中得到广泛发展，随着分布式技术和面向对象技术的结合，产生了大量基于分布式对象中间件的模型，如OMG组织的CORBA、Microsoft的DCOM、Sun公司的RMI等。然而，一些对性能要求苛刻(performance-critical)的DRES(distributed real-time embedded system)，如航空电子系统、舰载电子系统等，要求在满足系统功能需求的同时，也能提供可信(实时、仿危、安全、可靠等)保障服务。目前的实时中间件技术(如TAO)并没有提供有效的动态QoS(quality of service)确保功能，如何将动态QoS自适应机制应用于已有的实时中间件和构件框架中，被认为是构件技术要求解决的主要问题之一[1-2]。

为了满足上述要求，提高软件系统开发的效率和质量，本文采用CCM(CORBA component model)机制实现了DRES中应用程序的功能实现与性能确保分离。在模型设计中，引入性能确保构件，将应用系统要求的性能确保策略动态加载到该构件提供的策略库调用接口，并设计系统状态监测构件，使其提供为调用性能确保策略的依据。该方法不仅能提高DRES应用软件的可信性(尤其是实时自适应)，而且能通过构件技术提高性能构件的可重用性。

1 相关研究

CCM构件通过定义一组特性和服务扩展原CORBA对象模型，分离应用中的业务逻辑和服务支持逻辑，将对服务的需求、资源的依赖、构件的部署位置要求、构件间的连接信息等声明在描述文件中，由分布式构件平台解析描述文件，实现构件在分布式环境中的安装、创建和激活，并在运行时刻提供构件所需的服务支持逻辑。CCM构件模型提供了定义CCM构件外观特征的方法，该模型的独特之处在于构件提供服务和所需服务的对称接口定义。利用对称接口定义，借助于对构件连接依赖关系的

收稿日期：2008－09－12； 修回日期：2009－05－04

基金项目：国家863计划(2006AA01Z173、2007AA01Z131)

作者简介：桑 楠(1964－)，男，教授，主要从事嵌入式实时系统方面的研究.

描述，最终可以实现构件的组装重用[3-4]。但是，CCM并没有将应用系统的功能实现与性能确保QoS分离开，而是将系统的性能保障策略与功能实现整合为一体进行构件开发，势必会加大DRE系统开发的难度和周期，更不能实现性能保障机制的通用性。

文献[5]提出的QuO(quality of service for CORBA objects)机制推动了分布式中间件的发展。QuO是基于CCM的构件机制，以一种框架的形式存在于分布式对象应用中，并引入了自适应行为，为分布式应用提供各种各样的工具和设施以保障端到端QoS。为了实现自适应行为，QuO扩展了传统中间件技术，使用一些相关技术，如面向方面编程(aspect oriented programming, AOP)、代码生成、反射、开放实现、封装以及分布式服务和库[6-7]。虽然QuO实现了分布式应用系统中的功能与性能相分离，但并不能管理系统中不确定性任务的动态变迁。

2　QUOCCM构件模型

要实现多种性能保障机制广泛地、可重用地应用于各种应用系统，需要将系统的功能与性能分离，并且将各种性能保障机制独立，以构件的形式存在于整个应用系统。另外，考虑到要缓解系统资源的利用率与满足应用端功能需求间的矛盾，需要专门的对象检测系统资源，当系统资源的利用率发生改变时，能够主动改变应用端的实现行为，使得在满足功能需求的同时能满足系统的性能要求。为了解决上述问题，即针对当前实时系统不支持不确定载荷管理、不支持实时性可移植、系统性能保障策略不可重用以及QoS自适应行为实现，在现有中间件对性能保障机制研究[8-11]的基础上，本文在ACE-TAO平台上基于CCM框架提出了一个随环境和资源动态变化而自适应调整服务质量的QoS体系结构QUOCCM(quality objects common component management)模型。

2.1　QUOCCM模型框架

在QUOCCM模型中，引入了自适应行为(adapter action)和QoS级别(Qos_level)，同时也包含了QoS机制，并设置QoS合同对象(contract object)、系统状态对象(system condition)、任务监测器(task monitor)、资源感知器(resource perceive)、代理对象(delegate object)、自适应行为触发(transition behavior)、消息返回对象(callback object)等，模型框架如图1所示。

系统状态对象(system condition)：为监控系统的硬件资源、CPU利用率、内存占有量、应用系统运行机制等提供了查询接口。通过调用该对象，可获得应用系统的当前运行状况。

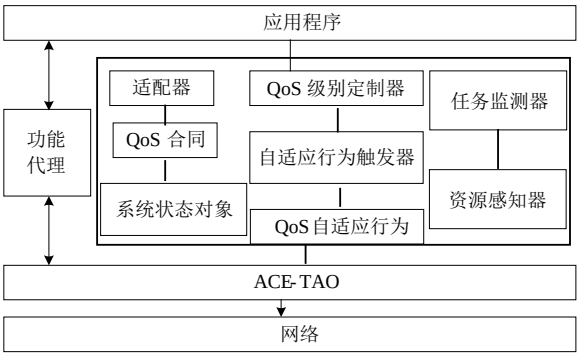

图1　QUOCCM构件模型框架

QoS级别(Qos_level)：一个状态级别由一系列的系统状态值组成，状态值即表明当系统处于该QoS级别时，所监控的系统对象要处于具体状态参数所设定的范围。同时，也表明了应用系统在响应某一服务请求时希望系统应该达到的性能状态，并以参数形式具体反映。

QoS合同(contract object)：根据一系列的系统状态值而预先定义不同的QoS级别及当系统状态发生改变时要触发的自适应行为(在本文应用系统中，自适应行为也即确保系统实时性能的调度算法)。

任务监测器(task monitor)：探听当前应用系统的任务动态运行状况，当发生动态改变时，触发QoS合同评估引发自适应行为。

资源感知器(resource perceive)：周期性地评估当前环境下任务的实际执行时间与预估计执行时间间的比值，当比值不为1就重新读取合同，修改该任务的QoS级别，调整该实时系统的运行状况。

自适应行为触发(transition behavior)：当监测到系统状态改变时，调用自适应行为确保系统的性能要求，且不同状态间转变要触发不同的行为操作。

自适应行为(adaptor action)：当系统引发新任务执行时，通过自适应行为调整每个任务的运行状态(QoS级别)确保系统的实时性能，如果任务迁移到新的运行平台，则重新调整该任务的QoS级别。

消息返回对象(callback)：当系统状态发生变化而触发新的操作时，都要将该消息以信号量机制回传给客户端。

2.2　QUOCCM模型内构件间通信机制

QUOCCM模型允许用户和应用对紧急实时任务的QoS需求参数、管理策略、运行机制以及系统自适应调整策略进行用户级定制。系统首先对用户的定制服务质量需求进行分析，结合系统当前的资源使用情况选择最佳QoS级别；然后对资源配置、

系统运行策略、运行平台作出自适应调整，提高资源利用率；最后可以向用户反馈当前系统QoS服务质量的供应情况，用户根据情况做出服务质量的调整。

应用开发者针对每个实时任务设定一系列的QoS级别参数(预估计任务的执行时间、执行周期、QoS级别等)，并且指定需要监控的系统状态对象，包括当前系统的网络带宽、系统资源(根据具体的应用而定)等。

由应用开发者定制的QoS合同将预先设定的一系列QoS级别映射为一组系统运行状态值，在合同中预先声明系统运行状态的变迁所引发的自适应行为(根据每个实时任务对应的QoS级别表，由实时调度算法实现对各个实时任务的运行参数进行动态调整)。

若资源的变化导致某些构件实例的资源需求无法满足，系统状态对象将通知QoS合同触发器，通过读系统状态对象获取当前系统运行状况以及读合同得到构件实例实际需要的系统运行状态值，由系统运行状态变迁事件通知自适应行为重新调整每个实时任务的运行参数，即每个任务的执行时间和运行周期，从而改变每个任务的优先级，调整应用系统的运行状态和运行策略，确保应用系统的性能需求；若系统突发新的任务使得当前CPU过载，则重新评估QoS合同，通过自适应行为动态调整每个任务的运行参数(QoS级别)确保系统的实时性，实现对不确定载荷管理；若发生实时任务的移植，资源自感知模块周期性地评定系统中每个实时任务的实际执行时间与估计执行时间间的比率，当两者间的比率不为1时，动态调整该任务的QoS级别，实现在确保CPU不过载的基础上最大限度地利用CPU。构件间的交叉联系如图2所示。

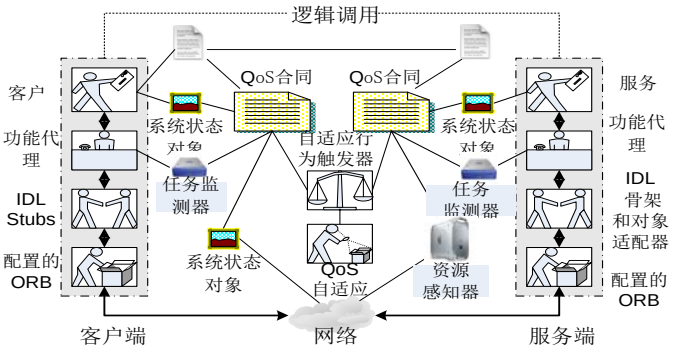

图2 QUOCCM构件模型中构件间的通信示意图

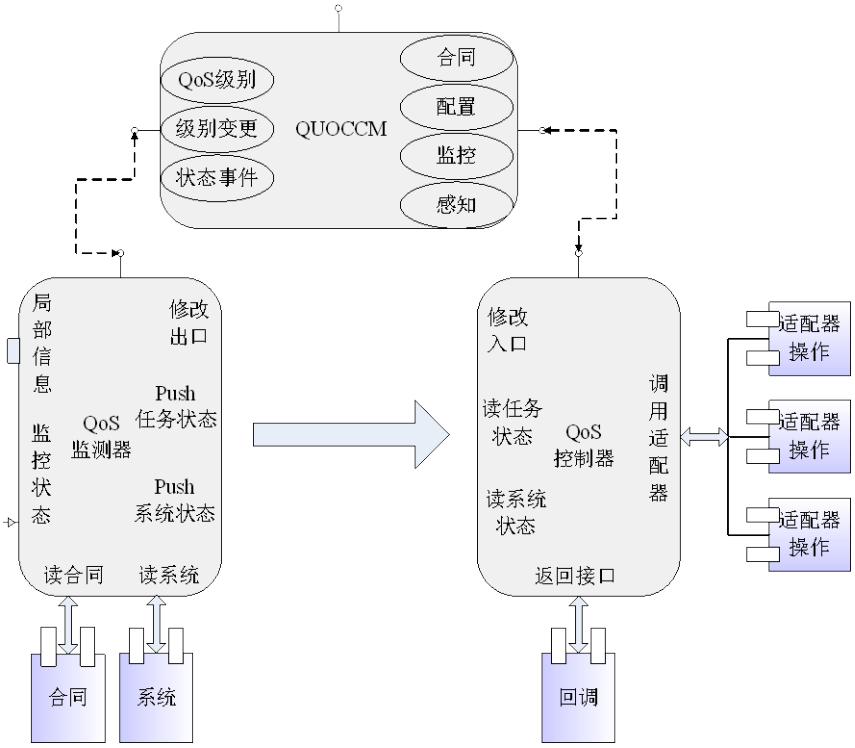

图3 QUOCCM模型构件组建

2.3 QUOCCM框架的构件描述

由一个模块QUOCCM预先声明所有构件公有的接口，而构件QoSMoniter和QoSControl继承其中的部分接口，通过继承的接口实现构件间的通讯，并在其中的一些接口定义中加载用户自定义函数，从而完成对系统设备状态的监控、调用性能保障的功能函数及评定QoS级别等，如图3所示。

2.3.1 构件接口预声明

```
module  QUOCCM {
    struct QosLevel{
        …
    };
    struct LevelChange{
        QosLevel  old_level;
        QosLevel  new_level;
};
    eventtype StatusEvent{
        public  Boolean  change;
        public  LevelChange  level_change;
        public  int  transition(in  LevelChange
level_change);
    };
    interface contract{
        QosLevel  get_level(in  string
task_name);
        LevelChange  get_change(in  string
task_name,);
    };
interface allocation{
        QosLevel  allocation(in  string
task_name,  in  QosLevel  old_level,  in  QosLevel
new_level);
    };
interface monitor{
        Boolean  task_monitor();
        QosLevel  new_task(in  string
task_name, in QosLevel task_level);
    };
interface perceive{
        float  task_test(in string task_name, in
QosLevel task_level);
    };
    …
};
```

类似于C++中类的定义，预先声明QUOCCM模型中各构件的刻面、接插口、事件源、事件槽。其中，struct QosLevel{}描述各个任务的QoS级别；struct LevelChange{}记录每个任务的QoS级别的动态变化；由定义的事件源StatusEvent监控系统中任务的QoS级别，一旦发现有任务动态改变了QoS级别，则立即将消息发布给该事件的证订者，即构件QosMonitor和构件QosControl间的传递的通知事件。而interface contract{}、interface allocation{}、interface monitor{}、interface perceive{}分别为模型提供读取QoS合同、启动动态调度触发器和监控系统状态对象接口。

2.3.2 性能监控构件接口

性能监控构件所对应的IDL描述文件为：

```
module  QUOCCM{
component QosMonitor supports contract{
        publishes StatusEvent modify_out;
        provide perceive resource_perceive;
        provide monitor task_monitor;
        randomly attribute QosLevel task_level;
        attribute QosLevel qos_level;
    };
    home  QosMonitorHome  manages
    QosMonitor{};
};
```

通过IDL描述文件描述性能监控构件QosMonitor对外的接口以及与其他构件(QosControl)间的通信接口。其中，supports contract表示构件QosMonitor继承了接口contract；modify_out表示构件QosMonitor监听到系统中实时任务的QoS级别的动态改变而向外界发布的消息接口；而resource_perceive和task_monitor分别表示由构件QosMonitor向外提供的Facet接口，通过该两个接口为构件QosControl提供自适应策略实施依据。

2.3.3 性能确保构件接口

对性能确保构件进行的IDL描述文件：

```
module  QUOCCM{
component QosControl supports allocation{
        uses perceive resource_perceive;
        uses monitor task_monitor;
        consumes StatusEvent modify_in;
    };
    home QosControlHome manages
        QosControl{};
```

};

通过IDL描述文件描述性能监控构件QosControl对外的接口以及与其他构件(QosMonitor)间的通信接口。其中，resource_perceive和task_monitor分别表示构件QosControl向构件QosMonitor请求的向外提供的Facet接口。通过该两个接口，构件QosMonitor为构件QosControl提供自适应策略实施依据；modify_in表示构件QosControl对监听事件StatusEvent的证订，即由modify_in接收构件QosMonitor监听到系统中实时任务的QoS级别的动态改变，向外发布事件源modify_out。

2.3.4 性能监控构件的CIDF

性能监控构件对应的CIDF描述文件为：

```
composition session QosMonitor_Impl {
    home executor QosMonitorHome _Exec {
        implements  QUOCCM::
            QosMonitorHome;
        manages QosMonitorHome _Exec;
    };
};
```

2.3.5 **性能确保构件的**CIDF

性能确保构件所对应的CIDF描述文件为：

```
composition session QosControl _Impl {
    home executor QosControlHome _Exec {
        implements  QUOCCM::
            QosControlHome;
        manages QosControlHome_Exec;
    };
};
```

通过编写CIDF文件，描述构件QosControl和构件QosMonitor，以及构件所对应的Home接口的实现(QosMonitorHome_Exec、QosControlHome _Exec)和持久状态。由编译器CIDLC根据IDL和CIDL文件自动生成构件程序框架，最后将自适应策略的源码加载至程序框架中就完成了整个模型的设计。

3 实 验

为了验证QUOCCM的可行性，本文在WIN XP操作系统上搭建ACE_TAO平台，在该平台上通过修改Mico_CCM中关于CCM中的实现框架，并增加部分与QoS管理相关的性能构件服务，实现支持QUOCCM的构件框架原型。下面以一个航空电子实时资源管理应用为例，对QUOCCM的QoS管理有效性进行验证。具体系统设计如图4所示。

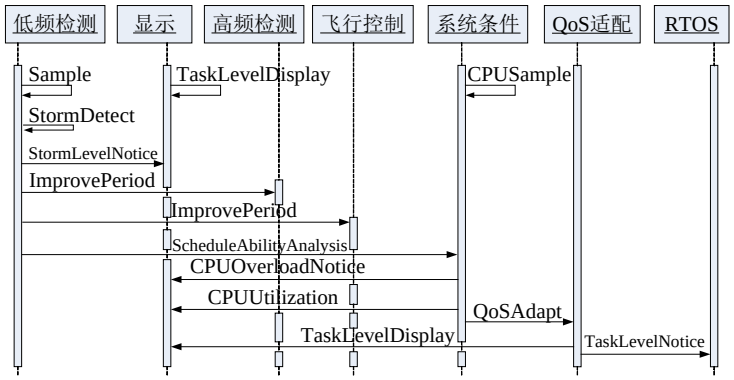

图4 飞行监控系统运行顺序图

在应用中，实验设计了飞行导航(guidence)、飞行控制(controller)、高频飞行监测(fastnavigation)、低频飞行监测(slownavigation)及显示控制(display) 5个实时任务，任务的具体参数如表1所示。每个任务有QoS级别数(level)、收益(reward)、执行时间(exec_time)及执行周期(period)4个参数。该模拟系统确认导航飞机的飞行路线以及周期性监控飞机的飞行轨迹是否偏离预定轨道，如果偏离则动态调整每个任务的执行状态。为了检验该系统中QoS管理机制的有效运作，引入一个系统变量(stromfind：0表示没有风暴发生；1表示有风暴发生)，其中stromfind的设定由StormLevelUpNotice触发。

在本文实验中，处于正常的环境，飞机的系统资源能够保证任务的正常运行，Guidance及SlowNavigation任务处于最高运行级别，此时它们的执行周期最短；而Controller及FastNavigation任务则运行在最低级别，它们的执行周期处于最长状态。当风暴发生时，引发飞机偏离航道，系统触发自适应行为，自动提高Controller及FastNavigation任务的运行级别，并且启动自适应调度算法将Guidance、SlowNavigation和Display的相关参数进行重新调整，以满足飞行安全需要的。实验的运行结果如表2所示。

表1　系统任务QoS级别对照表

任务	QoS级别	收益	执行时间/ms	周期/s
飞行导航	0	10	100	10.0
	1	15	100	5.0
	2	20	100	1.0
飞行控制	0	100	60	1.0
	1	104	80	1.0
	2	120	60	0.2
	3	124	80	0.2
高频飞行监测	0	1	60	5.0
	1	100	60	1.0
	2	120	60	0.2
低频飞行监测	0	10	100	10.0
	1	100	60	5.0
	2	120	60	0.2
显示控制	0	5	80	15.0
	1	8	70	10.0

表2　实验运行结果

任务	QoS级别			
	风暴程度0	风暴程度1	风暴程度2	风暴程度3
飞行导航	2	0	0	0
飞行控制	0	1	2	3
高频飞行监测	0	1	2	3
低频飞行监测	2	0	0	0
显示控制	0	0	1	1

4　结 束 语

本文对性能要求苛刻的DRES提出了一种基于CCM构件系统的QoS框架模型—— QUOCCM模型。该模型的主要特征为：(1) 支持动态不确定载荷管理，(2) 系统性能保障机制与功能实现相分离。通过实验证明QUOCCM模型能够以独立的构件形式实现动态QoS自适应机制，确保动态不确定任务的实时性。但是，为了更进一步保证DRES的实时性，将在未来的研究中继续探索系统的实时性可移植问题。

本文研究工作得到电子科技大学青年基金的资助，在此表示感谢。

参 考 文 献

[1] MIGUEL M A. Integration of QoS facilities in component container architectures[C]//Proc 5th IEEE Int'l Symp Object Oriented Real-Time Distributed Computing. Los Alamitos, CA: IEEE Computer Society Press, 2002: 394-401.

[2] SCHMIDT D, KACHROO V, KRISHNAMURTHY Y. et al. Developing next-generation distributed applications with QoS-enabled DRE middleware[J]. IEEE Communications Magazine, 2000, 17(10): 1-14.

[3] ZOU Yong, HUAI Xiao-yong, LI Ming-shu. The adaptive scheduling ap-proaches for open real-time systems[J]. Chinese Journal of Computers, 2004, 28(1): 58-65.

[4] StarCCM. StarCCM is a CORBA component model implementation in C++ language[EB/OL]. [2007-01-12]. http: //starccm.sourceforge. net/.

[5] SCHANTZ R, LOYALL J, ATIGHETCHI M, PAL P. Packaging quality of service control behaviors for reuse, object-oriented real-time distributed computing[C]// Proceedings of the Fifth IEEE International Symposium on, Washington D.C.：IEEE Computer Society, 2002: 375-385.

[6] WANG N, SCHMIDT D. C, GOKHALE A, et al. Total quality of service provisioning in middleware and applications[J]. Microprocessors and Microsystems, 2003, (27): 45-54.

[7] SCHANTZ R, ZINKY J, KARR D, et al. An object-level gateway supporting integrated-property quality of service. object-oriented real-time distributed computing[C]// Proceedings of the 2nd IEEE International Symposium on. Washington D.C.: IEEE Computer Society, 1999: 223-234.

[8] VAN Der ALST W, VAN H K. Workflow management: mod-els, methods and systems[M]. Boston: MIT Press, 2002.

[9] 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.

[10] 魏　乐, 雷　航. 基于中间件的实时可信构件的开发框架研究与实现[J]. 西华大学学报, 2007, 26(5): 78-80.
WEI Le, LEI Hang. Research of development framework for realtime and dependable component based on middleware[J]. Journal of Xihua University(Natural Science Edition), 2007, 26(5): 78-80.

[11] 廖　勇, 陈旭东, 桑　楠, 等. 分布式实时系统的自适应资源管理中间件[J]. 电子科技大学学报, 2008, 37(1): 101-104.
LIAO Yong, CHEN Xu-dong, SANG Nan, et al. Adaptive resource management middleware in distributed real-time systems[J]. Journal of University of Electronic Science and Technology of China, 2008, 37(1): 101-104.

编　辑　蒋　晓