一种面向任务关键系统的RISC风格的协作构架*)

朱智林[1,2] 郭敬林[1] 李 航[1] 刘西洋[1] 陈 平[1]

(西安电子科技大学软件工程研究所 西安710071)[1] (中国煤炭经济学院计算机系 烟台264005)[2]

A RISC Style Collaborative Framework for Mission Critical System

ZHU Zhi-Lin Guo Jing-Lin LI Hang LIU Xi-Yang CHEN Ping

(Institute of Software Engineering, Xidian University, Xi'an 710071)

Abstract The collaborative framework for the mission and safety critical system is required to support multiple quality of service properties. A RISC style collaborative framework is provided to meet this requirement. The collaborative characters of mission and safety critical system are discussed and the reliable multicast that supports collaborative computing is discussed emphsis on.

Keywords Mission and safety critical system, Collaborative framework, RISC style, Reliable multicast

1. 引言

相互协作完成某一任务是许多信息系统都具有的特征，任务导向及安全临界(MSC, Mission and Safety Critical System)系统也不例外。从软件工程的角度考虑，具有良好特性的协作构架对构建MSC系统是非常必要的。但MSC的协作构架与一般的民用系统是不同的，这是因为MSC系统对服务质量(QoS, Quality of Service)的要求通常都是多样的，其中包括可信性、效率以及可预测性等。而MSC系统的协作构架必须满足该类系统对多QoS的要求，因此如何支持协作特性是关系到MSC系统性能的重要因素。一般的民用系统大多采用商用中间件技术作为系统支撑平台，并采用SRM等可靠多播协议[1]作为通信手段来支持协作计算。另外，采用数据库作为支撑平台来构建特定的协作系统也有一些研究工作[2]。但这两种方式并不适合MSC系统。这是因为采用商用中间件技术可以加快协作系统的开发速度，也能保证系统功能的正确性，但商用中间件一般都由第三方提供，因而往往无法保证MSC系统对多QoS的需求。采用数据库技术支持协作特性虽然容易做到上层应用与协作构架的无缝结合，但也面临可预测性难以满足的问题。由于商业的对象关系数据库柔性不够，而面向对象数据库又很难真正应用，所以这样的协作构架的实用性往往较差。本文提出了一种RISC(精简指令集, Reduced Instruction Set Computer)风格的协作构架。之所以称为RISC风格的协作构架是因为我们在协作构架的设计上借鉴了RISC的设计思想。

2. RISC风格的协作构架

2.1 协作构架的设计思路

现代指挥控制系统属于典型的MSC系统，这类系统对多QoS需求的必要性是显然的。有的QoS是构建系统时的需求，例如系统所采用的构件的可信性。MSC系统对可信性的要求主要体现在两个方面：(1)系统的构件必须是可信的；(2)应用构件过程的过程也应该是可信的。而有的QoS属于MSC系统运行时的指标，例如可预测性。MSC系统对可预测性的要求体现在系统的对外部事件的反应行为以及时限必须是可预测的。MSC系统通常需处理多并发事件的输入数据流，环境的瞬息万变使得这些事件的到来次序和几率通常是不可预测的，但系统对这些事件的响应行为必须是可预测的。当然，MSC系统一般都有固定的运行时指标，所以这种可预测性有范围界定。但可信性等QoS需求是很难有明确的标准来衡量的。为了满足MSC系统对多QoS需求的要求，我们将在MSC系统的协作构架设计中借鉴RISC思想。

RISC原本是硬件设计中的概念。在硬件设计中，RISC的设计原则是使系统达到最高有效速度，将那些能对系统性能产生净增益的功能用硬件实现。其余大部分都用软件实现。这一设计思想在软件领域也开始有一定程度的应用，文[3]是其中比较有代表性的工作。在协作构架的设计中，我们分析了MSC的特点，结合以往项目的经验，按RISC的设计原则来确定协作构架的边界以及各组成部分的关系。由于本文的侧重点在于讨论RISC设计思想在协作构架设计中的应用，所以省略了较繁琐的构架接口定义。

2.2 实时任务关键系统的数据特性分析

实时任务关键系统处理多并发事件的输入数据流，由于系统的任务划分明确，所以绝大部分输入信息的数据类型在系统运行初始时可以确定，并且对输入信息的处理体现了一定程度的周期性。如何管理输入信息以及输入信息的派生数据，对实时任务关键系统的性能来说是非常重要的。在系统运行时，驻留在内存中的数据采用内存数据库技术进行管理是很自然的选择[6]。图1给出的是系统输入信息的数据结构随系统运行周期变化的示意图。对不同类型的数据进行区分是必要的，因为这直接关系到内存索引技术以及Checkpoint方式的选择。直接应用第三方提供的内存数据库系统必然给实时任务关键系统带来冗余的成分，而且这些冗余往往会影响系统性能，甚至使得系统性能变得不可预测，因此在协作构架中直接应用成熟的数据库系统并不是最佳的选择。轻量级数据库(Light Weight Database)生成器(Generator)[7]以及自调谐

*)基金项目：可信软件工程(413150501)。朱智林 博士生，副教授，主要研究方向为软件工程，面向对象技术，数据库技术；郭敬林、李航 博士生；刘西洋 博士，副教授；陈 平 教授，博导。

的 RISC 风格的数据库技术[8]均可以满足实时任务关键系统对 QoS 的要求。从软件复用的角度考虑，RISC 风格的内存数据库技术更适合实时任务关键系统。可预测、自调谐对数据管理是非常重要的。文[8]提出了一个基本的体系结构，采用这种结构的数据库被封装成极小的 RISC 风格的数据管理器，它通过提供精巧、特定的 APIs 实现内存数据管理。目前已在项目前期完成了 RISC 风格的内存数据管理工具的工作[9]，在协作构架的设计上，借鉴了 RISC 风格体系结构数据库的思想，这样做可以使系统的可伸缩性增强。采用这样的结构并通过一定的数学方法，就可以得到可信的可复用构件。

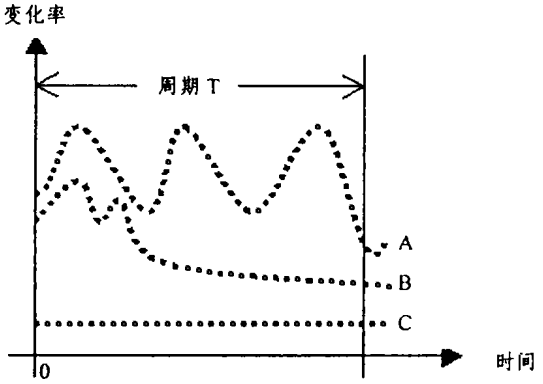

图1　数据结构随运行周期变化示意图

其中横轴是时间，纵轴是信息数据结构随时间的变化率。A 表示信息的数据结构在系统运行周期 T 内不稳定，B 表示信息的数据结构在系统运行周期 T 内较少变化，C 表示该类型信息的数据结构在系统运行时有特定的变化规律。

2.3　协作构架方案

协作是实时任务关键系统的一个重要特征，这个特征和其它特征一样属于实时任务关键系统的有机组成部分。通过上一节对数据以及行为特性的分析，可以了解实时任务关键系统的协作构架在整个系统中所应该起的作用。图2给出的是实时任务关键系统的一个 RISC 风格的协作构架。之所以称之为 RISC 风格是因为：从协议的角度来考虑，协议栈的每一层中的协议都尽可能地简化，通过不同而又相对简单的协议组合来实现相对复杂的功能；内存数据管理采用的是 RISC 风格的设计思想，在体系结构的整体风格上是统一的。

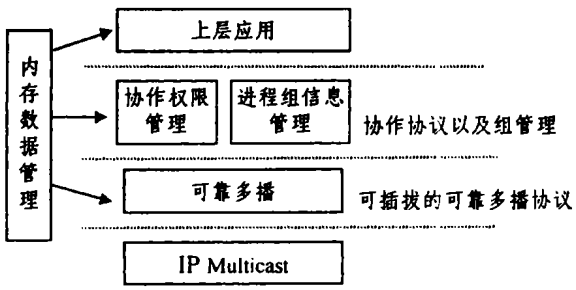

图2　RISC 风格的协作构架

作为一个整体，实时任务关键系统中所有的进程都需要可靠的实时调度。协作构架中不能含有使系统不可预测的成分。从图1中可以看出，该构架为上层应用提供接口，底层是不可靠多播协议。需要指出的是，图示中的 IP Multicast 不是必须的，实际上任何不可靠的 Multicast 的协议均可。同时，由于一般的通信协议都不提供可靠、高效的发送/接收缓冲区管理，而对实时任务关键系统来说这种缓冲区管理是必须的，所以协作构架也必须包含缓冲区管理。从可维护性的角度考虑，整个系统我们采用了同样的 RISC 风格的内存管理策略，该策略可以保证在进程出现问题（Crash）时通过 Fuzzy Checkpoint 实现一定程度上的数据恢复，这项恢复技术同样也适合通信缓冲区。

2.4　实时任务关键系统的行为特性分析

一个任务需要多个用户/节点协同完成在实时任务关键系统里非常普遍。实时任务关键系统的各个子系统之间通过可靠的通信来保证整个系统的正常运转。实际上，共享内存也是一种多播通信方式，因此，本文所讨论的内容对集中式实时任务关键系统在某种程度上也是适用的。可靠、可伸缩的多播协议对实时任务关键系统来说是非常重要的。IP Multicast 提供的是“Best Effort”服务，这对实时任务关键系统而言尚有不足。发送方消息到达收方的顺序也是一个需要考虑的问题，包括全序、因果顺序等。有关可靠多播协议的研究工作很多，DARPA（Defense Advanced Research Projects）支持许多研究多播的课题，文[10]介绍了军用多播的种类和用途。采用何种协议与应用密切相关。考虑两个例子：(1)在一个航迹处理子系统中，该子系统需要将雷达信息及时处理，并向相关的节点传送处理过的信息。对接收方来讲，它允许存在一定程度的信息丢失，这是因为数据融合等手段可以解决某些信息丢失问题，况且从另一个角度讲，最新的航迹消息更有价值（Most Recent First）；(2)容错机制，实时任务关键系统通过异地互为备份来增加抗毁性，容错消息传送需要原子性（即 All or Nothing）。概率多播协议非常适合(1)的应用，而虚同步机制适合(2)。实际上，概率多播协议在特殊环境下的作用是无法代替的，国外对此已有相当深入的研究工作，并在战区导弹防御以及远洋作战编队等许多应用中采用。ACE ORB 提供可插拔协议模式，可以通过配置的手段来解决多种多播协议的支持问题。实时任务关键系统的操作人员和计算机系统间存在着大量的交互。在这些交互中，绝大多数是操作人员/指挥人员做出的决策。决策是有权限区分的，所以支持决策的权限是有一定意义的，这样可以在决策信息发生冲突时依据权限来消解冲突。实际上，从协议的观点来看，权限区分也属于协议的范畴，另外一个属于协议范畴的内容是进程组管理。可以说，协作构架的优劣在很大程度上取决于两点，其一是该构架是否可以充分支撑上层进程组间的各种协议，其二是下层可靠多播协议的效率以及可伸缩性。

3. 协作构架中的不对称性

实际系统中的协作构架是非常复杂的，下面以协作构架中的不对称性为例，讨论该构架所采用的设计思想。

在实时任务关键系统中，节点往往是不对称的。这种不对称性体现在多个方面，例如，实时任务关键系统各个节点的任务一般是不相同的；上行通讯线路和下行通信线路一般是不同的。还有一种不对称性是：在上面所提到的航迹管理子系统中，考虑某个节点向外（需要得到消息的节点）多播航迹信息这一情况。由于可能存在各种干扰，丢包概率很高，通过临近的节点恢复丢失的包比向消息初始的源节点发出恳求信息更有效率。选择保留消息的节点数目和丢包的概率以及节点之间的拓扑结构有关，并不是每个节点都需要保留消息的备份以满足临近节点的需要。解决的策略是在相应的层加入一个节点选择微协议，通过修改配置文件，使用多种微协议的组合就可以达到目的，这恰好是采用 RISC 风格的体系结构的优

势所在。

由此可见，RISC 风格的协作构架具有很好的伸缩性，它的设计思想是通过微协议或者微构件的组合来实现系统目标。但需要指出的是，正如 RISC 不能完全代替 CISC(复杂指令集)一样，RISC 风格的协作构架也有自己的局限性。根据我们的经验，某些对实时性要求极高的系统是不适合采用 RISC 风格的体系结构的。

结论 目前我们的工作集中在可靠多播协议上。可靠多播协议有许多种类。我们主要是通过在开放式源代码中提取形式规范、并根据需要进行一定程度的修改，然后通过 PVS 验证该规范的正确性，这属于协议功能分析的范畴。实时任务关键系统所需的概率多播协议的性能分析也是非常重要的，该工作也在同步进行。任何工作都需要实际系统的验证，下一步的研究工作主要是要在实际系统验证所选用的多播协议，远期的工作目标是形成一定规模的协议库，并将其集成到通用的软件开发工具之中。

参 考 文 献

1 Floyd S, et al. A reliable multicast framework for light-weight sessions and application level framing. In: Proc. of ACM SIGCOMM [C], 1995

2 Huh S Y, Rosenberg D. A Change Management Framework: Dependency Maintenance and Change Notification. Journal of System and Software [J], 1996, 34: 231～246

3 Schmidt D C, Levine D L, Mungee S. The Design and Performance of Real-Time Object Request Brokers. Computer Communications [J], 1998, 21: 294～324

4 Thuraisingham B M, Maurer J A. Information Survivability for Evolvable and Adaptable Real-Time Command and Control Systems. IEEE Transactions ON Knowledge and Data Engineering [J], 1999, 11(1): 228～238

5 Ravindran B. Engineering Dynamic Real-Time Distributed Systems: Architecture, System Description Language, and Middleware. IEEE Transactions Software Engineering[J], 2002, 28(1): 30～57

6 Garcia-Molina H, Salem K. Main memory database systems: An Overview. IEEE Transactions on Knowledge and Engineering [J], Dec. 1992. 509～516

7 Batory D S, Leung T Y, Wise T E. Implementation Concepts for an Extensible Data Model and Data Language. ACM Transactions on Database Systems [J], 1988, 13(3): 231～262

8 Chaudhuri S, Weikum G. Rethinking Database System Architecture: Towards a Self-tuning RISC-style Database System. In: 26th Intl. Conf. on Very Large Databases[C], Cairo, Egypt, 2000

9 杨武军. 基于用户级锁的业务对象的并发控制机制:[硕士论文]. 西安电子科技大学, 2002

10 Macker J P, et al. Reliable Multicast Data Delivery for Military Networking. IEEE MILCOM'96[C]

(上接第146页)

$$Y(k)=[0 \quad c_{12} \quad c_{13}]\cdot\begin{bmatrix}X_1(k)\\X_2(k)\\X_3(k)\end{bmatrix}$$

其中，$k+1$表示下一步的状态，k 表示当前状态，$R(k)$、$U(k)$ 表示输入数据或触发事件，在系数矩阵中对角系数 a_{11}，a_{22}，a_{33}，为自相关系数或子矩阵，表示该数据点当前状态与下一状态的关系，其它系数为互相关系数或子矩阵，表示各数据点之间的相互关系。此处 a_{11}～a_{33}可以是数，也可以是带某种程序运算符的数，这种运算符可把 a 和 X 结合起来，表示某种运算或过程。关于相关系数的矩阵称之为关系矩阵。

实现：该模块的实现就是关系矩阵的细化、求精、程序化。

软件 IC：基于面向状态的风格，我们可以建立具有复杂功能组合的大型模块/组件，因为不需要考虑数据独立和屏蔽，所以状态和功能可以比较自由地组合。在上例中，假设 C 模块经常要根据顾客的要求修改，我们就把它拿出来，只组合 A、B。当模块与外界无耦合时，面向状态的风格就变成了面向对象的风格，面向对象的风格是它的一个特例。

重用：在系统设计时，使用软件 IC 的思想与类、函数等小模块是不同的，应以功能的组合为目的、以软件 IC 模块为中心，围绕这个中心配置外围程序。

系统描述：整个系统的状态空间也可以同样表示，系统与组件的不同在于其全部元素都为子矩阵。

状态空间描述是一种形式化的方法，它并不限于状态空间风格，也可以对其他风格进行形式化，它具有如下优点：

1、形式一致。系统、所有的组件其表达形式都是一致的，易于重用。

2、良好的可扩充性，可求精性。每个子矩阵都是可扩充的可细化的。

3、关系表达清楚。系统整体结构及全部组件的关系都在矩阵中表示出来，无需用复杂的图形表示，静态特性好。

4、动态特性好，因为包含了时序、事件。

5、便于数学分析。所有基于描述语言及结构图的分析都是定性的、表面的，只有基于数学模型的分析才是定量的深刻的。

6、可以利用已有的数学分析工具，如李雅普洛夫稳定原理、系统解耦。

总结 形式化是体系结构设计规范化的重要基石，可以充分开发系统结构中的共性，分析软件体系结构的性质，有利于在系统设计时选择合适的软件体系结构，从而对软件体系结构的选择起指导作用，避免盲目选择。形式化是软件重用的最有效手段。本文提出的面向状态的形式化风格可以构建大型软件 IC，并具有兼顾动态和静态特性等诸多优点。

参 考 文 献

1 Shaw M, Garlan D. Softare architecture: Perspectives on an emerging discipline. 北京：清华大学出版社, 1998

2 张广泉. 软件体系结构：概念、风格与描述语言. 重庆师范学院学报(自然科学版), 2000, 17(3): 1～5

3 Garland, Shawm. An introduction to software architecture [R]. CMU-CS-94-166, 1994

4 姜东胜，张苓，陈莘萌. 应用框架的形式化研究. 武汉大学学报(自然科学版), 2000, 46(3): 285～288

5 Maier M W, Emery D, Hilliard R. Computer, 2001, 34(4): 107～109

6 李也戈，张然. 软件重用的实现. 计算机工程, 1995, 21(4): 37～40