计算机科学1997 Vol.24 №.5

Client/Server 结构系统中的处理分布

Process Distribution in Client/Server Systems

赵洪彪 周立柱

(清华大学计算机科学与技术系 北京100084)

摘 要 Process distribution is critical for Client/Server systems. This paper discusses some issues and their solutions in this regard, including distribution steps, software architecture model and implementation approaches.

关键词 Client/Server, Process distribution, Software architecture, Implementation approach

在Client/Server系统的开发过程中,处理分布是一个重要的问题[1][2],基于Client/Server系统开发的实践,本文讨论了与Client/Server系统中处理分布有关的处理分布过程,处理分布类型的Client/Server软件结构模型和分布软件的构造途径问题。

1 处理分布过程

处理分布是确立Client/Server系统结构的重要步骤,处理分布的过程包括需求分解和处理分布两个步骤。

1.1 需求分解

对系统的功能需求进行分解是客户和服务器之间功能划分的基础,需求分解的步骤包括环境描述、逻辑描述和界面描述,如图1所示。

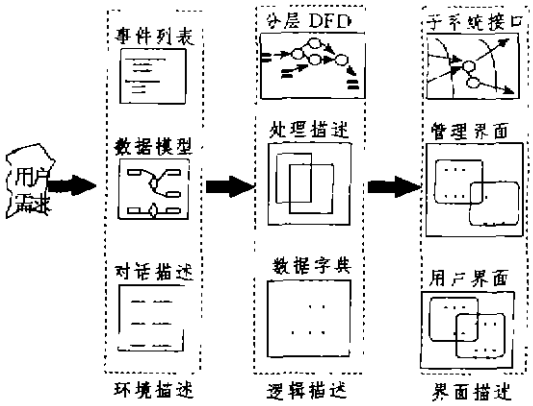

图1 需求分解

环境描述是从用户角度对系统解决方案的描述。在分析用户需求时首先从数据模型和事件表开始,数据模型使我们能够了解用户问题域中真实存在的实体,同时能够明确在数据存储、组织和访问等方面的需求。事件是从用户的观点定义的发生在工作环境中要求应用系统做出反应进行处理的各种用户请求。对话描述是从用户的观点要求应用系统对所发生的事件做出反应的用户界面要求。

逻辑描述表示为了满足用户的需求系统应该具有的数据和逻辑动作,包括系统对环境中的事件所执行的处理,执行各种处理所需要的数据,数据流动和存储的内容,数据内容的转换。

界面描述是对界面和接口需求的描述,包括管理界面、用户界面和系统内部的接口定义。由于GUI对应用系统的功能有较大的影响,有时需要用原型工具来模拟系统的外观和功能,征求用户意见及时进行修改。

1.2 处理分布

客户和服务器之间的处理分配是综合考虑各种功能和性能需求之后才能完成的,在这种分配之前设计者首先要决定哪些功能适合在哪些平台上完成,然后决定如何在客户和服务器各个平台上分割这些功能。处理分布包括机器层分配和任务层分配两个主要步骤,如图2所示:

机器层分配 确定客户平台或者服务器平台所能承担的处理集合,表示哪些功能被分配给了目标环境平台,以及这些平台相互之间的接口。

任务分配 在每个平台的处理集合内,将其划分为不同的任务集合,从应用限制和系统目标的角度对处理集合与任务集合的划分进行评价,并做必要的调整。通过处理器处理流图和任务流图的形式表达这种处理分布。

在确定任务分配时,主要考虑在服务器平台和客户平台内部如何将处理和数据分给任务,决定各任务之间如何相互通讯。任务划分的原则包括:

(1)将与某种服务请求处理有关的所有服务器数据访问需求分配给一个服务器任务。

(2)将支持某个用户界面的所有需求分配给一个客户任务。

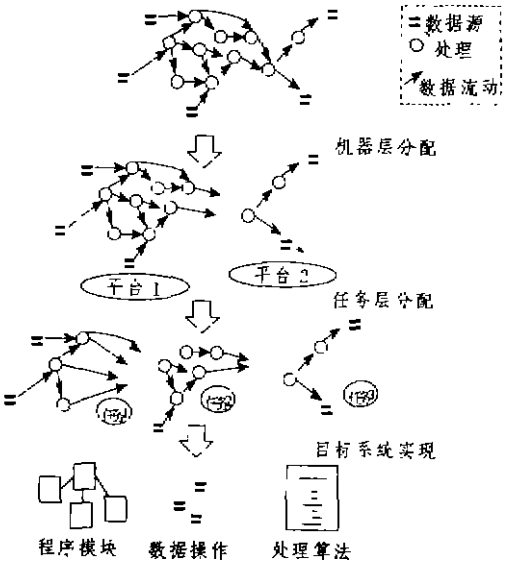

图2 处理分布

(3)将所有访问某个关键资源的需求分配给一个任务。

(4)将支持核心操作的需求分配给支持容错环境中的任务。

处理分布是一个权衡的过程,不存在唯一正确的方案。兼顾目标与限制,同时考虑风险代价和功能的方案才是可取的方案。处理分布必须有清晰的文档以便进行评价。

处理分布依赖于软硬件平台的功能,是权衡比较的结果[4],在完成系统的处理分布时,需要同时确定系统的数据分布,然后进行详细的数据模型设计和模块设计。

2 处理分布类型的C/S系统的软件结构

Gartner Group模型将Client/Server系统划分为不同的组成构件[5]:表达构件、计算构件和数据管理构件,根据这些构件在客户和服务器之间的分布方式,将Client/Server系统划分为五种类型:分布表示、远程表示、分布处理、远程数据访问和分布式数据库。其中分布处理这种类型是一种设计难度较大的Client/Server系统,就此我们提出如下的基于层次的Client/Server系统软件结构模型。

Client/Server系统从软件结构上可以划分为界面层、服务层、支持层和系统层(图3),不同层次对应系统的不同功能。界面层包括服务器接口和客户界面。服务层由服务器和客户两端的服务应用程序组成,负责完成与应用领域有关的具体的业务处理,满足应用系统的功能需求。支持层与具体的业务需求无关,提供了对分布式数据管理和操作的支持和Client/Server消息交换支持。系统层表达目标环境中往往由操作系统和数据库系统提供的软硬件平台的功能,是实现支持层的技术基础。合理地选择系统层的软件和硬件,可以提高应用系统的可移植性。

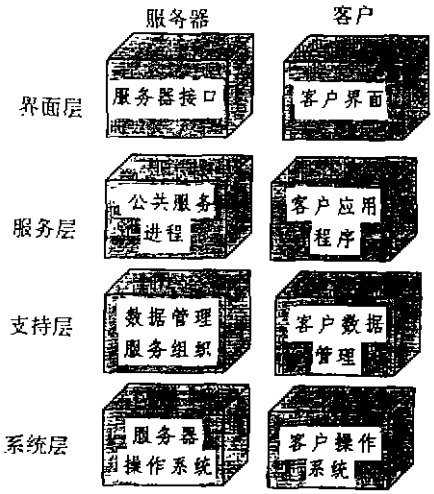

图3 Client/Server系统的软件结构

上述的处理分布问题,主要与服务层的设计与实现有关。要完成整个系统的设计则需要同时考虑与其它各层之间的分离,利用这个模型,可以将不同层次的设计问题突出出来,分别解决,统一集成[2,3]。

3 软件构造途径

在处理分布完成之后,实现服务层软件有两条途径[3,4],一是直接利用Client/Server结构的数据库实现,二是通过网络通讯和操作系统的支持来独立构造。

基于Client/Server数据库的实现方式(DBM)

利用数据库产品在客户和服务器之间分置处理功能,通常将与特定业务有关的功能放在客户机上,通用或共享的处理功能以存储过程或触发器的形式分配到服务器上。利用数据库中的某些智能处理实现服务功能。这些应用程序逻辑上以三种方式出现,即触发程序(TRIGGER)、存贮过程和完整性检查规则。存贮过程是一组编译后存放在服务器数据库端的SQL语句,然后在应用程序中直接对它们进行调用,数据库只在第一次调用时对存贮过程进行词法和语法检查,然后将编译版本放入内存缓冲区。以后的每次调用只需要进行保护级和参数检查后就可以执行,速度很快。存贮过程可以接受参数,嵌套或者调用其它系统上的过程,可以供多种应用程序使用。触发程序是可以自行被服务器数据库软件调用的特

(下转第94页)

求进行分析与抽象，采用定义语言进行半形式化，形成一个软件定义的合一化模型；第二阶段称为演化生成过程，是开发的后期工程，通过变换方法把定义模型转换成可执行的设计模型，获得目标系统原型。

用户可以用一个适当的定义语言来描述和维护软件定义（规格说明），维护仅仅是开发过程的继续，修改后的定义作为待开发系统的原型保证系统与用户需求相一致，最后借助合适的高层构造技术与相关工具将高级的规格说明转换为具体实现。这种由用户自己产生和维护的定义作为开发红线，从根本上改善开发过程。上述体现的基本思想是为所求系统建立一个操作模型。具有四个特点：①建立环境的模型，然后进行系统功能开发；②将面向问题的考虑与面向实现的考虑严格分离；③提供一种操作式的定义语言；④可实现从定义语言到具体实现的转换过程。

在开发范例上，与基于非形式化（图3）的范例及形式化的范例（图4）相比，变换方法的开发范例如图5。

参 考 文 献

[1]G. Arango, et al., TMM: Software Maintenance by Transformation, IEEE Software, May 1986

[2]Robert S. Arnold, Software Reengineering, IEEE Computer Society Press, 1993

[3]R. Balzer, Transformational Implementation: An Example, IEEE Trans. SE-7, (1)1981

[4]V. Berzins, et al., Using Transformations in Specification-Based Prototyping, IEEE Trans. SE-19, (5)1993

[5]E. A. Boiten, et al., How to Produce Correct Software-An Introduction to Formal Specification and Program Development by Transformations, The Computer J., 35 (6), 1992

[6]J. M. Boyle, Program Reusability Through Program Transformation, IEEE Trans. SE-10, (5)1984

[7]R. Burstall, J. Darlington, A Transformation System for Developing Recursive Programs, J. ACM, 24(1), 1977

[8]M. S. Feather, Constructing Specifications by Combining Parallel Elaborations, IEEE SE-15, (2) 1989

[9]W. Lewis Johnson, M. Feathers, Building an Evolution Transformation Library, Proc. 12th ICSE, 1990

[10]P. Kruchten, Software Prototyping Using the SETL Programming Language, IEEE Software, 1(4), 1984

[11]Michael R. Lowry, R. D. McCartney, Automating Software Design, AAAI Press, 1991

[12]C. Rich, R. C. Waters, The Programmer's Apprentice: A Research Overview, IEEE Computer 21(11), 1988

[13]S. Rotenstreich, Transformational Approach to Software Design, Information and Software Technology, Feb, 1992

[14]J. J.-P. Tsai, Intelligent Support for Specifications Transformation, IEEE Software, Nov. 1988

[15]Ying Jing, He Zhijun, et al., A Methodology for High-level Software Specification Construction, ACM Soft. Eng. Notes, 20(2)1995

（上接第86页）

殊的存贮过程，它与某些特定的数据实体相联系，当应用程序修改这些数据表时自动执行，而一般的存贮过程则需要显式调用。完整性检查规则是与某些数据表有关的数据检查约束，在进行数据更新时执行，以保证数据的完整性。

独立构造方式（ICM） 在某些复杂的 Client/Server 系统中，采用 DBM 方式无法满足系统的功能要求，这时需要在操作系统的支持下直接开发服务程序。服务进程可以是一个或一组，但必须有一个进程等待并接受来自客户进程的服务请求，它可以自己亲自服务，也可以启动其它的进程服务。在复杂的系统中 ICM 方式更具有代表性。服务进程也需要精心地组织。在 ICM 方式中可以利用后台进程支持并发服务和优先服务，提供安全检查，监视运行状态，提供异常处理等功能。

结束语 成功的 Client/Server 系统有两个关键因素，一是表达功能与其它应用程序服务功能的分离，二是应用程序处理逻辑在客户和服务器之间的分布。这种分布多数情况下是系统开发者在应用系统设计时决定的。本文围绕 Client/Server 结构应用系统的处理分布过程和系统结构及其实现进行了讨论。随着 Client/Server 结构应用系统复杂和扩大，系统处理的分布问题会日益突出，在功能完备的支持自动处理分布的工具出现之前，处理分布仍然会耗费开发者大量的精力。

参 考 文 献

[1]Sinha A., Client Server Computing, CACM, July 1992

[2]Philippe Kurchten, "4+1" View Model of Software Architecture, IEEE Software, Nov. 1995

[3]Frank J. Van der Linden, Creating Architecture with Building Blocks, Same to [2]

[4]A. Delis, Performance and Scalability of Client/Server Database Architectures, Proc. of the 18th Intl. Conf. on VLDB, 1992

[5]Pieter Mimno, Client/Server Development: State-of-the-Practice, CASE trends, Apr. 1993