基于对象服务的分布式数据库系统排队模型与分析

王治和　扬延娇　张　强
（西北师范大学数学与信息科学学院　兰州 730070）

摘　要　提出了一种基于对象服务的分布式数据库系统的设计方案，介绍了其工作原理和设计思想。根据系统工作模型建立排队模型，并通过对该模型的分析得出了影响系统执行效率的主要因素。利用该方案设计和实现的数据库系统已成功应用到某分布式电力监控系统中。

关键词　对象服务，分布式数据库，排队模型，模型分析

Qeuing Model and Analysis of a Distributed Database System Based on Object Service

WANG Zhi-He　YANG Yan-Jiao　ZHANG Qiang
(College of Mathematics and Information Science, Northwest Normal University, Lanzhou 730070)

Abstract　This paper brings forward a design scheme of distributed database system based on object service and presents its working principle and design thought. On the basis of working model of the system, the queuing model is built. The key factors for influencing performance efficiency are reached by analyzing the model. The database system, which is designed and realized by using the scheme, has been applied successfully in some distributed power monitor system.

Keywords　Object service, Distributed database, Queuing model, Model analysis

1　对象服务的设计思想

分布式数据库的基本要求是对各个分布数据库的透明操作，这就要求客户应用程序发出数据请求时，系统能够对网络上的数据库进行全局事务处理，至于各个数据库在网络上是如何分布的，对客户来说是完全透明的。根据以上要求，结合分布式计算技术和面向对象技术，提出一种对象服务的分布式数据库设计思想，将系统中全局事务处理抽象为代理对象，各个局部数据库也抽象为对象。各个对象提供了一定功能的服务，整个系统运行是这些对象之间的服务请求与服务，并且这些对象本身也可以分布，且由名字管理器关联。

2　系统设计

CORBA 适应分布式技术和面向对象技术的发展，为解决异构环境下的分布式对象间的通信规定了完整的体系结构，实现面向对象分布式计算[1]。笔者基于这种分布式对象技术设计和实现了基于对象服务的分布式数据库系统。

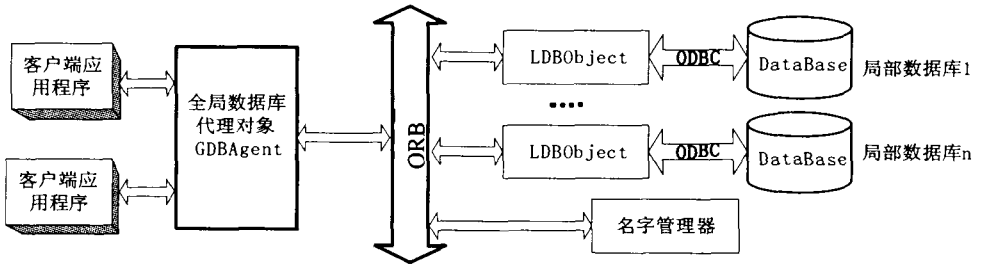

图 1　基于对象服务的分布式数据库系统

系统将每个局部数据库抽象为对象 LDBObject（Local Database Object），LDBObject 提供了对数据库的存取、查询和更新。同时有一个全局数据库代理 GDBAgent（Global Database Agent），客户端应用程序只需要向 GDBAgent 发出数据操作请求，具体的局部数据库对象定位及数据结果整理由 GDBAgent 完成。

GDBAgent 是一个可以独立运行的 CORBA 对象，向用户提供全局数据库模式。每个节点上的数据库都生成一个 LDBObject，LDBObject 中用 IDL 语言定义了对数据库的各种操作。名字管理器负责对象的关联，它是在 CORBA 现有的名字服务的基础上加以修改形成的。在 GDBAgent 中有一个全局事务管理器负责全局事务处理，将全局事务请求转化为多个子事务，按照存储冲突类（事务的操作对象是同一数据库对象的同一冲突类）分别排队向局部数据库对象请求局部事务服务，并由事务管理器中的事务执行器负责全局事务提交和事务恢复，保证分布式事务的 ACID，因为不同节点的同一个数据库都以不同的对象名在名字管理器中注册。事务管理器结构如图 2 所示。

王治和　副教授，硕士生导师，主要研究方向：数据库技术及应用。

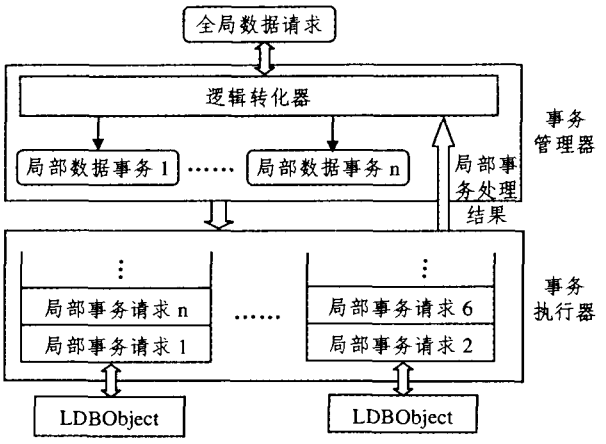

图 2　全局事务管理器

3　系统工作模型

系统工作模型如图 3 所示。

系统初始化时系统中至少运行一个全局数据库代理 GDBAgent 和名字管理器，所有分布的局部数据库抽象为可以访问数据的对象并在名字管理器中注册。具体工作过程为：

(1)客户端应用程序向 GDBAgent 发出全局事务请求，由 GDBAgent 解析为各个子事务，并根据冲突类策略，排成对各局部数据库请求的多个队列，这些队列由 GDBAgent 中的事务管理器负责管理。

(2)事务执行器根据三阶段提交协议，以一个全局事务为单位提交各个子事务，在各自的局部数据库上操作数据库。各个队列中的事务对不同局部数据库的访问是同时进行的。

(3)将各个子事务的操作结果返回到 GDBAgent，GDBAgent 整理并返回给客户端。

根据以上的运行模型可知，这里 GDBAgent 和局部数据库对象都是作为对象，而且分布的数据访问是通过这些对象之间的相互服务来完成的，所以整个系统的运行是对象请求服务和对象服务。

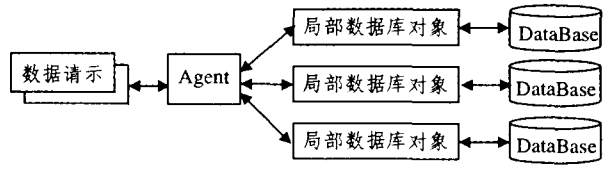

图 3　系统工作模型

4　系统排队模型

根据对本系统的运行模型分析，利用排队理论可以将本系统转化为图 4 所示的排队系统。

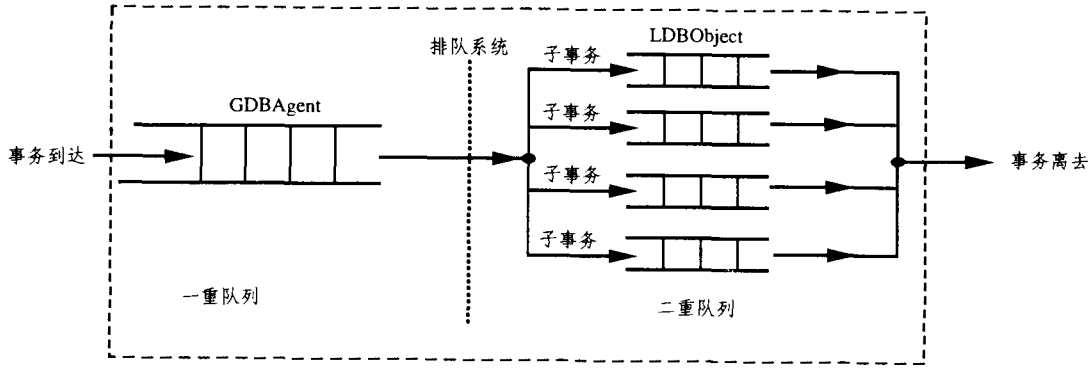

图 4　系统排队模型

系统中共有两重排队，其中第一重排队是客户程序向 GDBAgent 发出的全局数据事务请求队列，第二重是经过 GDBAgent 分解的多个子事务分别向局部数据库对象请求而形成的局部事务请求队列。

4.1　一重队列分析

事务流到达 GDBAgent 符合泊松分布，λ 表示单位时间到达的事务数(到达率)，μ 为 GDBAgent 忙时单位时间内服务的事务数(服务率)，事务到达时间间隔服从参数为 λ 的负指数分布 $a(t)=\lambda e^{-\lambda t}(t\geqslant 0)$。到达事务全部进入服务系统 GDBAgent 接受服务，假设在此接受服务的时间是固定值 T_0，则 $\mu=1/T_0$，那么事务在 Agent 处的停留时间为：

$$W_G = W_S + T_0 \tag{1}$$

其中，W_G：事务在 *GDBAgent* 处的停留时间；W_S：事务在 *GDBAgent* 处的等待时间；T_0：事务在 *GDBAgent* 处接受服务的时间。W_s 的分布函数为：

$$w_S(t)=1-\frac{\lambda}{\mu}e^{-\mu(1-e)t},t>0 \tag{2}$$

平均等待时间为：

$$\overline{W_S}=\frac{\lambda/\mu}{\mu(1-\lambda/\mu)}=\frac{\lambda T_0}{\mu-\lambda} \tag{3}$$

则事务在 *GDBAgent* 处的平均停留时间为：

$$\overline{W_G}=\overline{W_S}+T_0=\frac{\lambda T_0}{\mu-\lambda}+T_0=\frac{1}{\mu-\lambda} \tag{4}$$

事务在一重队列中停留 $\overline{W_G}$ 时间后，事务管理器将事务逻辑分解多个子事务进入二重排队。

4.2　二重排队分析

事务 T 经过 Agent 服务器后被划分为 m 个子事务，而这 m 个子事务根据存储冲突类原则排成 r 个并行队列，并分别向特定的局部数据库对象 LDBObject(N 个)发出服务请求，且 $r\leqslant m$。LDBObject 队列中的到达率为 λ_i，服务率为 μ_i，在第 i 个 LDBObject 队列中子事务接受服务的时间是固定值 T_i，则各个子事务在这个阶段的平均停留时间为：

$$\overline{W_L}=\overline{W_P}+T_i,i=1,2,3,\cdots,r \tag{5}$$

其中，$\overline{W_L}$：子事务在 LDBObject 处的停留时间；$\overline{W_P}$：子事务在 LDBObject 处的等待时间；T_i：子事务在 LDBObject 处接受服

(下转第 212 页)

表1是不同抽象模型在通信模式、计算时间和通信时间等方面的特性比较，由此可清楚知道各层模型的设计特点。

表1 不同抽象模型的特性

抽象模型	通信时间	计算时间	通信模式
CDM模型	Approximate	Approximate	Token
系统级模型	No	No	Port, Signal
行为级模型	No	Approximate	Event, channel
通信模型	Cycle accurate	Approximate	detailed bus chinnel
实现模型	Cycle accurate	Cycle accurate	wire

总结 在现有可供选择的嵌入式系统设计方案中，仍存在对同一任务，不同阶段使用不同的语言描述和工具进行的混合设计方法。在增大系统综合难度的同时，也容易造成系统的不一致。本文提出一个基于5层模型的iCDMdt设计平台，支持嵌入式系统数字产品的高层设计，能进行需求描述、功能建模、行为细化和通道细化，并在设计的每一抽象层次上进行验证。系统设计使用统一的软硬件描述语言，方便了软硬件的综合。利用组件和计算模型，通过逐步精化细节，有根据地调整控制策略后，实现从需求描述到真实系统的优化设计。利用该模型平台进行设计，可提高设计效率，并在多层验证基础上提高设计质量。

参考文献

1 System C. Functional Specification for SystemC2.0. [EB/OL] http://www.systemc.org, 2001
2 Keutzer K, et al. System level design: Orthogonalization of concerns and platform-based design. IEEE Transactions on Computer-Aided Design, Dec. 2000
3 Paulin P, et al. Stepnp: A system-level exploration platform for network processors. In IEEE Design and Test of Computers, Nov-Dec 2002
4 Cesario W, et al. Multiprocessor soc platforms: a component-based design approach. In IEEE Design and Test of Computers, Nov-Dec 2002.
5 Cai L, et al. Comparison of SpecC and SystemC Languages for System Design: [Technical Report CECS-TR-03-11]. UCI, May 2003
6 Boßung W, Huss S A, Klaus S. High-Level Embedded System Specifications Based on Process Activation Conditions. Journal of VLSI Signal Processing, Special Issue on System Design, Kluwer Academic Publishers, 1999, 21(3): 277～291
7 Cai Lukai, Gajski D. Transaction Level Modeling in System Level Design. CECS Technical Report 03-10 Mar 28, 2003
8 陈曦，徐宁仪. SystemC片上系统设计. 科学出版社，2004

(上接第74页)

务的时间。W_P 的分布函数为：

$$W_P(t)=1-\frac{\lambda_i}{\mu_i}e^{-\mu_i(1-\rho_i)t},t>0 \tag{6}$$

平均等待时间为：

$$\overline{W_P}=\frac{\lambda_i/\mu_i}{\mu_i(1-\lambda_i/\mu_i)}=\frac{\lambda_i T_i}{\mu_i-\lambda_i} \tag{7}$$

则子事务在LDBObject的平均停留时间为：

$$\overline{W_L}=\overline{W_p}+T_i=\frac{\lambda_i T_i}{\mu_i-\lambda_i}+T_i=\frac{1}{\mu_i-\lambda_i} \tag{8}$$

由于事务 T 经过GDBAgent后被划分为 m 个子事务，而这 m 个子事务又是根据冲突类原则排成 r 个队列的，因此在每个LDBObject处的子事务个数 $0\leqslant k_i\leqslant m$ 不同，所以事务 T 在LDBObject处的停留时间为：

$$\overline{W_M}=\max\{k_i\ \overline{W_L},i=1,2,3,\cdots,r\} \tag{9}$$

所以一个事务 T 从到达到离开服务系统共用时间为：

$$W=\overline{W_G}+\overline{W_M}=\frac{1}{\mu-\lambda}+\max\{k_i\ \overline{W_L},i=1,2,3,\cdots,r\} \tag{10}$$

5 模型分析

从以上结论可以看出，当GDBAgent在单位时间内处理的事务越多即 μ 越大、单位时间到达的事务数 λ 越小，在GDBAgent处停留的时间 $\overline{W_G}$ 越短。理想状态是一旦事务到达GDBAgent马上获得服务，不需要排队等待，这就要求系统为每个事务分配一个GDBAgent，但这在实际系统中是不现实的，所以整个系统中需要GDBAgent的数目是根据实际系统而定。事务到达后根据所有GDBAgent队列中事务个数，排在当前事务最少的GDBAgent队列中，如果整个系统中的GDBAgent排队事务都足够多时，可以申请系统启动一个新的GDBAgent。

在GDBAgent将事务分解为多个子事务时，这些子事务分别排成多个队列，各个队列中的事务数目 k_i 是不相同的，而事务在二重队列中的逗留时间是所有队列中处理子事务所需要的时间与子事务数目的积的最大值，即 $\overline{W_M}=\max\{k_i\ \overline{W_L},i=1,2,3,\cdots,r\}$。所以在事务分解时，将各个子事务均匀分布在各个LDBObject队列中，且各个LDBObject队列中事务个数最少时 $\overline{W_M}$ 最小。因此，一个全局事务分解后是否平均分布到各个局部数据库对象上将直接影响整个事务处理的效率，理想状态是各个子事务均匀分布到不同的局部数据库对象。同时为了提高整个系统工作效率，可以将多个全局事务请求并发执行，使得各个LDBObject不会出现空闲状态。但这同时又增加了全局事务处理的难度，这个任务只能由事务执行器来完成。

结束语 本文介绍的分布式数据系统基于对象服务的思想设计，将全局事务分解为多个子事务分散到各个局部数据库对象上同时进行，提高了并发执行的速度，并且所有数据库都被抽象为对象，可以实现跨平台异构分布式数据库系统构建。通过对系统的排队模型分析得出了影响系统执行效率的主要因素。所以在实际系统中可以根据需要增加GDBAgent的个数，提高GDBAgent的工作效率，事务执行归类，提高局部数据库对象执行速度等途径来提高系统效率。根据该方案设计和实现的数据库系统已经成功应用到某分布式电力监控系统中。

参考文献

1 [美] Henning M, Vinoski S. Advanced CORBA Programming with C++[M]. 北京：清华大学出版社，2000
2 Silberschatz A. Database System Concepts(Fourth Edition)[M]. 北京：机械工业出版社，2003
3 舒后. 分布式数据库系统的并发控制方法讨论[J]. 微型机与应用，2002，12：14～16
4 田俊峰，王凤先. 一种基于C/S模式的分布式数据库系统及其排队模型[J]. 计算机工程与科学，2002.24(2)：88～91