高可用性软件架构设计和实现

孟剑萍
(中国电子科技集团公司第二十八研究所,江苏 南京 210007)

摘 要：硬件冗余可以极大地提高计算机应用系统的可用性,然而,一旦关键硬件出现故障或数据库宕机,正在进行中的业务流程通常会中断。探讨了一种如何实现应用系统高可用性的软件架构的设计方案,以弥补纯硬件冗余应用系统的不足。

关键词：高可用性;软件容错;分布式数据库

中图分类号：TP31 **文献标识码**：A **文章编号**：1674-7720(2011)17-0019-03

Design and implementation of high availability software architecture

Meng Jianping
(The 28th Research Institute of China Electronics Technology Group Corporation, Nanjing 210007, China)

Abstract: As everyone knows, hardware redundancy can greatly enhance the availability of computer application system. However, once key hardware or database fails, application service will be interrupted. This paper discusses a kind of high available software architecture to solve the problem of hardware redundancy.

Key words: high availability; software fault-tolerant; distributed database

在业内,计算机应用系统的可用性定义为计算机应用系统保持正常运行时间的百分比,通常用表 1 所示的"9"的个数来划分可用性的类型。

表 1 计算机应用系统的可用性定义

可用性分类	可用水平/%	每年停机时间
容错可用性	99.999 9	31.5 s
极高可用性	99.999	5.25 min
具有故障自动恢复能力的可用性	99.99	52.5 min
高可用性	99.9	8 h 45 min
产品可用性	99	3.65 d

通常,硬件冗余(容错计算机、双机或多机集群、磁盘阵列、SAN 等)、数据复制、合理的灾难备份和恢复策略都可以极大地提高计算机应用系统的可用性。正因为如此,当前,对于计算机应用系统的高可用性、业务的可持续性要求,业内通常以硬件系统的高可用性来应对或代替。常见的解决方案是双机(或多机)集群方案或直接采用容错计算机来保障系统的高可用性,应用软件的设计和开发往往仅注重业务流程的分析和过程控制。在这种完全依赖硬件来保障整个系统的可用性的系统里,一旦关键硬件出现故障或数据库宕机,正在进行中的业务流程(如需较长执行时间的事务处理、后台批处理过程等)必然会中断,这是因为双机切换也需要时间。对此,应用软件本身并无多少作为,该类业务必须等待系统重新恢复后全部或部分重做。

本文以基于大型数据库的应用系统为例,从"软件容错"设计的概念出发,参考"分布式"数据库结构设计,以"系统服务总线"为核心,给出了一种可行的高可用性软件架构的设计方案,可以极大地提高应用软件的可用性和业务系统的可持续性。无论是传统的 C/S 架构,还是近年来流行的 B/S 架构,本文中给出的设计方案都有一定的参考意义。

1 软件结构模型

任何基于大型数据库的应用系统,都可以抽象为对数据的"读"和"写"操作。至于客户端如何展现"读"到的数据,以及"客户端"与"服务端"基于何种通信协议通信,不在本文讨论之列。

软件结构的设计其实就是针对"读"和"写"的一系列流程的设计。如何最大限度地保证系统中的所有"硬件"和"软件"协同工作,正确完成每一次"读"和"写"的操作,也就是对系统"高可靠性"和"高可用性"的要求。

图 1 是基于“软件容错”和“分布式数据库系统”的原理，并参照了计算机“总线”的工作原理给出的一种基于分布式数据库或文件系统的高可用性的软件架构设计方案。系统采用 3 层架构：客户端、中间应用层和数据库层。

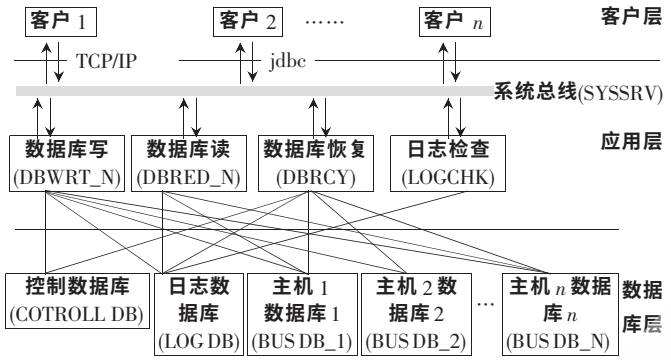

图 1 软件结构模型

2 系统设计

2.1 数据库配置

为了更清楚地阐述本文的设计方案，先对数据库的配置及其功能进行描述。本系统中，数据库按角色可划分为如下三类数据库：控制数据库(COTROLL DB)、日志数据库(LOG DB)、业务数据库(BUS DB_N)。

2.1.1 控制数据库

控制数据库也可以是一个或多个系统控制(参数)文件。它存放要访问的目标数据库的节点(N)、端口、用户、文件头、表、视图等信息；存放对节点、业务数据库、表或视图的授权或访问控制信息；目标数据库(或文件)的当前状态(联机/脱机、忙/空闲等)；目标数据库中的表或视图的当前状态(联机/脱机、忙/空闲、加锁/解锁等)。

2.1.2 日志数据库

日志数据库独立于业务数据库之外，用于记录客户端节点信息、请求时刻和发来的所有请求的原始内容，但不做业务流程相关的处理、运算等。记录每次数据操作分配的唯一的“事件号”(EVENT_ID)。对每一次客户端的“请求”，“系统服务总线”(SYSSRV)会分配唯一的标识符号，可以定义为有一定意义的字符串，比如，“当前时刻+流水号”。以上信息可以被压缩、打包、加密后存放，以记录格式保存于数据库的表或文件中。它可以设计为数据库中的一个或多个表，也可以是文件格式。

2.1.3 业务数据库

业务数据库记录所有业务相关的数据信息。所有业务数据库的相关业务逻辑的数据结构相同，即，N 个节点的业务数据库中与业务模式相关的表、视图、过程或其他程序设置相同。

需要特别指出的是：

(1)控制数据库、日志数据库和业务数据库可以是不同数据库厂家或品牌的产品。比如，日志数据库可以采用低端的数据库产品或开源数据库系统，业务数据库可以采用高端的大型数据库产品。

(2)控制数据库、日志数据库和业务数据库在物理上和逻辑上是可以相互隔离的，这可以极大地提高系统的整体安全性。目标数据库和要访问的表或视图对客户端来说是“不可见”的，由控制数据库动态定义和控制。

(3) 所有类别的数据库在物理上位于一个或多个节点上，即节点 N>=1；任意一个节点 N 上建有一个或多个业务数据库(逻辑数据库>=1)；任意一个节点是一个完整的、可独立工作的计算机。根据性能要求，可以是高性能 PC 机、PC 服务器、小型机、集群或超级计算机，或是它们的“混合体”；任意一个节点是指定网络中的一个指定节点。

2.2 应用层设计

中间应用层由 5 个后台进程构成：(1)系统服务总线(SYSSRV)；(2)数据库写进程(DBWRT_N)；(3)数据库读进程(DBRED_N)；(4)数据库在线恢复进程(DBRCY)；(5)日志检查进程(LOGCHK)。

2.2.1 系统服务总线

这是一个后台监听、分发、调度总进程。设计目标具有一定的“自我修复”和“自我复制”动能。它可以根据负载情况，自我复制或开启子进程响应新的负载；可以动态配置可服务的节点或客户端；可以为特定节点或客户端指定专用进程；它通过“DBWRT”和“DBRED”“读/写”日志数据库或日志文件。

2.2.2 写进程

写进程负责向所有节点写数据。它可以配置成多进程/单进程模式；多进程模式，指对应每个业务数据库 N 都有独立的“写”进程；单进程模式，指对应多个业务数据库只有一个主进程，主进程开启多个线程提供“写”服务。

2.2.3 读进程

读进程负责向所有节点读数据，它可以配置成多进程/单进程模式。多进程模式指对应每个业务数据库 N 都有独立的“读”进程，单进程模式指对应多个业务数据库只有一个主进程，主进程开启多个线程提供“读”服务。

根据需要，读进程可以配置成：向所有在线节点并发读数据，返回最快的结果集，抛弃其他的结果集，并中断其他读进程；也可以配置成：随机读某个节点的数据，如果失败或超时，则再随机读余下的在线节点，直到“读”成功或失败；还可以配置成向所有节点顺序读数据，过程类似上面“随机读”。

以上“读写”业务数据库的进程，设计上支持多种数据库访问接口，针对“表”或“视图”提供统一格式的、标准的、动态的 SQL 数据操作接口和方法，完成对数据库中表或视图的增、删、改、查和批处理操作。它们可以设计为数据库中的存储过程，也可以是 C++，Java 程序的 API 或混合体。

2.2.4 数据库在线恢复进程

该进程负责检查全部或部分节点数据库（包括所有授权控制数据库、业务数据库和日志数据库）或文件的工作状态；检查数据库或文件表中数据的一致性；将以上检查结果写入日志数据库（或日志文件）。

当某个业务数据库中的表写入失败时，它负责从“日志数据库”的表或日志文件中读出原始数据，接着写入出现问题的业务数据库的表中，并检查结果。或从其他节点的数据库中读相关数据并写入到出现问题的业务数据库的表中。

接收外部命令，根据“时间点”或“事件号”从特定时刻、特定数据库（包括日志数据库）、特定表恢复数据到特定目标数据库的表或文件。

2.2.5 日志检查进程

该进程负责读、写日志文件，检查数据操作结果的一致性。如果不一致，则报告给“系统服务总线”，将问题数据库或数据库中的表、视图设置为“离线”状态。

3 系统实现

3.1 系统初始化

启动配置好的后台进程即完成系统初始化过程。

3.2 数据“写”流程

数据“写”流程的主要步骤如下：

(1) 客户端通过给定协议（或混合多种通信协议）向后台“系统服务总线”发送“写”请求。

(2)激活“数据库写进程”，将客户端的“请求”写入“日志数据库”（或日志文件），并分配一个唯一的“事件号”。

(3)“系统服务总线”查询“授权/控制数据库”（或/配置文件）得到客户端请求访问的数据存放的目标数据库（或文件）节点 N（或文件存放的节点 N）、端口、用户、表、文件头等信息。节点 N 可以是多个，即节点 N>=1。

(4)“系统服务总线”向 N 个“数据库写进程”发送数据“写”访问请求，并得到各节点的返回结果集。

(5)只要有 1 个节点写入成功，“系统服务总线”就将写入成功的标志发回客户端；“数据库写进程”将各节点的返回结果状态写入“日志数据库”（或日志文件）中。

(6)“日志监控”查询“日志数据库”（或日志文件），比较 N 个节点的写入状态。如发现写错误、失败、超时等状态，则将该“业务数据库”（或文件、表、视图）标志为“非正常联机数据库”（或文件、表、视图不可用）。

(7)激活“数据在线恢复进程”，进程为“非正常联机数据库”，则执行数据库数据“同步”。在线同步恢复如失败，则将该“数据库”标志为“需要 DBA 维护”的类别，留待 DBA 或软件维护工程师处理。

3.3 数据“读”流程

数据“读”流程的主要步骤如下：

(1) 客户端通过给定协议（或混合多种通信协议）向后台“系统服务总线”发送“读”请求。

(2)激活“写进程”，将客户端的“请求”写入“日志数据库”（或日志文件），并分配一个唯一的“事件号”。

(3)“系统服务总线”查询“授权/控制数据库”（或/配置文件）得到客户端请求访问的数据存放的目标数据库节点 N（或文件存放的节点 N）、端口、用户、表等信息。节点 N 可以是多点，即节点 N>=1。

(4)“系统服务总线”查询“授权/控制数据库”（或/配置文件）得到可用的、空闲的目标数据库节点 N（或文件存放的节点 N）。

(5)激活“读进程”（或随机、或顺序）向 N 个节点的“业务数据库”（或文件）发送数据“读”访问请求，并得到各节点的返回结果集。

(6)“系统服务总线”将最快返回的结果集发回客户端；抛弃其他结果集，中断其他读进程。

在本系统的设计和实现中，由于采用了“分布式”数据库或文件系统部署，只要 N 个节点中至少有一个节点的“业务数据库”正常工作，因为一个或几个“业务数据库”系统（或节点硬件）故障所引起的业务系统的不可持续性理论上将可以完全避免，因而提高了系统的“容错”性。

由于 N 个数据库同时在线，且节点是否可用、空闲等状态可实时监控，这为特定业务快速访问和独享访问提供了先决条件。如可以指定某特定“业务数据库”仅为某个或几个特定客户端服务提供“读”访问。

因为设计了统一、标准的增、删、改、查的过程方法或 API，前端开发人员甚至不必写任何 SQL 语句就可以完成对数据库中表或视图的操作，可以大大地缩短编程和调试时间。

需要指出的是，虽然“系统服务总线”具有“自我修复”和“自我复制”的特点，但因为“节点”硬件故障或“授权/控制数据库”（或/配置文件）或“日志数据库”故障而引起的全系统不可用依然存在，因此，建议该节点采用性能好、可靠性高的中、高端服务器。

参考文献

[1] Mostofa Abd-El-Barr. Design and analysis of reliable and fault-tolerant computer systems[M]. London: Imperial College Press, 2006.

[2] KOREN I, KRISHNA C M. Fault-tolerant systems [M]. Morgan Kaufmann Publishers, 2007.

[3] Hoang Pham. Handbook of reliability engineering [M]. Springer, 2003.

[4] MARCUS E, STERN H. Blueprints for high availability. 2nd, edition. John wiley&Sons, 2003.

（收稿日期：2011-06-14）

作者简介：

孟剑萍，女，1963 年生，研究员级高工，主要研究方向：综合信息系统设计与开发。