第 25 卷 第 3 期
2010 年 6 月

天津科技大学学报
Journal of Tianjin University of Science & Technology

Vol.25 No. 3
Jun. 2010

基于 BerkeleyDB 的嵌入式实时数据库研究

黄建岗，李孝忠
（天津科技大学计算机科学与信息工程学院，天津 300222）

摘 要：针对嵌入式实时数据库对数据库系统结构的要求，提出一种基于 BerkeleyDB 数据库引擎、运行于 VxWorks 平台的 ERTDB 系统结构. 在模型中引入 Shell 层，事务管理器获得事务命令后将处理转入 Shell 层，在调用 BerkeleyDB 相应的函数前 Shell 层与并发控制器进行同步通信，由并发控制器确定该事务命令是否可以执行. 对于并发控制器，通过事件驱动的方式来实现并发控制协议，协议结构体的设计可以更方便地使用不同的并发协议；采用定时地根据正在运行的事务实时性和截止期为各事务动态分配 VxWorks 下的任务优先级的机制实现时限控制器功能. 实验表明，该系统的读写速度可以满足千万级的数据处理要求.

关键词：嵌入式；实时数据库；BerkeleyDB；VxWorks
中图分类号：TP392 **文献标志码**：A **文章编号**：1672-6510(2010)03-0066-05

Research on Embeded Real-Time Database Based on BerkeleyDB

HUANG Jian-gang，LI Xiao-zhong
(College of Computer Science and Information Engineering，Tianjin University of Science & Technology，
Tianjin 300222，China)

Abstract：In consideration of embedded real-time database's requirement on database system structure，the embedded real-time database(ERTDB) structure running on VxWorks platform and BerkeleyDB was proposed. First，shell layer was added. The transaction manager handed on the results to Shell layer when it got the transactions. Before calling the BDB's functions，the synchronization communication between Shell layer and concurrency control manager was completed，and the concurrency control manager determined whether the transactions could be executed. For the concurrency control manager，the concurrency control protocol was realized through the event-driven approaches. The design of protocol structure makes it more convenient to use different concurrency protocols. Each transaction was dynamically assigned VxWorks's task priority according to the running real-time transaction，which realized the timer controller functions. The test result shows that the system's read-write speed is fully able to meet the 10 million data-processing requirements.

Keywords：embedded；real-time database；BerkeleyDB；VxWorks

传统的数据库系统包括层次数据库、关系数据库系统等，都旨在处理永久、稳定的数据. 其性能目标是高的系统吞吐量和低的代价，不考虑有关数据及其处理的定时限制. 所以，传统的数据库管理系统(DBMS)不能满足实时应用的需要，只有将数据库技术和实时系统的概念、技术、方法与机制“无缝集成”的实时数据库(RTDB)才能同时支持定时和一致性[1]. 因此，研究 RTDB 技术并实现 ERTDB 系统原型是非常必要的.

目前各软件开发机构和技术研究组织都开发了具有某些 RTDB 特性的产品，各有优劣. 如 Microsoft 公司的 SQL Server CE、Oracle 公司的 Oracle Lite 10g 等 C/S 关系型系统. 它们对于嵌入式系统开发者非常友好，因为程序员已经熟悉了 SQL 和关系数据库设计. 其缺点是，这些产品对于嵌入式系统来说依然过于庞大，应用这些产品增加了部署、维护的复杂

收稿日期：2009–09–05；**修回日期**：2010–03–04
基金项目：天津科技大学自然科学基金资助项目(20070223；20080208)
作者简介：黄建岗（1975—），男，河北武安人，讲师，huangjg@tust.edu.cn.

度. 华中科技大学开发的 ART-DB 作为 RTDB 领域新技术的原型，要求和底层的 ART-OS 紧密结合，开发支持较少，可靠性也逊于商用的系统.

BerkeleyDB（以下简称 BDB）是一个源码开放的、为应用系统提供可伸缩、高性能、事务管理支持的嵌入式数据库引擎，能够为应用程序提供高性能的数据管理服务[2]. 其作为嵌入式数据库在许多方面有着独特的优势，非常适合资源匮乏的嵌入式系统.

本文介绍一种基于 BerkeleyDB 的 ERTDB 原型系统，重点阐述该系统在 ERTDB 方面解决的技术难题和其特点.

1　ERTDB 事务处理系统模型及系统改进

1.1　ERTDB 事务处理系统模型

根据 ERTDB 的时间约束、空间约束、高可靠性等特点，借鉴传统数据库的体系结构提出了如图 1 所示的 ERTDB 事务处理模型（没有包括存储器管理部分），它是运行于 VxWorks 平台的 RTDB 引擎的基础. 该事务处理系统由事务接收器、事务等待队列、时限控制器、事务管理器、并发控制器和系统全局空间构成，各部分协同合作共同完成事务管理、动作派发功能. 模型和底层的操作系统紧密联系，因此该模型在考虑充分利用现有 RTDB 理论的同时也应用了 VxWorks 操作系统提供的实时特性[3–4]. 它采用了原子数据模型，并在这个数据模型上提供了基本的数据访问接口.

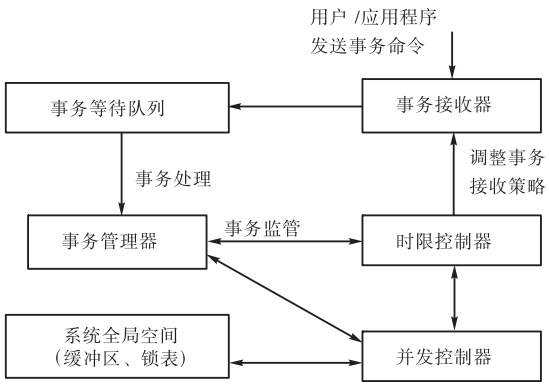

图 1　RTDB 事务处理系统模型

Fig.1　Real-time database transaction processing system model

1.2　VxWorks

VxWorks 是由美国的 WindRiver System 公司开发的一种功能强大的高性能实时操作系统，VxWorks 系统运行环境支持的 CPU 包括：PowerPC、68K、CPU32、SPARC、i960、x86、Mips 等；系统同时支持 RISC 和 DSP 技术. 微内核 Wind 是一个具有较高性能、标准的嵌入式实时操作系统内核，其主要特点包括：快速多任务切换、抢占式任务调度、任务间通信手段多样化等；它包括多任务调度（采用给予优先级的抢占方式），任务间的同步和进程间通信机制以及中断处理、看门狗和内存管理机制.

2　BDB 作为 ERTDB 的不足及改进

2.1　不足之处

BDB 缺少事务接收器、事务等待队列、时限控制器和并发控制器，但更为重要的是在事务处理的方式上 BDB 与预期的结构有很大的出入. BDB 没有事务接收的机制，但是在提供充足系统信息的前提下很容易在 BDB 代码外添加这样的功能. 另外一个缺乏的功能部件——时限控制器则复杂的多，因为时限控制器需要与并发控制和事务管理器合作，受 BDB 内部实现的影响.

2.2　改进

可以把 BDB 整体看作一个复杂的对象，根据 Erich Gamma 的总结[5]，可以通过继承这个对象的类或者修饰这个对象的类来改变对象的行为. 因此采取继承的思路可以对 BDB 的代码作内部修改使其行为完全符合上面提到的数据库结构；也可以采用外部修饰的方法，通过实现一个 Shell 层封装 BDB 内部的函数，使其暴露给外界的行为符合 ERTDB 的要求. 这两种方法各有优势，前者可以让熟悉 BDB 接口的开发人员很快地在新系统上工作，方便系统移植，避免了重新设计一套接口带来的风险，但是直接修改内部代码难度较大. 外部修饰法的优点也很明显：可以最大程度利用原来代码的成熟和可靠性. 比如著名的 Web 服务器 Apache、SUN 公司的轻量目录服务器软件（LDAP）就都是依靠 BDB 来完成记录数据的[6].

由上综述，本文采用从外部修饰的方法实现一个符合 ERTDB 结构的原型系统如图 2 所示. 图中事务接收器、并发控制器、时限控制器和事务管理器都是独立运行的任务，而共享系统空间是内存中的一块区域. 根据它们实现方式的不同分为两类，一类是建立在 BDB 原有代码基础上的，如共享系统空间和事务管理器；另一类是新增到系统中的，如事务接收器、并发控制器以及时限控制器. 事务管理器封装了 BDB 原来的事务代码，使其在并发控制器的协调下工作.

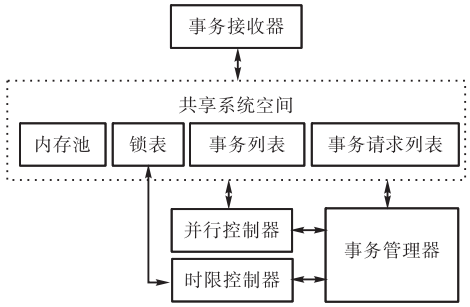

图 2　原型系统结构

Fig.2　Structure of the prototype system

系统的运行方式如下：启动后首先将所有的任务初始化，事务管理器的数目可以根据应用的不同需要有所不同；然后建立共享系统空间；事务接收器开始服务，将接收的事务请求置入事务等待队列；并发控制器为空闲的事务管理器挑选合适的事务；时限控制器则监控系统的状态并且根据调度策略调整各事务的优先级.

图 3 为引入 Shell 层的事务管理器模型. 在此模型下，每当事务管理器的执行线索在进入或者离开 BDB 代码时，Shell 都会通知并发控制器. 由于 Shell 的存在，事务管理器可以根据并发控制器的命令控制事务的动作，比如当实时性不同的事务发生冲突时，事务管理器可以将低实时性的事务挂起、回滚甚至终止保证高实时事务的执行. 由此，BDB 事务处理对于实时性考虑的缺乏得以弥补.

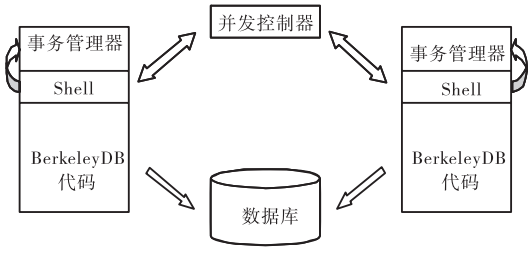

图 3　引入 Shell 层后的并发事务处理

Fig.3　Concurrent transactions with Shell layer

2.2.1　事务管理器的实现

事务管理器作为一个 VxWorks 中独立的任务运行. 事务管理器启动时，没有分配到事务，此时它的优先级可以是任意的. 启动完成后，事务管理器向并发控制器发送就绪消息，并且等待事务分配，为了节约处理器资源，此时事务管理器挂起. 当并发控制器从共享系统空间中的事务请求列表中获得新的事务并指派后，事务管理器恢复执行，并且根据事务的实时性设定自身在 VxWorks 系统中的优先级，在并发控制器的范围之外，即所有的事务都能获得所需的资源的情况下，各事务之间的调度由 VxWorks 系统承担，这样高优先级的事务可以抢占到更多的处理机时间. 事务管理器获得事务命令后将处理转入 Shell 层，在调用 BDB 相应的函数前 Shell 层与并发控制进行同步通信，由并发控制器确定该事务命令是否可以执行. 如果无法执行，事务管理器将挂起直到被并发控制器唤醒. 在确认可以执行后，Shell 层调用 BDB 相应的函数. 当执行线索返回后，Shell 层再次与并发控制器同步通信，成功执行则重复执行下一条命令，如果执行失败则根据并发控制器返回的调度命令将该事务回滚、重启甚至中止. 事务完成提交后，事务管理器又返回到空闲的状态.

Shell 层是事务管理器与并发控制器之间的接口，事务管理器进入 Shell 层获得并发控制器的命令如同操作系统中系统由用户态进入内核态进行进程/任务调度一样. 它的实现过程就参照了操作系统的状态切换. Shell 层和并发控制器之间没有硬件的支持，只能通过编程来实现，本文利用了 VxWorks 提供消息队列(Message queues)的通信机制. 每个事务管理器在启动时建立一个消息队列，在就绪和运行状态时事务管理器不受并发控制器的调度，此时的消息队列没有任何消息；进入 Shell 层后，事务管理器向并发控制器请求调度，接着通过这个消息队列等待接收来自并发控制器的调度命令，由于在此之前消息队列是空的，因此事务管理器被 VxWorks 系统阻塞直到消息队列中收到了并发控制命令. Shell 的结构代码如下：

```
void Shell(Command Todo)
{
int ret=0;
/*请求并发控制命令*/
InformConcurBef(Todo);
/*等待消息*/
 msQReceive(MsgQJd, sgBuf, MAX _ MSG _
LEN, WAIT _ FOREVER);
/*处理接收到的并发控制命令*/
Handle Concur(sgBuf);
/*调用 BerkeleyDB 相关的函数*/
ret=Dispatch(Todo)
/*请求并发控制命令*/
InformConcurAft(Todo, ret);
/*等待消息*/
msQReceive(MsgQJd, sgBuf, MAX _ MSG _
LEN, WAIT _ FOREVER);
/*处理接收到的并发控制命令*/
```

```
Handle Concur(sgBuf);
}
```

2.2.2　并发控制器的实现

事务管理器和并发控制器主要依靠消息队列通信，因此，并发控制器的主要工作就是处理来自系统中所有事务管理器的消息，然后根据并发控制协议返回控制命令. 由于这个原型系统的目标是能够灵活地使用当前各种主要的 ERTDB 并发控制技术，因此，并发控制器的控制命令及结构设计必须要适应不同技术的要求. 目前各种并发控制协议为解决冲突采取各种不同的解决方法：阻塞、回滚、重启和中止. 根据这些冲突处理方法，事务管理器需要执行的并发控制命令有：NewJob、SetPriority、Go、Wakeup、Suspend、Rollback、RollSus、Retry 和 Abort，这些命令到达事务管理器的消息队列后又有不同的处理方式.

并发控制协议对于并发事务冲突的探测和解决发生在这些冲突发生以及事务操作的事件中，由于事务管理器的 Shell 层在执行事务命令的前后都会将控制权交给并发控制器，因此，在这事件发生的时候进行并发控制协议处理是非常合适的. 本原型系统就是通过事件驱动的方式来实现并发控制协议的，为了便于使用不同的并发协议定义了一个结构 rb_protocol.

```
stuct _ rb _ protocol; typedef _ rb _ protocol  rb _ protocol;
stuct _ rb _ protocol
{
void(*on _ protocol _ init)();
COMMAND(*on _ txn _ begin)(DB _ ENV  *env, DB _ TXN  *parent, DB _ TXN  **tid, u _ int32 _ t flags);
COMMAND(*on _ txn _ abort)(DB _ TXN *tid)
COMMAND(*on _ txn _conmmit)(DB _ TXN *tid, u _ int32 _ t flags);
COMMAND(*on _ txn _discard)(DB _ TXN *tid, u _ int32 _ t flags);
COMMAND(*on _ c _get _ in)(DB _ TXN  *tid, DBT *key, DBT *data, u _ int32 _ t flags);
COMMAND(*on _ c _get _ out)(DB _ TXN *tid, DBT *key, DBT *data, u _ int32 _ t flags);
COMMAND(*on _ c _get _ in)(DB _ TXN  *tid, DBT *key, DBT *data, u _ int32 _ t flags);
COMMAND(*on _ c _get _ out)(DB _ TXN *tid, DBT *key, DBT *data, u _ int32 _ t flags);
        ……
}
```

这个结构的成员都是函数指针，它们指向事务操作事件对应的并发协议处理函数，当事务管理器的 Shell 层将控制权交给并发控制器后，并发控制器根据消息中的事件描述调用这个结构中的成员，这样只要更换这个结构中成员指向的地址就可以方便地使用不同的并发协议. 如 on_c_put_in 是事务管理器的 Shell 层在执行一个写操作的动作前触发的事件. 并发控制器的伪代码如下：

```
void ConcurController(rb _protocol protocol)
{while(数据库运行)
{
获取事务管理器的请求信息;
Switch(请求消息)
{
case 事件一：调用 protocol 中相应的函数;
case 事件一：调用 protocol 中相应的函数;
}}}
```

2.2.3　时限控制器的实现

本原型系统中的时限控制器的实现机制采用定时地根据正在运行的事务实时性和截止期动态为各事务分配 VxWorks 下的任务优先级. 与并发控制器一样，这里定时执行的优先级分配函数也可以定制，适用不同的调度策略.

2.2.4　系统各任务优先级的分配

对所有任务按照优先级进行排队，高优先级任务优先处理. 在终端提交每一个任务请求时，系统将该任务前面所有高优先级事务所需要的执行时间返回给终端，也即告知终端本任务将等待的时间，如果等待时间大于终端所要求的时间，将按照终端指定的超时处理策略，启动超时处理程序. 在优先级排队过程中，可以采用多种实时调度算法，例如典型的静态调度算法——速率单调调度算法(RMSA)[7]，或截止期最早最优先算法 EDF(Earliest Deadline First)[8]和最小松弛度优先算法 MLF (Minimum Laxity First)[8]等动态调度算法.

3　实验结果

硬件平台采用了 Inte1 Pentium 系列的 PC104 CPU 板，在 VxWorks 6.0 操作系统和 Tornado 2.2 编译环境下调试开发. 采用支持 VxWorks 的 BDB 嵌入式数据库最新版本 Berkeley DB 4.7.25.

BDB 运行在单独的目标机上，利用提供的 API 接口实现数据的写入和读出. 利用在Windows 系统的客户端应用程序访问测试数据库系统，连续向系统写

入100多万条的测试数据，测试结果如图 4 所示.

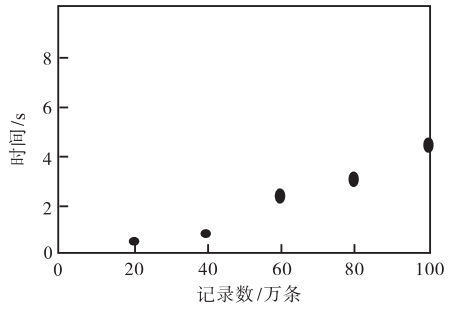

图 4　写记录时间

Fig.4　Time of inserting data

可以看出，系统每秒可以插入约 20 万条记录，而读取速度还要稍快一点，对于千万级的数据处理，其性能没有问题.

4　结　语

本文在 BDB 代码的基础上实现了一个 ERTDB 原型，实现的过程解决了两个关键技术问题：使用 Shell 层改变 BDB 的事务处理模型，使其符合 ERTDB 结构的要求；采用灵活的结构实现并发控制器，使其可以方便地使用不同的并发控制协议.

参考文献：

[1] 王琳，喻成，李昌一. 实时数据库的现状与发展[J]. 河北理工学院学报，2003(4)：93–99.

[2] 刘巍巍，徐成，李仁发. 嵌入式数据库 Berkeley DB 的原理与应用[J]. 科学技术与工程，2005，5(2)：86–90.

[3] 李任，曹万华. 一种基于 VxWorks 平台的实时数据库事务处理模型[J]. 计算机与数字工程，2005，33(4)：96–98.

[4] 阎月. 基于 VxWorks 嵌入式数据库管理系统的设计思路[J]. 辽宁师专学报：自然科学版，2007，9(4)：33–35.

[5] 李英军，马晓星，蔡敏，等. 设计模式——可复用面向对象软件的基础[M]. 北京：机械工业出版社，2007.

[6] 黄昂. 基于 VxWorks 的水下机器人嵌入式数据库开发[D]. 哈尔滨：哈尔滨工业大学，2007.

[7] Liu C L，Layland J. Scheduling algorithms for multiprogramming in a hard-real-time environment[J]. Journal of ACM，1973，20(1)：46–6l.

[8] Leung J Y T，Whitehead J. On the complexity of fixed-priority scheduling of periodic，real-time tasks[J]. Performance Evaluation，1982，2(4)：237–250.

（上接第 60 页）

参考文献：

[1] 周学斌，杨学友，杨楠，等. 立体视觉传感器优化设计技术研究[J]. 计算机测量与控制，2007，15(6)：831–832.

[2] 孙凤梅，黄凤荣，胡占义，等. 视觉测量误差的研究与讨论[J]. 北方工业大学学报，2005，17(1)：61–63.

[3] 吴彰良，卢荣胜，胡鹏浩，等. 双目立体视觉传感器精度分析与参数设计[J]. 郑州轻工业学院学报：自然科学版，2006，21(3)：32–34.

[4] 王建华，韩红艳，王春平，等. CCD 双目立体视觉测量系统的理论研究[J]. 电光与控控制，2007，14(4)：94–96.

[5] 张蕾，徐镜波，杨丽. 析因试验设计在环境污染物联合毒性研究中的应用[J]. 干旱环境监测，2004，18(1)：20–22.

[6] 李伟，朱红. 均匀实验设计在驱油剂配方研制上的应用[J]. 北京交通大学学报：自然科学版，2008，32(3)：67–69.

[7] 娄国强，吕文彦. 回归最优设计在麦田杂草防除中的应用[J]. 安徽农业科学，2006，34(12)：2766–2768.

[8] 刘红波，陆刚，边宽. 几种实验设计方法的比较[J]. 安徽农业科学，2007，35(36)：11738–11739.

[9] 吕桂善，杨志伟，杨正兴. 最优回归设计在食品加工工业中的应用[J]. 广西大学学报：自然科学版，2001，26(2)：125–126.

[10] 罗宇锋，范耀祖，富立. 正交设计及 D 最优设计在陀螺测试中的应用[J]. 压电与声光，2008，30(4)：393–394.

[11] Montgomery D C. Design and Analysis of Experiments[M]. 6th ed. New York：John Wiley& Sons Inc，2004.

[12] 齐力旺，韩一凡，李玲. 应用 311–A 最优回归设计研究 ABA、PEG4000 及 $AgNO_3$ 对落叶松体细胞胚发生数量的影响[J]. 生物工程学报，2001，17(1)：84–86.

[13] 张雅文，陈梁俊. 最优回归设计在落叶松遗传转化体系主导因子优化筛选试验中的应用[J]. 数学的实践与认识，2004，34(1)：99–100.