

DTS 中实时内存数据库的构建 及匿名管道进程通信

常鲜戎^{1,2}, 张帅民^{1,2}, 姜志宇^{1,2}

(1. 华北电力大学 电力系统保护与动态安全监控教育部重点实验室,
河北 保定 071003; 2. 华北电力大学 电气工程学院, 河北 保定 071003)

摘要: 针对调度员仿真培训系统 DTS (Dispatcher Training Simulator) 对数据库快速传输数据的要求, 结合 DTS 中访问数据库的特点及进程间通信的特征提出一种结合内存数据库与对象数据库构建实时内存数据库的方法。该实时内存数据库的优点是读/写速度快、无操作冗余、根据给定的条件主动向用户传输数据。针对所构建的实时内存数据库, 提出了使用匿名管道方法进行计算进程与实时内存数据库进程间的通信 (IPC), 同时, 对比了共享内存和套接字方法, 给出了部分程序代码及对比试验数据。开发实践证明效果良好。

关键词: DTS; 实时内存数据库; 匿名管道; 进程间通信; 实时性; 共享内存

中图分类号: TM 73

文献标识码: B

文章编号: 1006-6047(2006)03-0061-05

0 引言

调度员培训仿真系统^[1-2] DTS (Dispatcher Training Simulator) 是利用先进的计算机技术, 模拟实际电网的运行、用于调度员培训的数字仿真系统^[3]。一个功能完备的 DTS 需要处理大量数据, 并支持局域网内的多用户访问, 因此, 技术成熟的关系数据库成为首选。但是, 随着 DTS 研究的深入发展, 传统的关系型数据库暴露出许多问题, 并严重制约 DTS 的发展。目前, 许多 DTS 不具备系统稳定计算功能, 这不仅是因为稳定计算本身需要占用较多时间, 而且更是由于传统关系型数据库的性能无法满足实时性要求。用 DTS 模拟电网运行情况, 尤其是模拟故障后系统动态现象要求尽量减少数据库操作占用时间, 以满足电网动态过程的实时仿真。本文分析了传统关系型数据库存在的 I/O 瓶颈和大量重复查询等在 DTS 应用中出现的特有问题, 随着现代数据库技术的发展, 内存数据库、实时数据库、主动数据库^[4]等技术得到广泛研究, 合理运用这些技术可很好地满足 DTS 的实时性要求, 改善 DTS 数据库性能。本文将 DTS 中用到的数据库分为 2 部分: 一部分是离线库, 采用技术成熟的关系数据库 Oracle 数据库; 另一部分是结合内存数据库和对象数据库构建的实时内存数据库。该实时内存数据库具备主动输送数据的特性, 尽可能减少 I/O 时间和 I/O 次数、优化查询、避免重复查询等方法提高数据传输速度, 满足实时计算要求。

提出的主动型实时内存数据库以 3 台 PC 机组成的局域网为硬件基础; 以 Windows 操作系统、大型

关系型数据库 Oracle, TCP/IP 网络通信协议为软件平台, 使用 VC++6.0 开发。其中网络通信使用套接字实现。而服务器上计算进程与实时内存数据库进程间的数据传输采用匿名管道的方法实现, 此方法的突出优点是传输数据速度快、可靠性高、CPU 占用率低并且节省 CPU 时间。

1 传统关系型数据库制约 DTS 的发展

传统关系型数据库在 DTS 的应用中存在着许多缺陷, 如实时性不好, 查询效率低, 并且只能是“被动的”, 不能根据用户需要主动的传输数据。下面简要介绍传统关系型数据库的缺点。

1.1 I/O 瓶颈

首先, 分析 DTS 中数据库的工作情况, DTS 中数据库的使用如图 1 所示。

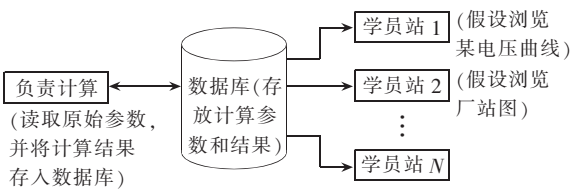


图 1 DTS 中数据库的使用
Fig.1 Database usages in DTS

从图 1 可以看出, 数据库在 DTS 中的工作情况: 首先, 计算模块 (PSM 模块, 本文中 PSM 模块和数据库在同一台 PC 机上) 从数据库读取初始计算参数并开始计算, 1 个步长的仿真结束后, 将计算结果存入数据库, 然后其他用户 (学员站) 从数据库读取实时计算数据。一方面计算模块在不停地将计算结果写入数据库, 同时又有多个用户在查询读取数据库中的实时数据。这时对硬盘的操作为 1 次写操作和 N 次读

操作。显然,这对数据库的查询(即对硬盘的 I/O)次数是很多的,而且对硬盘的 I/O 时间从 DTS 的时间要求看显得过长,远大于对内存 I/O 操作所需的时间。I/O 瓶颈的存在使 DTS 的实时性受到严重制约,CPU 的时间都浪费在 I/O 等待当中。减少每次 I/O 占用的时间,减少 I/O 次数,改善 I/O 瓶颈对保证 DTS 的实时性具有重要意义。很明显,关系型数据库不能满足要求。

1.2 重复查询

DTS 中,学员站或教员站在某一时段需要的实时数据是一些固定的数据变量,如某个变电站的电压、功率变量。传统的数据库输送数据是被动的,即用户查询 1 次就输送 1 次数据,用户要显示 1 幅画面就要不停地向数据库重复相同的查询。在 DTS 中要经常实时观测某些变量的波形(如发电机出口电压波动),此时也同样会出现不停向数据库重复相同的查询,即便画面是 1s 刷新 1 次,DTS 运行时也会造成严重的资源浪费,加上 I/O 瓶颈的存在,重复查询降低了效率、浪费大量的时间,严重制约着 DTS 的数据传输速度。

2 实时内存数据库的提出

针对第 1 节提出的问题,本文将 DTS 中的数据库分为离线数据库和实时内存数据库 2 部分。数据的定义、存档、记录及原始参数,如电力系统结构参数、培训事件表、图形文件、操作记录、历史数据等,统一由离线数据库管理。实时数据部分,经实时内存数据库管理系统初始化后常驻内存,形成完整的实时内存数据库,以支持各种实时应用(教员台、学员台的实时刷新显示)。实时数据的变化将及时返回离线关系数据库,以保证数据的完整性,并定时自动生成历史数据库,提供各种离线应用(教员台和学员台的数据统计显示,曲线、报表的生成等)^[5]。如图 2 所示。

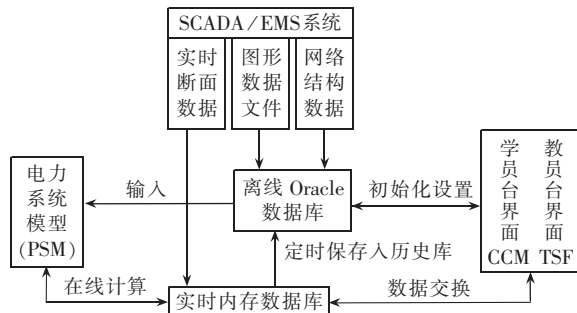


图 2 数据库数据交换关系

Fig.2 Data exchanging relationship in database

本文采用链表结构存储用户信息及其对应的查询信息,实时内存数据库接收到计算模块的仿真数据后,遍历链表,同时主动把每个用户需要的数据发送出去,变“被动”为“主动”,避免了用户重复执行相同的

查询,提高了效率。最后,实时内存数据库遍历链表结束、发送完数据后,再将实时内存数据库的数据写入离线数据库,这样在 1 个仿真步长内把对硬盘查询的 I/O 降低为 1 次,最大程度地减少了 I/O 次数,再加上能够主动输送数据,学员站要显示 1 幅画面也仅需要 1 次查询便可完成,没有冗余操作,查询效率明显高于“非主动”方式的数据库。

离线数据库和实时内存数据库配合使用很好地解决了 DTS 中单纯使用传统关系型数据库存在的问题,提高了效率,可很好地满足 DTS 对实时性的要求。

3 实时内存数据库的实现

使用 VC++ 作为开发工具。采用结构体的方式存储内存中的数据,用 C++ 构建实时内存数据库类,并使用 Windows 下多线程编程实现多用户共享,其中网络数据传输采用客户机/服务器(C/S)模式,利用 Windows Socket API 编程实现。计算模块和实时内存数据库模块间的通信通过匿名管道进行。

3.1 数据存储

计算模块的仿真结果包括仿真时间、仿真步长,电网的所有节点数据、发电机数据、负荷数据和线路数据。针对仿真结果的上述特征,结合现代数据库的发展,同时考虑 DTS 中进程间通信的特点,采用内存数据库与对象数据库^[6]相结合的方法构建实时内存数据库。

首先,将 1 个步长的仿真结果,看成 1 个对象,建立 1 个结构体。

```

struct NetInfo { // 仿真时间和步长
float flagtime, fstep; // 发电机数据
int bamGeId[GeNum]; // GeNum 为发电机总数
float bamGe[GeNum][5]; // 节点、线路、负荷数据
:
};
  
```

其次,创建实时内存数据库类 Crmdb,在类里声明结构体 NetInfo 的对象,仿真结果就保存在结构体中;同时,定义查询函数 GetParam(),可以根据用户的查询要求取得需要的数据发送出去。

```

class Crmdb: public CObject{
DECLARE_DYNAMIC(Crmdb)
public:
int ng, nload, nline; // 发电机、负荷、线路总数
public:
Crmdb();
~Crmdb();
NetInfo NetPara;
float GetParam(int type, int id, int yParam);
};
  
```

3.2 容量

实时内存数据库的数据存放在内存中,如果存放的数据太多,必然会占用大量内存,同时使查询变得

复杂,效率不高;放得少了又会使数据一致性的维护变得复杂(因为可能会只存放某一时刻的部分数据)。同时,因为实时内存库具有主动送数据的特性,若存放多个仿真步长的数据,则在往其中写数据的同时又要往外送数据,这时会使数据的维护十分复杂,而且意义不明显。因为不论内存中存放多少数据,最后往离线库存放的数据总量不变。综合考虑以上因素并结合本文实现的软硬件基础,实时内存数据库只在内存中存放当前时刻的仿真数据,即当前对象。这样数据库的维护、查询就变得相对简单、程序思路清晰、便于实现。

3.3 多用户实现

DTS 中服务器和学员站、教员站之间的数据传输采用 C/S 模式,利用 Windows Socket API 编程实现^[7]。Windows Sockets API 是基于 BSD Sockets 的,它与 Berkeley 软件 4.3(4.3BSD)相兼容,很容易地实现与 Unix(Linux)操作平台的连接、传送数据,具有很强的移植性。

4 匿名管道通信

4.1 匿名管道特征

Win32API 提供 2 种通信管道^[8]:匿名管道(anonymous pipes)和命名管道(named pipes)。匿名管道用于父进程与子进程之间通信,其工作原理是在子进程端用管道的读、写口替代子进程的标准输入或标准输出,这样子进程就能与父进程交换数据。子进程用写句柄(WriteHandle)写数据到管道的同时,父进程用读句柄(ReadHandle)从相应管道中读取这些数据。如图 3 所示。

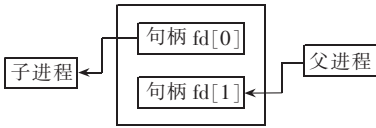


图 3 匿名管道工作原理

Fig.3 Working principle of anonymous pipes

4.2 在 DTS 中的应用

4.2.1 父进程端的设置

把实时内存数据库所在的进程作为父进程。首先,在父进程里创建匿名管道。
`CreatePipe(&hRead,&hWrite,&sa,47190);` //47190 为创建管道的大小
其次,修改管道读句柄使不被继承,同时填充进程启动结构 STARTUPINFO 的信息。
`DuplicateHandle(GetCurrentProcess(),hRead,GetCurrentProcess(),NULL,0,FALSE,DUPLICATE_SAME_ACCESS);`
再次,创建计算模块子进程。DTS 中计算子进程负责向管道中写入仿真数据,父进程则从管道中读取这些仿真数据。

`CreateProcess(NULL,“e:\\bamboo\\run1.exe”,NULL, NULL,TRUE,NULL,NULL,NULL,&si,&pi);`

最后,读取匿名管道中的数据。本文直接把仿真数据写入实时内存数据库类的结构体中,不需要定义新的内存缓冲区。

`ReadFile(hRead,&rmDB.NetPara,sizeof(struct NetInfo),&ReadNum,NULL);` //rmDB 为 Crmdb 类对象

4.2.2 子进程端写入仿真数据

匿名管道建立好后,子进程调用 `GetStdHandle()` 函数取得继承自父进程的写入句柄,并调用 `WriteFile()` 将数据写入到管道。父、子进程间传输的数据格式要一致,按照 `struct NetInfo` 结构体的格式传输数据。

`HANDLE hWrite=GetStdHandle(STD_INPUT_HANDLE);`
`BOOL bTest=WriteFile(hWrite,&NetPara,sizeof(NetPara),&dwWritten,NULL);`

4.3 匿名管道通信方式的优点

匿名管道进程间通信的机制是一种系统内部高效的数据通信和同步机制,它不生成额外的文件,又能有效地管理通信过程中的内存。这种隐藏的数据通信方式为应用程序间的通信和同步提供了极大的灵活性。

4.3.1 共享内存进行进程间通信

共享内存是对将要在多个进程间映射和共享的内存区域(段)所作的映射^[9]。消息和数据可以直接从某内存段映射,并映射到调用进程的寻址空间。使用共享内存技术,不同进程可以同时访问相同的内存区域。能够通过共享它们地址空间的若干部分,然后对存储在共享内存中的数据进行读和写,从而实现彼此直接通信,其原理如图 4 所示。

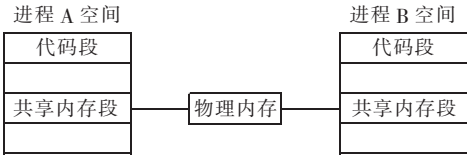


图 4 共享内存的工作原理

Fig.4 Working principle of shared memory

对进程通信是一种很快但在某种程度上并非最容易的方法。这些数据以一种非顺序的随机方式访问。为避免数据不一致的情况,常引入信号量协调对共享内存段的访问。多个进程共同使用同一段物理内存空间,不需要数据的多次复制,提高了进程间的通信性能,但必须解决复杂的同步和互斥问题。因此,具体实现时比较复杂。

具体实现时,在计算模块进程创建 `CSMServer` 类,建立共享内存对象、映射共享内存对象到本进程的地址空间内,然后把仿真数据写入上述地址空间。在实时内存数据库进程创建 `CSMClient` 类,用来访问共享

内存对象、映射对象视图及读取共享内存地址空间的仿真数据。

4.3.2 DTS 仿真中的对比试验

在 DTS 开发过程中,使用匿名管道进行进程间通信相比套接字与共享内存技术具有以下优点。

a. 安全可靠、速度快。DTS 对可靠性与实时性的要求很高,而计算进程与实时内存数据库进程间传输的数据量很大(本文为 47 188 Bytes),采用匿名管道的方法速度最快^[10],安全性最高,因为数据在管道中传输,不会丢失。基于本文的软、硬件平台,测试 4 000 次,取平均值,每次传输数据为 47 188 Bytes。测试结果如表 1 所示。

表 1 进程间通信时间比较
Tab.1 Comparison of IPC time

使用方法	平均时间 / μ s
Anonymous Pipes	718
File Mapping	987
Socket	63 135

b. 编程简单、思路清晰。DTS 中,计算模块按照预先设置的步长仿真,如果使用共享内存技术通信,计算模块写共享内存区与实时内存数据库读共享内存区之间没有同步机制,必须使用消息或者其他方法进行读/写同步处理。而匿名管道则不需要进行同步处理,这种方法本身就包含了同步读/写机制,并且,从上面的介绍可以看到其编程实现十分简单。虽然套接字也不需要额外的同步处理,但是它要分多次传输 1 个步长的仿真结果,效率较低。

c. CPU 占用率低、节省 CPU 时间。基于本文的软硬件条件,在 DTS 中传输同样大小的数据,匿名管道的方法,相比共享内存与套接字的方法 CPU 占用率低、节省 CPU 时间。仿真循环 3 000 次,所得试验数据见表 2。为简化,用子进程代替计算进程,父进程代替实时内存数据库进程,Pipe 代表匿名管道方法,SM 代表共享内存方法。

表 2 IPC 通信占用 CPU 情况对比
Tab.2 Comparison of CPU status in IPC

使用方法	CPU 占用率 / %	占用 CPU 时间 / s
子进程(Pipe)	96	518
父进程(Pipe)	0	2
子进程(SM)	98	558
父进程(SM)	1	3
子进程(Socket)	77	560
父进程(Socket)	21	180

5 结语

本文结合某省 DTS 的具体情况,首先提出了一种新的构建实时内存数据库的方法,并介绍了其具体实现的步骤;此方法从减少查询次数、优化查询、减少 I/O 次数等方面入手,很好地解决了 I/O 瓶颈、重复

查询等一些浪费时间的问题,实时内存数据库具有快速高效(比单独使用关系数据库,访问效率提高 3~5 倍),满足了 DTS 的实时要求;采用面向对象的方法构建实时内存数据库,简洁直观、移植性强;实时内存数据库,在类里声明查询函数和结构体对象,使用结构体存储仿真结果,使得采用共享内存或管道技术读取数据时,方便、简单。其次,针对本文所构建实时内存数据库的特点,提出使用匿名管道的方法进行计算进程与实时内存数据库进程间的通信;开发实践证明,这种方法速度快、可靠性高、编程实现简单,能够很好地满足 DTS 对实时性和可靠性的要求。

参考文献:

[1] 张力平. 电网调度员培训模拟(DTS)[M]. 北京:中国电力出版社,1999.

[2] 谷毅,赵玉柱,张国威. 我国 DTS 技术及其应用[J]. 电力系统自动化,2002,26(13):60-62,65.
GU Yi,ZHAO Yu-zhu,ZHANG Guo-wei. A survey of the dispatcher training simulator and its application in China [J]. **Automation of Electric Power Systems**,2002,26(13):60-62,65.

[3] 白玮,高平,逢健鹏,等. DTS-SOTS 实时通信接口的设计与实现[J]. 电力自动化设备,2002,22(6):33-35.
BAI Wei,GAO Ping,PANG Jian-peng,et al. Design and implementation of real time communication interface between DTS and SOTS[J]. **Electric Power Automation Equipment**,2002,22(6):33-35.

[4] 刘云生. 现代数据库技术[M]. 北京:国防工业出版社,2001.

[5] 董朝霞,高建军,陈青华,等. 电网调度员培训仿真系统数据库的应用研究[J].水电能源科学,2002,20(1):66-68.
DONG Zhao-xia,GAO Jian-jun,CHEN Qing-hua,et al. Research on database in power system dispatcher training simulator[J]. **International Journal Hydroelectric Energy**,2002,20(1):66-68.

[6] 王意洁. 面向对象的数据技术[M]. 北京:电子工业出版社,2003.

[7] 琼斯,欧伦德. Windows 网络编程[M]. 2 版. 杨合庆,译. 北京:清华大学出版社,2002.

[8] 汤春林,吴东,卢焕章. 匿名管道在 Win 9x 环境下的应用研究[J]. 计算机应用研究,2002,19(6):57-59.
TANG Chun-lin,WU Dong,LU Huan-zhang. Application of anonymous pipes in Win 9x environment[J]. **Application Research of Computers**,2002,19(6):57-59.

[9] 刘晓达,张志祥,刘霞. 基于共享内存的 IPC 的应用研究[J]. 舰船电子工程,2003(1):25-28.
LIU Xiao-da,ZHANG Zhi-xiang,LIU Xia. Application research of IPC based on shared memory[J]. **Ship Electronic Engineering**,2003(1):25-28.

[10] 胡亮,王敏珍,蒋春晓,等. 机内进程间通信的性能测试和评价[J]. 吉林大学学报,2003,21(4):98-101.
HU Liang,WANG Min-zhen,JIANG Chun-xiao,et al. Testing and estimation of interprocess communication per-

formance within a host[J]. **Journal of Jilin University**,2003,21(4):98-101.

(责任编辑:汪仪珍)

作者简介:
常鲜戎(1956-),男,吉林长春人,教授,博士,研究方向

为非线性系统控制、电力系统分析和稳定控制、电机故障监测与控制等;

张帅民(1979-),男,河南洛阳人,硕士研究生,研究方向为电力系统分析和稳定控制(**E-mail**:bamboo_21@tom.com);

姜志宇(1979-),男,辽宁抚顺人,硕士研究生,研究方向为电力系统分析和稳定控制。

Construction of real-time memory resident database and
anonymous pipes used in IPC of DTS

CHANG Xian-rong^{1,2},ZHANG Shuai-min^{1,2},JIANG Zhi-yu^{1,2}

(1.Key Laboratory of Power System Protection and Dynamic Security Monitoring
and Control under Ministry of Education,North China Electric Power University,
Baoding 071003,China;2. School of Electrical Engineering,North
China Electric Power University,Baoding 071003,China)

Abstract: A new real-time memory resident database combining the memory database and objective database is put forward for high-speed data transmission in DTS databases,which takes into account the characters of database access and IPC(Inter Process Communication) in DTS. It has fast I/O without redundant operations and sends data to users actively according to the query received. Anonymous pipes are adopted in IPC between calculation process and real-time memory database process,and the shared memory method and the socket method are compared. Partial codes and test data are provided. Practice shows it effective.

Key words: DTS; real-time memory resident database; anonymous pipes; inter-process communication; real time; shared memory