多线程异地备份发动机数据软件设计与应用

黄国良[1]，张 锐[2]，罗朝辉[1]
（1. 西安航天动力研究所，陕西 西安 710100；2. 北京航天动力研究所，北京 100076）

摘 要：通过开发发动机信息管理系统后台数据的异地备份软件，介绍了以 Delphi 10 Seattle 为开发工具，应用 Indy 组件，多线程方式实现客户端向服务器端发送备份指令，完成数据的备份、加密压缩以及向异地客户端推送数据，以客户端完成压缩数据的接收结束异地传输，同时服务器端实时向不同客户端发送执行过程信息。因发动机数据量较大，备份和传输时间较长，在异地客户端采用同步进度条显示。软件设计采用 TCP 可靠传输，提高数据传输的安全性。

关键词：发动机信息管理；多线程异地备份；Delphi 10 Seattle；Indy；软件设计
中图分类号：V434-34 **文献标识码**：A **文章编号**：1672-9374 (2016) 04-0062-06

Software design and application for multithreading offsite backup of liquid rocket engine data

HUANG Guoliang[1], ZHANG Rui[2], LUO Zhaohui[1]
(1. Xi'an Aerospace Propulsion Institute, Xi'an 710100, China;
2. Beijing Aerospace Propulsion Institute, Beijing 100076, China)

Abstract: Software for background data offsite backup of liquid rocket engine information management system is developed. Based on Delphi 10 Seattle as developing tool and Indy component, the data backup, encryption, compression and data pushing to offsite client-side are realized by sending the backup instruction from customer end to server end in multithreading mode. The data offsite transmission is finished at the end of the compressed data reception of client-side. The information in executing process is sent to different client-sides in real time by server end. Transmission of the data backup needs a long time period due to large size of the engine data, so a synchronous progress bar is displayed at each client-side. The reliable TCP transmission is adopted in the software design to ensure the safety of data transmission.

Keywords: engine information management; multithreading offsite backup; Delphi 10 Seattle; Indy; software design

收稿日期：2016-06-15；**修回日期**：2016-06-22
作者简介：黄国良（1970—），男，工程师，研究领域为数据库技术

0 引言

发动机信息管理系统项目投入使用后，该系统的数据具有数据增速快、业务依赖性高、数据交换频繁等特点。同时，保障管理系统数据的安全越来越成为系统长期稳定运行的关键，也成为保障科研生产、试验顺利进行的关键，而数据备份的重要性却往往被忽视，只要在网络中发生数据的传输、存储和交换，就有可能产生数据故障，不及时采取数据备份，有时造成的损失是无法弥补与估量的。

数据备份不是单纯的数据复制，尤其是关系型数据库，各种关系图、数据表和视图等组成的数据之间的逻辑关系，如控制文件、数据文件。使用单纯的文件复制方式无法保障数据的完整性，数据恢复后会造成应用系统因缺少数据之间的逻辑控制而崩溃。数据库虽然提供简单的备份工具，但远不能满足多客户端对异地数据备份的需要。多线程方式发动机数据异地备份软件是基于封装系统数据库的动态链接库，独立开发服务器应用系统，采用服务器端多线程数据备份，以客户端交互方式实时显示备份和异地文件传输的进度，既解决了多客户端任务处理、数据备份、异地传输、数据安全性的问题，也便于客户端直观了解数据在备份和网络传输过程中的进度。

1 软件需求分析

发动机数据备份软件的功能需求：软件由两部分组成，第一部分由服务器应用程序配置服务器端，将连接到服务器上的客户端分配不同 ID 的线程，处理来自客户端的消息、会话及指令，由指令完成数据库的备份、压缩及数据的推送；第二部分由客户端向服务器端发送客户端消息、会话及指令，并接收来自服务器 TCP 传输的数据包，存储备份文件，以进度条形式显示、接收服务器端推送数据，完成客户端的异地存储。

2 软件设计

2.1 软件功能设计

软件功能设计见图 1，分服务器端和客户端，根据需求服务器端软件主要包括：服务器设置、数据处理、文件压缩、Tcp 数据传输 4 个功能模块。客户端软件主要包括：连接设置、客户端数据处理两个功能模块。

2.1.1 服务器端

1）服务器设置。服务器设置包括端口设置、基本设置、数据库设置、传输设置，分别对服务器 IP 地址、服务器端口号、数据库备份文件目录、压缩文件目录、压缩应用文件目录及口令、数据库 IP 地址、用户名口令、数据库名称、传输端口号进行配置等。

2）服务端数据处理。在数据处理功能中包括客户端多线程处理、指令多线程处理、客户端到服务器端消息处理、客户端至客户端会话处理、来自客户端指令处理等。

3）文件压缩。文件压缩功能包括文件压缩处理，调用配置好的压缩工具对需要备份的数据库进行加密压缩，减少数据在网络中传输的压力提高传输安全性。

4）Tcp 数据传输。Tcp 数据传输包括流文件传输、备份进度数据传输。在服务端将压缩的文件以流文件的形式传输到客户端，同时交换传输进度数据。

2.1.2 客户端

1）连接设置。客户端连接设置功能包括基本设置、传输设置、连接的服务器 IP 地址、服务器端口号、存储备份文件目录及传输端口号设置。

2）客户端数据处理。客户端向服务器端发送消息、对话和指令数据，接收由服务器端反馈的消息、流文件以及其他客户发送的对话，流文件接收过程中显示接收的进度，全部接收后对文件进行解压处理，完成数据库在异地存储。

2.2 软件逻辑设计

发动机信息管理系统的管理客户端分布在局域网的不同物理接入点，每个客户端连接服务器程序，同时数据备份的冲突较大，相互并不知道管理系统后台的数据库在执行状态，服务器应用程序创建多线程，分别解析每一个连接的客户端的请求，服务器端向客户端发送不同的实时信息，执行相应的操作。一个客户端就是一个独立

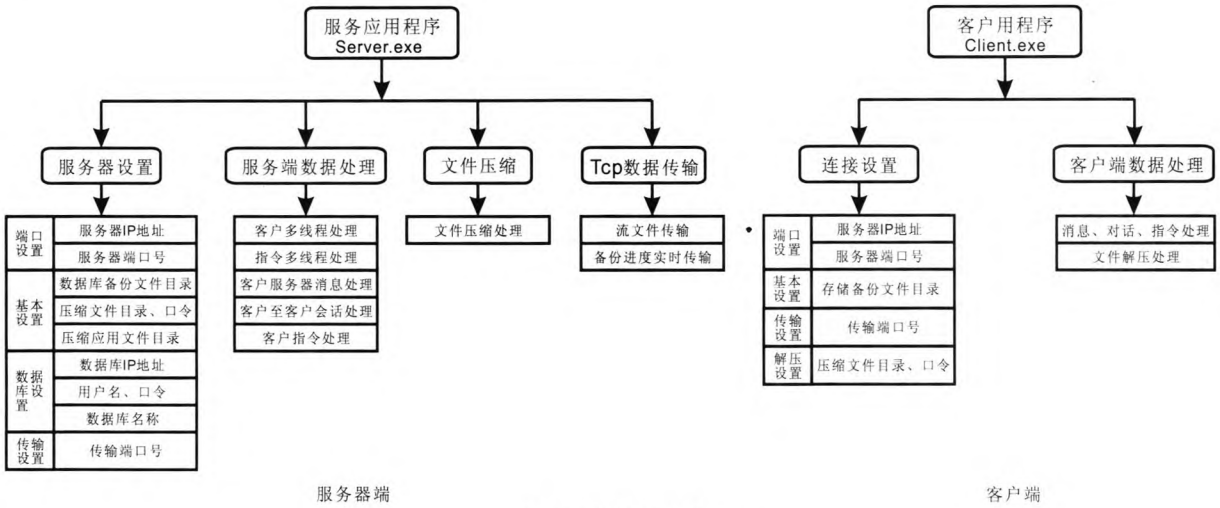

图 1 软件功能设计

Fig. 1 Function design of software

线程直到该用户退出时销毁该线程释放内存，为创建下一个线程做准备，以此反复。

2.2.1 服务器端程序逻辑设计

服务器端应用程序正确配置后以系统的服务模式运行，在指定的端口监听客户端的连接请求，客户端向服务器端发出请求后，创建一个新的唯一的线程 ID，区别不同客户，同时创建指令线程，以客户端发送的请求作为条件，服务器端根据记录类型参数中的标示判断不同的客户端，提供相应的服务。如果是指令则执行备份指令线程，客户端可以通过进度条实时了解服务器端的备份状态，备份完成后查看压缩配置的文件是否正确，如果配置正确调用压缩程序压缩备份文件，压缩完成后将打开文件传输端口以流文件的形式向客户端传输文件，此时客户端同样以进度条的状态了解流文件传输的进度，当传输完毕后一个完整的备份指令流程执行完成。在整个流程中如果客户端发出的是消息或对话，则要判断是对所有客户端还是对某一个 IP 地址的客户端，服务器端应用程序判断后执行会话的操作完成客户之间的会话。备份指令执行后要调用压缩应用程序，如果配置不正确程序将转入查找客户端线程 ID 发送消息或对话框通知客户端，修改服务器端的配置。

2.2.2 客户端程序逻辑设计

客户端通过连接服务器端的 IP 地址和端口，利用 TCP 协议连接双方，选择发送消息、对话或指令，同时接收服务器端的反馈信息，如果向服务器端发送的是指令则实时显示备份的进度，等待备份完成后，客户端接收到服务器端发送的压缩消息，等待接收流文件，在接收的过程中，分离服务器端传输的数据获得流文件的大小，显示接收文件的进度，同时判断接收是否完成，没有完成循环接收，接收结束后判断解压应用程序配置是否正确，配置正确则调用解压应用程序，错误则提示对话框结束程序，重新配置后再重复一个流程，将文件解压恢复备份文件，完成客户端的实时异地文件的备份，服务器端和客户端程序逻辑设计见图 2。

3 开发环境及关键代码实现

3.1 开发环境

发动机信息管理系统的开发采用的是 Borland 公司的可视化开发工具 Delphi7，后台数据库使用的是 Microsoft 的关系数据库 SQL Server 2000。为与原系统开发语言保持一致，选择了 Delphi 10 Seattle 开发工具，它是原开发工具的升级版，拥有全新的可视化集成开发环境 (IDE)，基于组件的开发结构框架和采用 ObjectPascal 面向对象的编程语言，可以直接编译生成可执行代码，编译速度快，运行效率高，程序运行稳定。支持一个项目包含客户端或服务器端同时开发方

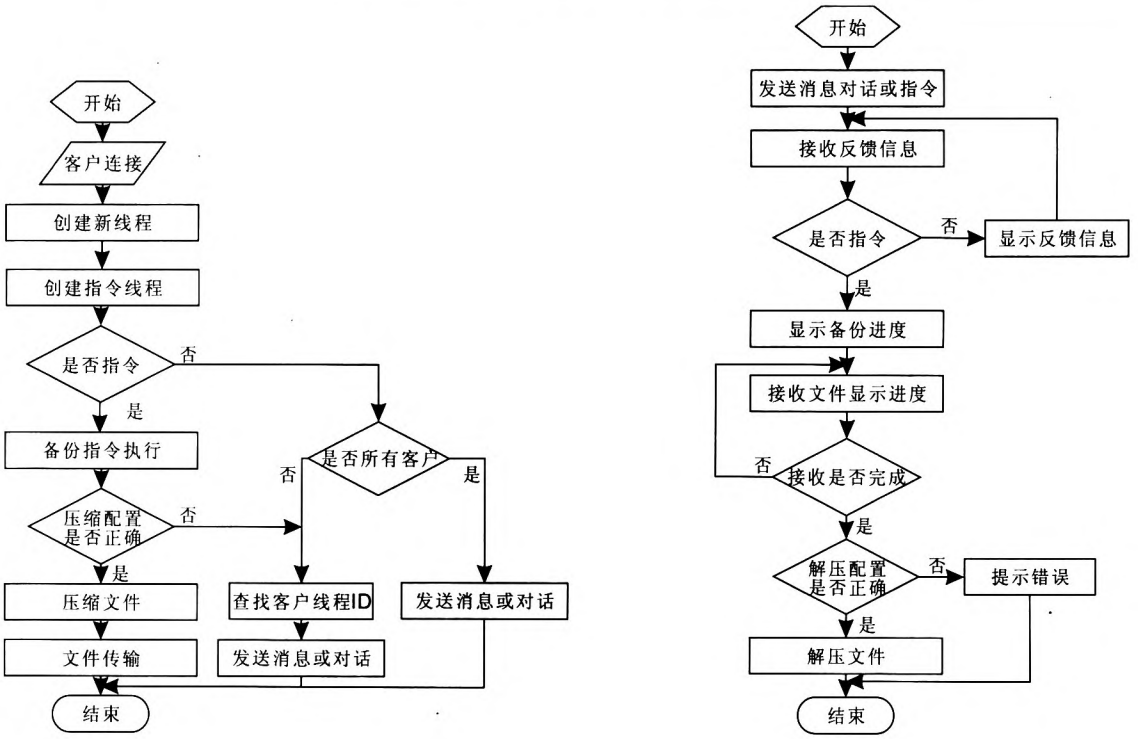

图 2　服务器端和客户端程序逻辑设计

Fig. 2　Program logic design for server end and customer end

案，而且允许开发人员建立组件或组件集合，封装所有的规则，并独立于服务器端和客户端，所有的数据传输通过标准组件来完成。这样，大大减少了对服务器的请求和网络上的数据传输量，提高了开发应用的速度。Delphi 10 Seattle 包含的 INDY 组件，支持大部分 Internet 协议，包括 TCP，UDP，DNS，FTP，HTTP，POP3，SMTP，TELNET，WHOIS 等，提供 INTERNET 协议的客户端和服务器组件，方便快速的建造各种服务器、客户端程序，并且这些组件都是支持多线程的。在本软件开发过程中主要用到了 INDY 的 idTCPServer 和 idTCPClient 两个组件建立客户端与管理端之间的 TCP 连接。操作系统支持 Windows xp 及以上版本。

3.2　关键代码实现

3.2.1　记录类型声明

定义服务器端和客户端信息交换的标准，分别在服务器端和客户端声明后，引用该声明单元。程序片段如下：

```
type
  TCommBlock = record
               Command,
               MyUserName,
               Msg,
               ReceiverName: string [100] ;
             end;
```

3.2.2　服务器端创建新的线程和指令线程

服务器端使用第一个 idTCPServer 组件监控客户端的请求，对于接收的每一个客户端，都创建一个新的线程来提供服务，所有与这一客户端相关的事务都由该线程来处理，由于执行数据库备份指令也是多线程，在一个新线程创建后同时创建一个指令线程。创建过程在 idTCPServer 组件被请求连接过程中完成。程序片段如下：

```
  try
coinitialize (nil) ;
GetMem (NewClient, SizeOf (TClient)) ;
 NewClient.DNS          : =  AContext.Connection.
```

```
Socket.Binding.PeerIP;
  NewClient.Connected   := Now;
  NewClient.LastAction  := NewClient.Connected;
  NewClient.Thread      := AContext; //创建新的线程
   NewClient.AdoQuery_N  : =  TADOquery.create (ServerFrmMain) ; //创建新的指令线程
  AContext.Data:=TObject (NewClient) ;
     finally
      couninitialize (nil) ;
    end;
  try
    Clients.LockList.Add (NewClient) ;
  finally
    Clients.UnlockList;
  end;
```

3.2.3 服务器端文件传输

使用第二个 idTCPServer 组件传输压缩后的流文件，服务器端创建流文件，进入监听状态，文件压缩完成后开始发送文件，服务器端根据流文件传输位置按接收缓冲区大小一块一块的发送给客户端，直到整个文件流发送完毕，客户端接受后再保存到接收文件。

程序片段如下：

```
with AThread.Connection do //已经创建的一个进程
    AFileStream.Seek (StrToInT (cmd) , soFromBeginning) ;
    ASize := Min (AFileStream.Size - AFileStream.Position, RecvBufferSize) ;
    OpenWriteBuffer;
    WriteStream (AFileStream, false, false, ASize) ;
    CloseWriteBuffer;
    StatusBar1.SimpleText := Format (´当前传输位置%s/大小%d´, [cmd, AFileStream.Size]) ;
    Gauge1.Position := Gauge1.Position + ASize;//显示发送的进度
end;
```

3.2.4 客户端备份进度显示

服务器端在执行备份指令过程中，调用 SQLDOM 类，SQLDOM 是 SQL server2000 企业管理器的应用程序接口，客户端实现远程备份和恢复功能，在客户端声明 TBackupSink.PercentComplete 函数，在 SQLDOM 被调用时函数将显示备份进度。.程序片段如下：

```
function  TBackupSink.PercentComplete (  const Message: WideString; Percent: Integer) :HResult;
begin
 result:=0;
 Form1.Gauge2.Progress:=percent;//  备份进度显示
end;
//////////////先将数据备份到服务器上//////////////////
BS:=TBackupSink.Create;
 MySQLServer:=coSQLServer.Create;
 MyBackUp:=coBackUp2.Create;
    MySQLServer.Connect (  ´10.139.1XX.XX´,´sa´, ´XXXXX´) ;;
 MyBackUp.Database:=´LibraryDBa´;
 MyBackUp.Initialize:=true;
 MyBackUp.PercentCompleteNotification:=1;
 MyBackUp.Action:=0;//0 完整备份，1 差异备份，2 文件组备份，3 日志备份
 Randomize;
 Server_FileName:= ´LibraryDB067a_hgl_´+stringreplace ( udpsocket1.LocalHostAddr,´.´,´_´,  [ rfRe - placeAll]) +´.BAK´;
 MyBackUp.Files:=´H:\ 全文 \´+Server_FileName;//´d:\LibraryDB067.bak´;
InterfaceConnect (MyBackUp, IID_BackupSink, BS, FInterfaceConnection)  ;  MyBackUp.SQLBackup (MySQLServer) ;
 InterfaceDisconnect (MyBackUp, IID_BackupSink, FInterfaceConnection) ;//
 messagebox (handle,´服务器数据库备份成功！´,´提示´,mb_ok) ;
```

3.2.5 客户端接收文件进度显示

使用第二个 idTCPClient 组件接收压缩后的流文件，在一个线程中，以缓冲区大小一块一块的接收流文件，与流文件总字节数之比形成接收的进度，直至接收完毕，保存到文件。程序片段如下：

```
TotalSize := StrToInt (Copy (cmd, 0, Pos ('|', cmd) - 1)) ; //分离文件大小
    AFileStream := TFileStream.Create (SaveDialog1.FileName, fmCreate) ;
  try //循环开始接收
    repeat
          IdTCPClient1.WriteLn ( IntToStr (AFileStream.Size)) ;//发送当前传输的位置
      ASize := Min (TotalSize - AFileStream.Size, IdTCPClient1.RecvBufferSize) ;
      //选择剩余大小和缓冲区大小作为传输的大小
       IdTCPClient1.ReadStream ( AFileStream, ASize) ; //接收流
      StatusBar1.SimpleText := Format ('当前传输位置%d/大小%d', [AFileStream.Size, TotalSize]) ;
      Gauge1.Progress:= AFileStream.Size;//显示接收进度
      Application.ProcessMessages;
    until AFileStream.Size = TotalSize; //大小一致了表示结束
  finally
    AFileStream.Free; //释放文件流
  end;
  IdTCPClient1.WriteLn ('END') ; //提示服务器传输完成
Protocol.Lines.Add ('客户端数据备份成功！') ;
```

4　软件测试与应用

经过调试和运行后，为测试多线程的并发执行的效率，又鉴于整个执行过程有大量的网络数据传输，受到当时网络状态的影响较大，于是选择将服务器端应用程序和数据库部署在一台服务器上由一个用户单独执行备份指令，三个用户并发执行备份指令以及十五个用户并发执行备份指令，由程序发出指令自动计时到数据库备份至服务器完成的时间进行测试。通过对测试结果分析，多线程的并发并未明显延迟指令的执行时间，指令反应及时，有细微的时间消耗，而执行过程中有大量的 IO 操作并未进行时间的统计，时间对比见表 1。

表 1　指令执行时间对比

Tab. 1　Comparison of instruction executing time

用户数量	执行时间	备份库文件大小	运行环境
1 个用户	2′06"	3.2 GB	双 CPU X5660 2.8 GHz 12 GB DRAM windows 2003 SQL Server2000
3 个用户	2′10"		
15 个用户	2′13"		

在应用过程中，该软件服务器端可以和数据库部署在同一个服务器上，也可以部署在应用程序服务器上，通过软件的配置连接数据库服务器，充分利用服务器硬件资源提高应用程序的执行效率，减小并发程序时服务器的运行压力。

5　结论

多线程异地备份发动机数据软件以服务器端应用软件为核心，基于 TCP 传输协议传输备份流文件。采用多线程编程技术，客户端方便及时了解数据库备份进度，解决了发动机信息管理系统多用户实时数据库异地备份的要求，在系统数据库出现故障或系统瘫痪时能及时通过异地备份的数据快速、完整、简捷地恢复系统的正常运行。经过实际应用，软件运行良好、性能稳定，提高了异地数据备份效率，为发动机信息管理系统数据安全提供了可靠的保障。

参考文献：

[1] 王文龙, 张少博, 陈海峰. 一种试验数据处理软件设计[J]. 火箭推进, 2012, 38(1): 76-80.
WANG Wenlong, ZHANG Shaobo, CHEN Haifeng. Design of a test data processing software[J]. Journal of rocket propulsion, 2012, 38(1): 76-80.

[2] 鲍敏, 吴昊. Delphi 网络高级编程[M]. 北京:人民邮电出版社, 2001.

[3] 何进, 谢松巍. 基于 Socket 的 TCP/IP 网络通讯模式研究[J]. 计算机应用研究, 2001 (8): 134-135.

（编辑：王建喜）