基于ADO技术的分段数据存取方法研究

Research Based on ADO Segmented Data Access Method

郭蕴华 陈定方（武汉理工大学智能制造与控制研究所 430063）

摘 要：文中介绍了分布式应用系统体系结构的变迁，概括了多层分布式应用系统在执行效率方面的一般性设计原则。在此基础上，设计并实现了基于ADO技术的分段数据存取方法，用以提高多层分布式应用系统的执行效率。

关键词：ADO 分布式对象 数据库

1 分布式应用系统体系结构的变迁

分布式应用系统体系结构的变迁是一个沿着两条主线渐进的发展过程。这两条主线，一条是Internet的演进过程，一条是分布式对象的演进过程。

Internet的出现为软件的开发带来了多元的方式，其发展的历程中发生了3次重大的技术革命：第一次技术革命是HTTP，为电子商务的出现和发展奠定了基础；第二次技术革命是Java语言的兴起，为跨平台带来了契机；第3次技术革命是XML技术，让数据的交换不受操作系统和开发工具的限制。

面向对象技术是软件工程发展的必然趋势，分布式对象技术则是面向对象技术在分布式计算领域内的延伸和拓展。1991年，国际非盈利组织OMG制订了CORBA规范，并不断增加新内容和新技术，版本也不断升级，目前的最新版本为CORBA 2.4。1993年，Microsoft公司在发行OLE 2.0时实现了COM的第一个版本。随后，Microsoft又先后推出DCOM和COM+，形成了CORBA与DCOM在分布式对象领域长期竞争的局面。

关系数据库的出现使得两层的客户/服务器应用系统得到了充分的发展，但客户/服务器结构在维护、分发和可重用性等方面都有天生的缺陷，很快就逐步被多层分布式结构所取代。1997年前后，分布式对象技术已经日趋成熟，各厂商也纷纷推出面向多层分布式系统的解决方案。到2000年，市场上已经出现了IBM、HP、BEA、Oracle和Borland等多家公司的应用服务器，此时，在大型应用中，两层结构已经基本上被多层结构完全取代。然而，由于厂商林立，标准不统一，在系统整合时，不同类型的分布式对象之间的数据交换成为主要的技术难点。同时，由于层次环节的增加，多层分布式体系结构中的执行效率问题也逐步暴露出来。

Microsoft提出的SOAP协议是分布式计算技术的发展过程中的一个重要的里程碑。由于SOAP协议使用XML作为封装和交换信息的标准，因此可以在不同的平台、不同的语言和不同的分布式对象之间解析和处理SOAP封包。至此，分布式应用系统体系结构变迁的两条主线汇聚成一条，分布式体系结构逐步向WebService演进。

但是由于XML是一种文本格式的数据交换标准，它必然导致层次之间传递的数据量增大。例如，当以SOAP协议从数据库获取BLOB类型的字段的内容时，必须将二进制流转换成Base64的编码，使数据量约增加了1/3。所以，使用SOAP协议时，整个系统在执行效率方面的矛盾将会更加突出。

2 多层分布式应用系统的执行效率瓶颈

2.1 导致执行效率下降的主要因素

导致多层分布式应用系统执行效率的下降涉及到众多复杂的因素，简单地讲，以下两点是对执行效率的影响比较大。

（1）区性就实际的架构来说，由于数据需要从数据库先传递到中间层，再由中间层传递到客户端应用程序，因此会比客户/服务器结构直接将数据从数据库传递到客户端应用程序进行的缓慢。尤其是使用DCOM的分布式应用系统，由于使用COM/DCOM Marshalling的方式传递数据，数据负荷比直接使用TCP/IP等通信协议要大很多。

（2）目前的分布式应用系统基本上都是以分布式对象技术为基础的，在完成远程过程调用的过程中需要花费大量时间建立和调用远程对象。当然，目前的一些中介软件提供了对象池功能，避免每一个客户端需要重复建立相同的对象，但前提是提供服务的远程对象必须是无状态对象(Stateless Object)，即这些远程对象在客户端调用完毕以后，不需要为客户端维护任何状态信息。

2.2 提高执行效率的一般性设计原则

为了提高分布式应用系统的执行效率，在开发和设计时应遵循一定的原则，具体概括如下：

（1）不要一次传递大量的数量，特别是用户不需要的数据。事实上，即使用户选择的查询条件可以从数据库中一次获取10000条记录以上的数据，用户也不可能一次对如此多的数据完成浏览和处理。所以，除非在特殊情况下，传递数据时应遵循“少量多

次”的原则。

（2）尽量使用无状态对象，以便能够使用应用服务器的对象池功能，从而可以有效的降低客户端建立和调用远程对象的时间。

3 基于ADO技术的分段数据存取方法

3.1 ADO技术的基本原理

ADO是Microsoft存取通用数据源的标准引擎。ADO由一组COM对象组成，每一个不同的COM对象都负责不同的工作。其中，RecordSet对象是ADO对象的核心对象之一，它负责向数据库下达SQL命令，并维护数据库回传的数据集。RecordSet有一个重要的属性，即MaxRecords，它控制从数据库中获取数据集的最大记录数。MaxRecords对于分段存取非常重要，因为对于从数据库获取指定记录数的数据集这一功能而言，各数据库厂商的实现方法不尽相同，例如SQL Server是通过扩展的SQL语法来实现的，而Oracle则是用伪列来实现的。有了MaxRecords属性，那么多数支持ADO的数据库在获取指定记录数的数据集这一功能上就基本统一了，从而保证了分段存取的通用性。

3.2 概要数据与详细数据

概要数据和详细数据是指相同数据源的两种不同的存取数据的方式。概要数据以分页的形式向客户显示某个表的主要字段内容，每页中的记录数量是确定的；详细数据向客户显示某个表的全部字段内容，每次只显示一条记录。概要数据只能用于浏览，不能用于更新；详细数据既可以用于浏览，也可以用于更新。缺省的情况下，向用户显示概要数据，且允许用户翻页。当用户在概要数据中选中某条记录，并要求获取更为详尽的信息时，就弹出一个新的界面，显示详细数据。

3.3 分段获取概要数据

分段获取概要数据时，假设每段的记录数为缺省显示前条记录，即第一页。同时，还应该提供“下一页”、“上一页”和“到指定页”等功能，以便让用户翻页。此处，给出一个COM+环境下、以Delphi 6.0为开发工具，同时结合Borland公司的MIDAS技术的设计实例。在Delphi中，TADOQuery对象的MaxRecords属性就反映RecordSet的MaxRecords属性。

首先，在中间层建立一个提供概要数据的COM+对象ComtsGData，该对象通过接口ImtsGData(它的实现类为TmtsGData)向客户端提供GetNextPage、GetPriorPage和GetAPage方法，分别实现“下一页”、“上一页”和“到指定页”的功能。这几个方法通过TADOQuery对象ADOQGData向数据库发出SQL命令，数据库返回的数据集由TDataSetProvider对象dspGData来维护（即dspGData的DataSet属性为ADOQGData）。假设数据源的表名为TableName，关键字段为整型字段KeyID，其他字段为Fd1，Fd2，……，FdK，下面给出这几个方法的实现。

```
Procedure TmtsGData.GetNextPage (
KeyValue: integer; var vDatas: OleVariant);
Var
 RecsOut: Integer; //获取数据时发生错误的次数
  Options: TGetRecordOptions; //获取数据时的选项
Begin
 Try
ADOQGData. MaxRecords := 20;
//设定每页记录数为20
//****设定ADOQGData的SQL，查询条件为***
//****关键字段KeyID大于KeyValue的记录****
 ADOQGData.Active := false;
ADOQGData.SQL.Clear;
ADOQGData.SQL.Add(
     'Select KeyID, Fd1, Fd2, Fd3 from TableName' + 'where KeyID > ' + IntToStr (KeyValue) + ' ASC');
ADOQGData.Active := true;
// ************************
Options := [grMetaData, grReset];
//设置获取数据时的选项，要获得元数据
vDatas := dspGData. GetRecords(
-1, RecsOut, Byte(Options)); //获取数据
  SetComplete; //提交COM+事务
 Except
  SetAbort; //回滚COM+事务
 End;
End;
```

GetNextPage方法的KeyValue参数由客户端传入，是指当前页的最后一条记录的关键字段的值；vDatas参数以MIDAS数据封包的格式向客户端返回所要求的数据集。GetNextPage方法首先设定ADOQGData的MaxRecords为指定的记录数20，然后向数据库发出SQL命令，获取KeyID大于KeyValue的前20条记录，最后用vDatas参数向客户端返回数据集。

```
Procedure TmtsGData.GetPriorPage (
KeyValue: integer; var vDatas: OleVariant);
Var
 RecsOut: Integer; //获取数据时发生错误的次数
  Options: TGetRecordOptions; //获取数据时的选项
Begin
 Try
ADOQGData. MaxRecords := 20;
//设定每页记录数为20
//****设定ADOQGData的SQL，查询
```

```
条件为***
    //****关键字段KeyID小于KeyValue的记录****
    ADOQGData.Active := false;
    ADOQGData.SQL.Clear;
    ADOQGData.SQL.Add(
    'Select KeyID, Fd1, Fd2, Fd3 from TableName' + 'where KeyID < ' + IntToStr(KeyValue) +' DESC');
    ADOQGData.Active := true;
    // ************************
    Options := [grMetaData, grReset];
    //设置获取数据时的选项，要获得元数据
    vDatas := dspGData. GetRecords(
    -1, RecsOut, Byte(Options)); //获取数据
      SetComplete; //提交COM+事务
    Except
      SetAbort; //回滚COM+事务
    End;
  End;
```

对于GetPriorPage方法而言，KeyValue参数是指当前页的第一条记录的关键字段的值。该方法首先设定ADOQGData的MaxRecords为指定的记录数20，然后向数据库发出SQL命令，以逆序的方式获取KeyID小于KeyValue的前20条记录，最后用vDatas参数向客户端返回数据集。逆序的方式是非常必要的，否则数据库返回的是全部符合查询条件的数据集的前20条记录，而不是上一页的20条记录。如果客户端仍然需要正序的数据，可以在ADOQGData的OnAfterOpen事件中，设定的ADOQGData的Sort的属性为"KeyID ASC"，从而重新按正序排列数据

```
  Procedure TmtsGData.GetAPage(
  PageNum: integer; var vDatas: OleVariant);
  Var
  RecsOut: Integer; //获取数据时发生错误的次数
    Options: TGetRecordOptions; //获取数据时的选项
    KeyValue: integer;
  //指定页的前一页的最后一条记录的值KeyValue
    ADOQTemp: TADOQuery; //临时数据集
  Begin
    Try
  ADOQTemp :=
  TADOQuery.Create(Application);
  //创建临时数据集
  ADOQTemp.MaxRecords := (PageNum - 1) * 20;
  //设定临时数据集的记录数为(PageNum - 1) * 20
  ADOQTemp.Active := false;
  ADOQTemp.SQL.Clear;
  ADOQTemp.SQL.Add(
    'Select KeyID from TableName');
  ADOQTemp.Active := true; //获取临时数据集
  ADOQTemp.Last; //数据集指向最后一条记录
  KeyValue :=
  ADOQTemp.Fields[0].AsInteger;
  //获取指定页的前一页的最后一条记录的值
    ADOQTemp.Free; //释放临时数据集
  ADOQGData. MaxRecords := 20;
  //设定每页记录数为20
  //****设定ADOQGData的SQL，查询条件为***
  //****关键字段KeyID大于KeyValue的记录****
  ADOQGData.Active := false;
  ADOQGData.SQL.Clear;
  ADOQGData.SQL.Add(
      'Select KeyID, Fd1, Fd2, Fd3 ' +
  'from TableName where ' +
  'KeyID > ' + IntToStr(KeyValue) +
  ' ASC');
  ADOQGData.Active := true;
  // ************************
  Options := [grMetaData, grReset];
  //设置获取数据时的选项，要获得元数据
  vDatas := dspGData. GetRecords(
  -1, RecsOut, Byte(Options)); //获取数据
    SetComplete; //提交COM+事务
  Except
    SetAbort; //回滚COM+事务
  End;
 End;
```

GetAPage方法的PageNum参数是要获取的数据的页码。该方法首先通过一个TADOQuery对象ADOQTemp获取指定页的前一页的最后一条记录的值KeyValue，然后向数据库发出SQL命令，获取KeyID大于KeyValue的前20条记录，最后用vDatas参数向客户端返回数据集。

在客户端，通过一个GetGeneralData过程完成对以上3个远程过程的调用，代码示例略。

3.4 获取和处理详细数据

在中间层，建立一个提供详细数据的COM+对象ComtsDetailData，该对象通过接口ImtsDetailData(它的实现类为TmtsDetailData)向客户端提供GetDetailData方法，实现向客户端提供详细数据的功能。这个方法通过TADOQuery对象ADOQDetailData向数据库发出SQL命令，数据库返回的数据集由TDataSetProvider对象dspDetailData来维护。下面给出这个方法的实现。

```
  Procedure TmtsDetailData.GetDetailData(
    KeyValue: Integer; Var vDatas: OleVariant);
```

```
Var
  RecsOut: Integer; //获取数据时发生错误的次数
  Options: TGetRecordOptions; //获取数据时的选项
Begin
  Try
ADOQDetailData.Active := false;
ADOQDetailData.SQL.Clear;
ADOQDetailData.SQL.Add(
'Select * from TableName where ' +
'KeyID = ' + IntToStr(KeyValue));
//SQL语句的查询条件为关键字段等于KeyValue
ADOQDetailData.Active := true;
Options := [grMetaData, grReset];
//设置获取数据时的选项，要获得元数据
vDatas := dspDetailData. GetRecords(
-1, RecsOut, Byte(Options)); //获取数据
    SetComplete; //提交COM+事务
  Except
    SetAbort; //回滚COM+事务
  End;
End;
```

在客户端，通过一个GetDetailData过程完成对远程过程GetDetailData的调用，代码示例如下。

```
...
Var
  CdsDetailData: TClientDataSet;//客户端数据集
...
Procedure GetDetailData;
Var
  DetaildataObj: ImtsDetailData; //声明详细数据接口
  vDatas: OleVariant; //用于返回数据集的变体
  KeyValue: Integer;
Begin
  DetaildataObj := ComtsDetailData.CreateRemote('Svr');
//实例化ImtsDetailData接口，假设远程对象位于服务器Svr
  KeyValue := CdsGData.Fields[0].AsInteger;
//以概要数据的当前记录的关键字段的值作为KeyValue
  DetaildataObj.GetDetailData(KeyValue, vDatas);
//调用COM+远程过程，获取详细数据，获取的数据返回给vDatas
  CdsDetailData.Data := vDatas;
//将vDatas中的数据赋给客户端数据集
End;
```

当CdsDetailData所维护的数据集发生了增加删除、新建和更改等操作，要把这些操作反映到后台数据库中，可以调用CdsDetailData的ApplyUpdates方法，Borland公司的MIDAS机制会自动的生成相应的SQL命令，完成对数据库的处理。

3.5 基于ADO技术的局限性

前文已经述及，ADO实际上是一组COM对象，所以本文所介绍的方法实际上不适合在基于CORBA的系统中使用，因为这涉及到COM与CORBA的互操作。当然，目前COM与CORBA的互操作技术已经相当成熟，但这种互操作本身要占用大量的计算资源，那么就抵消了分段存取所带来的积极影响，甚至得不偿失。

如果ADO以外的数据库引擎，能够提供类似ADO中的MaxRecords功能，而且这样的数据库引擎能够在CORBA环境中使用，那么分段数据存取的方法仍然可以移植到基于CORBA的系统中。

4 结束语

随着分布式应用系统的体系结构逐步向WebService演进，分布式应用系统执行效率的矛盾将越来越突出。本文所介绍的基于ADO技术的分段数据存取方法，符合“少量多次”和无状态对象的原则，是提高分布式应用系统执行效率的有效手段之一。由于ADO是COM对象，所以它不适合在基于CORBA的系统中使用。但随着数据库引擎技术的发展，分段数据存取的方法也可以被移植到不涉及ADO技术的领域，从而具备更好的适用性。

参考文献

1 施明辉、孙荣胜，用基于XML的SOAP机制构建应用系统[J]，计算机应用，2002，22(4)：80-83。

2 李维，Delphi 5.x ADO/MTS/COM+ 高级程序设计篇[M]，机械工业出版社，2000。