网格环境下基于OGSA-DAI的异构数据集成研究

罗清地,蒋秀凤
(福州大学 数学与计算机学院,福建 福州 350002)

摘 要:OGSA-DAI(Open Grid Services Architecture-Data Access and Integration)致力于构造通过网格访问和集成来自不同孤立数据源的中间件,符合基于OGSA的网格标准。文中介绍了OGSA-DAI的框架模型和体系结构,分析了实现数据访问、集成的原理。通过具体的实例,详细介绍了OGSA-DAI的开发和应用过程。

关键词:网格数据服务;数据服务资源;数据访问与集成
中图分类号:TP311 **文献标识码**:A **文章编号**:1673-629X(2009)03-0144-04

Research on Isomerous Data Integration Based on OGSA-DAI in Grid Environment

LUO Qing-di, JIANG Xiu-feng
(College of Mathematics and Computer Science, Fuzhou University, Fuzhou 350002, China)

Abstract: The OGSA-DAI project is concerned with constructing middleware to assist the access and integration of data from separate data sources via the grid. OGSA-DAI is compliant with OGSA based grid standards. Introduces the frame model and architecture of OGSA-DAI, analyses the principle of data access and integration, the development and application of OGSA-DAI were illustrated by an example.

Key words: grid data service; data service resource; data access and integration

0 引 言

随着计算机网络和数据库系统的迅速发展,企业竞争与兼并的加剧,多样化新技术的采用,使得信息资源的异构性在企业的信息系统中无处不在,越来越多的应用需要访问各种异构数据源。任何企业应用首先需要解决的,就是如何对企业中的异构数据源进行集成和一致化处理,形成标准、统一和可靠的数据源,作为应用系统的基础。而为了达到异构数据源的共享,必须先解决异构数据源集成与转换问题。

OGSA-DAI是在GLOBUS平台上建造的通过网格访问和集成不同孤立数据源的中间件,提供对现有的、自主管理数据库的访问。基于OGSA体系提供网格数据库服务,可使网格用户或其它网格服务通过网格数据库,访问网格中的各种异构数据库,达到数据资源共享和协同处理的目的,满足虚拟组织对数据处理的需求[1]。

1 OGSA-DAI体系结构

OGSA-DAI提供了将现有数据资源,如关系数据库和XML数据库集成到网格环境中的基本架构。OGSA-DAI的体系结构根据不同的功能分为四层[2],分别是数据层、业务逻辑层、表示层和客户端层,每个层都有它特定的功能和目的,见图1。

(1)数据层:数据层由数据资源组成,这些数据资源通过OGSA-DAI对外暴露。

(2)业务逻辑层:这一层封装了OGSA-DAI的核心功能。它包括引擎(Engine)和活动(activity)两部分。其中引擎检查表示层传递的执行文档,并将执行文档分解为活动。在OGSA-DAI中,活动可以分为查询类(Query)、表示转换类(Transform)以及传输类(Delivery)三类。其中查询类活动主要是处理与数据源的交互(查询和更新数据),表示类活动主要是将查询类活动返回的结果表示为指定的格式或进行压缩处理,传输类活动主要是处理数据输出,将经过表示类活动处理后的数据发布给第三方(如Web服务器、文件系统或网格FTP服务器)等。业务逻辑层主要是检查

收稿日期:2008-07-31
基金项目:福建省教育科研基金资助项目(JA04161);福建省发展改革基金资助项目(SX2004-29)
作者简介:罗清地(1983-),男,硕士研究生,研究方向为异构数据库集成、分布式查询;蒋秀凤,副教授,主要研究方向为网格技术。

并处理执行文档(将其中与数据源相关的活动提交给数据层来执行)并生成响应文档。

数据层与业务逻辑层间通过数据资源访问器(Data Resource Accessor,DRA)进行通信,每一个数据服务都有它自己的数据资源访问器,这个数据资源访问器用于控制对内部资源的访问。

(3)表示层:这一层封装了使用 Web 服务接口暴露数据服务资源所需要的功能。OGSA－DAI 包括两个实现,一个和 WSRF 兼容,一个和 WSI 兼容。对于每一种实现,都有一个 WSDL 文档描述该接口。

(4)客户端层:一个客户端能够通过相应的数据服务和数据服务资源交互。根据发布的数据服务是 WSRF 还是 WSI,客户端的应用程序必须和 WSRF 或 WSI 标准兼容。OGSA－DAI 也包括一个 Java 客户端工具包,该工具包提供了用于和数据服务交互的高层 API。这个客户端工具包通过提供装配和发送请求和解释响应的方便方法,简化了客户端应用程序的开发。客户端工具包的另一个好处是它提供的在 WSRF 和 WSI 数据服务之间的相互可操作性。一个使用客户端工具包写的应用将能够透明地与 WSRF 和 WSI 数据服务访问和交互。

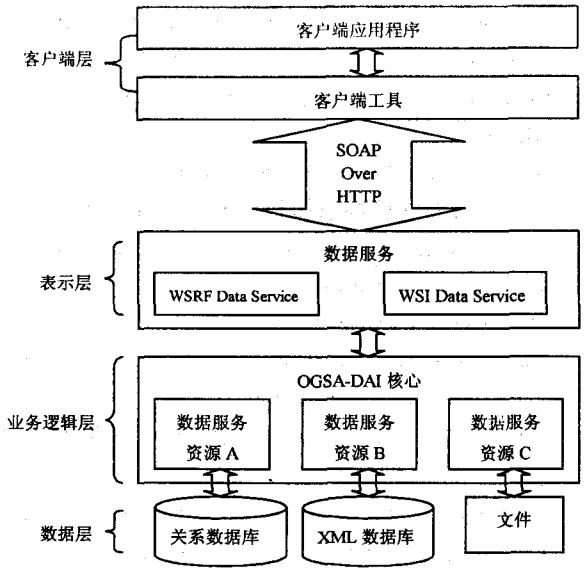

图 1　OGSA－DAI 体系架构

2　OGSA－DAI 工作原理

2.1　OGSA－DAI 工作流程

OGSA－DAI 的工作流程如图 2 所示。一旦 OGSA－DAI 启动,工厂就随之启动,然后用注册器对数据源进行注册,并能通过预先配置文件中的静态信息和配置文件提供的 MetaDataExtractor 类能访问到服务数据。客户机在服务列表中确定在注册器中列出的众多服务源应该使用哪一个。一旦选定合适 Data Service Resources,客户机就请求工厂创建一个 GDS 实例,以访问特定的数据资源。现在,GDS 已经准备好了,可以接收执行文档,运行数据库查询,传输查询结果和传送数据[3]。

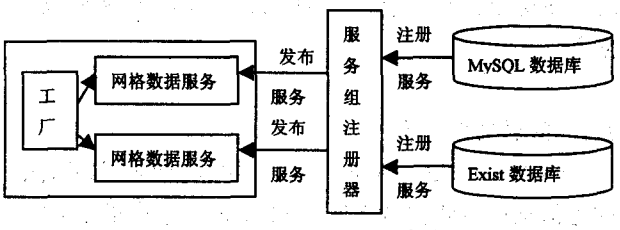

图 2　OGSA－DAI 工作流程

2.2　和数据服务资源交互

OGSA－DAI 支持通过一个面向文档的接口和数据服务资源交互。客户端没有直接和数据服务资源对话,而是发送执行文档给数据服务。数据服务然后继续将这个文档交给表示实际数据资源的数据服务资源。数据服务资源解释执行文档并且执行文档中的动作。这些动作在某些方面可能包含和内部数据资源的交互,例如通过执行一个 SQL 查询语句。一个描述请求结果的响应文档然后被数据服务资源生成并且通过数据服务发送给客户端。过程如图 3 所示。

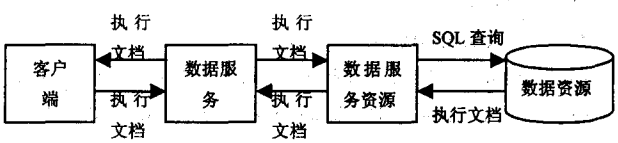

图 3　和数据资源交互过程

数据服务资源能够处理多个并发的请求。当达到并发的极限时,数据服务资源开始将后面的请求排队[4]。当一个数据服务被发布时,并发的极限和队列的最大长度被指定,这些值以后可以通过 Web 服务发布描述器进行调整。OGSA－DAI 交互模型的主要组件:

* 活动－ 数据服务资源能够执行的操作,包括数据资源操作、数据转换和数据传输操作。

* 会话－跨多个请求存储在数据服务资源中状态的对象。

* 执行文档－ 被客户端用于描述他们希望数据服务资源执行的活动。

* 响应文档－ 描述给客户端被数据服务资源处理执行文档的结果。

3　OGSA－DAI 实现异构数据访问与集成

OGSA－DAI 项目是基于 Java 实现的网格环境下数据访问与封装的中间件,为访问目前流行关系数据库(通过 JDBC)和 XML 数据库(通过 XMLDB)提供了

大量的接口,能以多种不同的方式查询、操作、格式化数据集以及传送数据且屏蔽各种数据源的差异。OGSA-DAI 被设计成易于用户扩展,使用户能方便地为特殊的数据源开发满足需要的访问集成方法及其它的附加功能,并通过编写相应的 Activity 类来操纵数据[5]。

3.1 数据文件配置

OGSA-DAI-WSRF-2.2 中,与数据源相关的配置信息被存储在/WEB-INF/etc/目录中的 dataResourceConfig.xml 和 DatabaseRoles.xml 文件中。dataResourceConfig.xml 文件中定义了有关数据源连接的一些信息,例如数据源驱动、URI 等。主要内容如下:

```
<!--数据源的元数据信息,用于描述数据源的产品信息-->
<metaData>
  <relationalMetaData>
    <databaseSchema callback="uk.org.ogsadai.datare-source.MySQLMetaDataExtractor"/>
  </relationalMetaData>
</metaData>
<!--指定角色映射文件-->
<roleMapname="Name"implementation="uk.org.ogsadai.common.rolemap.SimpleFileRoleMapper"Configuration="/usr/local/tomcat-4.1.3/webapps/wsrf/WEB-INF/etc/ogsadai_wsrf/MySQLResource/DatabaseRoles.xml/">
<!--指定数据源连接相关信息-->
  <dataResource>
    <driver implementation="org.mysql.jdbc.Driver">
    <uri>jdbc:mysql://localhost:3306/ogsadai</uri>
  </driver>
</dataResource>
```

角色映射文件 DatabaseRoles.xml 文件定义了有关用户名,密码的信息。内容如下:

```
<!--指定访问数据源的用户名与密码-->
<Database name="jdbc:mysql://localhost:3306/ogsa-dai">
    <User dn="*"userid="root"password=""/>
</Database>
```

3.2 异构数据访问与集成实例

根据 OGSA-DAI 的提供的异构数据访问接口,以 eXist 数据库和 MySQL 数据库集成为例,把 eXsit 数据库里面的文档 littleblackbook 插入到 mysql 数据库表 mytable 中,设计异构数据集成的步骤如下:

(1)定义数据服务并选择数据数据资源:

```
String handle = "http://localhost:8080/wsrf/services/ogsadai/DataService";
String sinkID = "MySQLResource";
String sourceID = "eXistResource";
DataService sinkService = GenericServiceFetcher.getInstance().getDataService(handle, sinkID);
DataService sourceService = GenericServiceFetcher.getInstance().getDataService(handle, sourceID);
```

(2)在 MySQL 数据库中创建新表 mytable:

```
String tableName = "mytable";
String createTable = "create table if not exists" + tableName + "(id INTEGER, name VARCHAR(64)," + "address VARCHAR(128), phone VARCHAR(20))";
SQLUpdate create = new SQLUpdate( createTable );
sinkService.perform( create );
```

(3)为源数据资源请求创建 session,把从 eXist 数据库中查询到的数据转换成结构化数据:

```
ActivityRequest request = new ActivityRequest();
request.setSessionRequirements(new JoinNewSession());
Response response = sourceService.perform(request);
Session session = response.getSession();
XPathQuery query = new XPathQuery ("/entry[@id<500]");
XSLTransform transform = new XSLTransform();
transform.setXMLInput( query.getOutput() );
DeliverFromURL deliver = new DeliverFromURL( url );
transform.setXSLTInput( deliver.getOutput() );
OutputStreamActivity outputStream = new OutputStreamActivity();
outputStream.setInput(transform.getOutput());
```

(4)把所有活动都添加到 source request 中:

```
ActivityRequest sourceRequest = new ActivityRequest();
sourceRequest.add( deliver );
sourceRequest.add( query );
sourceRequest.add( transform );
sourceRequest.add( outputStream );
```

(5)开始 session 请求,把获得的数据插入到 mytable 表里面:

```
DeliverFromDT deliverFromDT = new DeliverFromDT();
deliverFromDT.setDataTransportInput ( outputStream.getDataTransport());
deliverFromDT.setDataTransportMode( DataTransportMode.BLOCK );
SQLBulkLoad bulkload = new SQLBulkLoad ( deliverFromDT.getOutput(), tableName );
ActivityRequest sinkRequest = new ActivityRequest();
sinkRequest.add( deliverFromDT );
sinkRequest.add( bulkload );
sinkService.perform(sinkRequest);
bulkload.getInsertedRowsCount();
```

启动网格服务,在客户端运行命令:ant -f build-

examples.xml runClient －Ddai.class=DataIntegration.

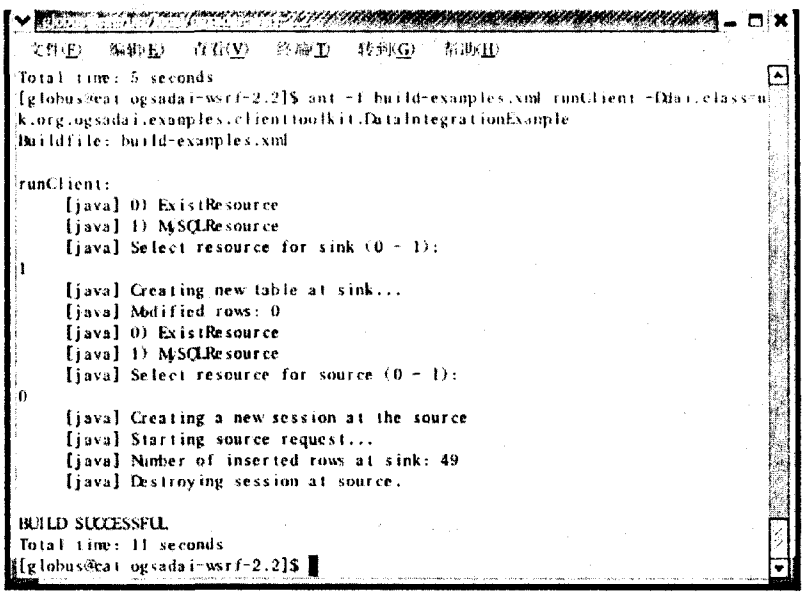

图 4　数据集成过程

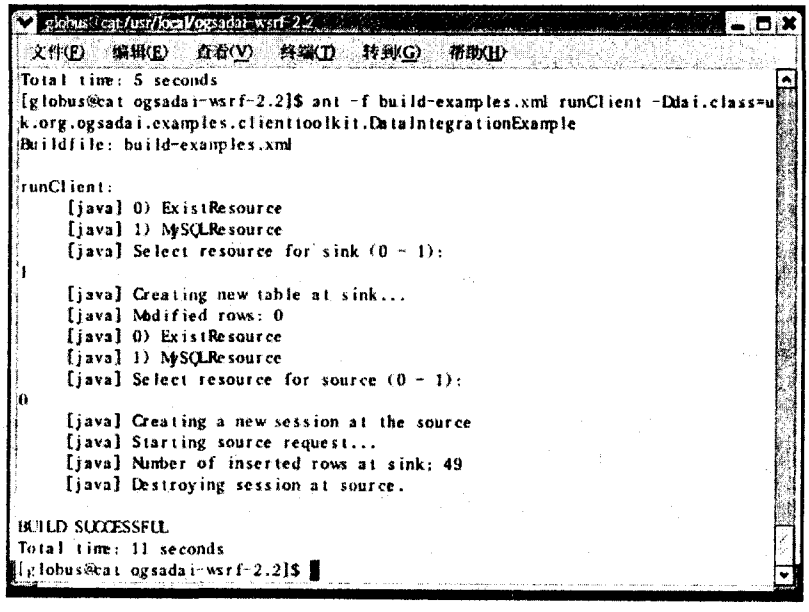

图 5　集成查询结果

class。在客户端看到的数据集成过程与结果如图 4、图 5 所示。

4　结束语

基于网格的异构数据源集成,从访问数据库的角度看,在网格环境下访问数据库的方式和在网格环境之外访问数据库的方式相似,但是利用网格开发工具可以屏蔽各个业务节点数据库的结构、运行环境上的差异、网络分布状况和具体的物理位置,保证各个节点数据库的独立性和数据的安全。相比较其它异构数据源的集成环境这是它的优势所在。随着网格技术的进一步完善与规范,数据网格的应用会越发的成熟。

参考文献:

[1] 任　浩,李志刚,肖　依.数据库网格:基于网格的多数据库系统[J].计算机工程与应用,2006(2):172－175.

[2] OGSA－DAI WSRF 2.2 User Guide [EB/OL]. 2006 － 05. http://www.ogsadai.org.uk/documentation/ogsadai－wsrf－2.2/doc/.

[3] 南　凯,阎保平.扩展 OGSA－DAI 的数据集成框架及原型[J].计算机工程,2007,33(10):55－57.

[4] 范会联.网格环境下数据访问与集成方法研究[J].信息工程大学学报,2007,8(1):11－14.

[5] 金宝轩.网格环境下的异构空间数据库集成技术[J].计算机工程,2008,34(5):74－76.

(上接第 143 页)

4　结束语

对 Aggarwal 和 Yu 近期提出的基于子空间投影和遗传算法(GA)的离群点检测方法进行了改进,使检测的结果更加有效,但在变异过程中增加了一个变异候选集,相应地增加了时间复杂度,这将使算法有待进一步改进。

参考文献:

[1] Agrawal R, Gehrke J, Gunopulos D, et al. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications[C]//Haas L M, Tiwary A. Proc. of the ACM SIGMOD International Conference on Management of Data. Seattle: ACM Press, 1998:94－105.

[2] Hawkins D. Identification of Oudiers[M]. London: Chapman and Hall, 1980.

[3] 黄洪宇,林甲祥,陈崇成,等.离群数据挖掘综述[J].计算机应用研究,2006,8(6):8－9.

[4] Aggarwal C C, Yu P S. An Effective and Efficient Algorithm for High－dimensional Outlier Detection[J]. The VLDB Journal, 2005, 14(2):211－221.

[5] Aggarwal C C, Yu P S. Outlier Detection for High Dimensional Data[M]. [s.l.]:ACM, 2001.