分类号_____ 密级_____

UDC _____ 编号_____

中国科学院研究生院

硕士学位论文

虚拟天文台数据访问服务（VO-DAS）客户端
设计与实现

杨 阳

指导教师 赵永恒 研究员、 崔辰州 副研究员

中国科学院国家天文台

申请学位级别 硕 士 学科专业名称 天文技术与方法

论文提交日期 2008 年 4 月 论文答辩日期 2008 年 5 月

培养单位 中国科学院国家天文台

学位授予单位 中国科学院研究生院

答辩委员会主席 邓李才 研究员

National Astronomical Observatories
Chinese Academy of Science

The Design and Implementation of Client System of Virtual Observatory Data Access Service

Yang Yang

Advisor:

Prof. Yong-Heng ZHAO

Asso. Prof. Chen-Zhou CUI

National Astronomical Observatories, Chinese Academy of Sciences

Beijing 100012, P.R. China

April 2008

摘　要

伴随观测技术的进步，天文观测数据如同雪崩一样高速地产生出来。旨在将各种天文研究资源以某种统一的服务模式无缝地集成在系统中的虚拟天文台便应运而生。虚拟天文台数据访问服务（VO-DAS）基于新兴的数据网格技术，实现了异地异构海量天文数据统一联合资源访问的简单、实用和规范的操作接口。但是这些接口仅限于程序使用，最终用户难以直接访问它们。为了最大程度地发挥 VO-DAS 系统的效率和使用潜力，让它来为天文学的研究带来更大的效益，本文基于 VO-DAS 的接口，设计并实现了用户访问分布存储的异构天文数据资源的两种方式：GUI 和命令行，它们分别满足了初级用户和高级用户对 VO-DAS 数据访问请求。其次，针对目前 VO-DAS 系统中生成数据结点过程繁琐的现状，本文设计并实现了数据结点配置工具。对于拥有海量观测数据的用户，使用此工具就可将数据轻易地投放到 VO-DAS 系统中供天文学界的同仁共享，从而提高生成数据结点的工作效率。最后，作为数据访问必不可少的环节，本文对 VO-DAS 系统中既相对独立又相互联系的服务部署问题进行了讨论。通过这一系列的工作，促进了 VO-DAS 系统的实现并尽可能地发挥出它的使用价值，对于推动中国虚拟天文台的工作具有非常重要的现实意义。

关键词：中国虚拟天文台 数据访问 数据网格 客户端 数据结点配置

Abstract

With the development of observational technologies, astronomical data have been generated at high speed, just like avalanche. Therefore the Virtual Observatory has come into being, which aims to integrate various astronomy research resources into one system in a unified way. Based on the emerging technology of Data Grid, Virtual Observatory Data Access Service (VO-DAS) has achieved simple, practical and standardized interfaces which access distributed, heterogeneous and vast data resources. Although these interfaces are only provided to the application of programs, it is very difficult for the end user to directly visit the interfaces. In order to show efficiency and potential of VO-DAS system and make it more beneficial for the study of astronomy, we designed and implemented two ways to visit VO-DAS: GUI and Command Line, which meet primary and senior users, respectively. Then we also designed and developed Data Node configuration tool so as to deal with complex operation of Data Node in the VO-DAS system. As users with vast observational data can easily put their data in the VO-DAS system and conveniently share the data with others, thus this improves the efficiency of generating Data Node. At last, we discussed how to deploy each independent and interdependent service of VO-DAS as an absolutely necessary step to access data. A series of our work has promoted the implementation and realization of the VO-DAS system as soon as possible, this has been of very important practical significance to the development of the China-VO project.

Key Word: Virtual Observatory Data Access Data Grid Client Part Data Node

目 录

图表目录

第一章 引言

1.1 虚拟天文台的背景与现状

1.1.1 虚拟天文台产生的背景

近几十年以来，天文观测技术经过几个阶段性的发展获得了长足的进步。最初，照相技术和光谱观测技术在天文观测中的应用结束了人眼作为唯一的天文观测手段。几十年前，随着无线电技术的蓬勃发展，宇航时代的到来，空间天文学的产生，天文观测的范围扩展到了γ射线、X射线、紫外和红外波段，这使得天文学观测进入了全电磁波观测的时代。近十年来，望远镜技术和大尺寸探测器技术已经取得了突破性的进展。目前天文探测器技术正迈向超导器件的时代，天文学的发展已经出现了崭新的前景。伴随观测技术的进步，天文观测数据如同雪崩一样高速地产生出来。面对如此巨大的数据量，如何从数十亿天体的多波段海量数据中探求科学的发现，如何才能有效地访问、处理和分析这些数据，已经成为天文学家亟待解决的问题。

幸运的是，与此同时，特别是近十几年，计算机技术和互联网技术也飞速地发展起来，网格技术、XML 技术、WEB 服务技术等新兴 IT 技术日趋成熟。这为解决天文学家基于海量数据的研究课题提供了技术手段，因此，虚拟天文台(Virtual Observatory, VO)应运而生。VO 是一个密集型在线天文研究和教育环境，它利用先进的信息技术实现对全球天文信息无缝的访问[1]。虚拟天文台将把世界上的各种天文研究资源，包括巡天观测数据、天文文献、计算资源、数据处理工具、天文观测设备等以某种统一的服务模式被无缝的汇集在 VO 系统中，使 VO 真正成为一个数据密集型的在线研究平台。天文学家只需要登陆到 VO 系统便可以享受其提供的丰富资源和强大的服务，使自己从数据收集、数据处理这些繁琐的事务中彻底摆脱出来。如果说利用γ射线、X射线、紫外巡天、光学巡天、红外巡天和射电巡天所得到的观测数据，用适合的方法对数据进行统一规范的整理、归档，便可以构成一个全波段的数字虚拟天空；而根据用户要求获得某个天区的各类数据，就仿佛是在使用一架虚拟的天文望远镜；如果再根据科学研究的要求开发出功能强大的计算工具、统计工具和数据挖掘工具，这就相当于拥有了虚拟天文台的各种探测设备。这样，由虚拟数字天空、虚拟的望远镜和虚

拟的探测设备所组成的机构便是一个独一无二的虚拟天文台[2]。

虚拟天文台将使天文学取得前所未有的进展，它将成为开创“天文学发现新时代”的关键性因素[3]。它将 TB 甚至 PB 量级的数据库、波长遍及从 γ 射线到射电波段的数十亿个天体的图像库、高度复杂的数据挖掘和分析工具、具有数千 PB 量级容量的存储设备和每秒运算万亿次的超级计算设备、以及各级主要天文数据中心之间的高速网络连成一体；它能使世界各地的天文学家可以快速查询每个 PB 量级的数据库；使隐藏在庞大星表和图像数据库中的多变量模式可视化；增加发现复杂规律和稀有天体的机会；能够对大样本星表进行数据挖掘和统计分析研究工作。

1.1.2 VO 国内外发展现状

虚拟天文台的概念提出来以后，世界各国纷纷提出自己的虚拟天文台的计划项目。这些计划项目都在力求寻找数据密集型天文研究的出路，力图挖掘海量天文数据的潜力。如果把这些各具特色的项目联合起来，共同迎接虚拟天文台所面临的科学与技术挑战，如：不同部分的互操作性，数据和接口的标准化，软件包、源代码库、开发工具等之间的协调，就需要一种机制来促进国际上不同项目之间的合作。因此，2002 年 6 月成立了国际虚拟天文台联盟（IVOA）[4]。IVOA 的重要使命就是推进国际合作与协作，为建设一个能综合利用国际天文台数据的、完整的、能协作工作的虚拟天文台，开发、配置必要的工具、系统和组织结构[2]。截至到现在，历经五年多时间，各国虚拟天文台都希望肩负起 IVOA 的重要使命，目前 IVOA 成员已增加至 16 个，如表 1.1 所示。

IVOA 中的各国虚拟天文台相互借鉴和促进，分别取得了卓有成效的成绩。例如：英国虚拟天文台建立了一套体系架构较为成熟的虚拟天文台系统 AstroGrid[5]。美国国家天文台（NVO）先后提出了 SkyQuery 原型[6]和 OpenSkyQuery 原型[7]。法国的 Centre de Données astronomiques de Strasbourg（CDS）设计了一个完整的天文资料的访问系统[8]等等。

以中国科学院国家天文台为首的国内天文学家于 2002 年初提出建设“中国虚拟天文台”（China-VO）[9]的设想。同年 10 月，中国虚拟天文台成为国际虚拟天文台联盟成员。China-VO 的产生除了有国际上各国虚拟天文台积极响应的这个大背景外，同时，它的发展还获得了国内两大驱动力。首先，China-VO 与大科学工程 LAMOST[10]紧密结合，建设“VO-enabled LAMOST”，充分发挥 LAMOST 光学光谱数据中心的作用。LAMOST 的需求带动了 China-VO 的发展。此外，China-VO 同时也处于中国科学信息化（e-Science）的大浪潮中。2004 年

表 1.1 各国虚拟天文台项目

项目	地区	网址
亚美尼亚虚拟天文台（ArVO）	亚美尼亚	http://www.aras.am/ArVO/arvo.htm
英国虚拟天文台（AstroGrid）	英国	http://www.astrogrid.org/
澳大利亚虚拟天文台（Aus-VO）	澳大利亚	http://www.aus-vo.org/
中国虚拟天文台（China-VO）	中国	http://www.china-vo.org
加拿大虚拟天文台（CVO）	加拿大	http://services.cadc-ccda.hia-iha.nrc-cnrc.gc.ca/cvo/
欧洲虚拟天文台（Euro-VO，AVO）	欧洲	http://www.euro-vo.org/
德国天体物理虚拟天文台（GAVO）	德国	http://www.g-vo.org/
匈牙利虚拟天文台（HVO）	匈牙利	http://hvo.elte.hu/en/
日本虚拟天文台（JVO）	日本	http://jvo.nao.ac.jp/
韩国虚拟天文台（KVO）	韩国	http://kvo.kasi.re.kr/
国家虚拟天文台（NVO）	美国	http://www.us-vo.org
法国虚拟天文台（VO-France）	法国	http://www.france-vo.org/
俄罗斯虚拟天文台（RVO）	俄罗斯	http://www.inasan.rssi.ru/eng/rvo/
西班牙虚拟天文台（SVO）	西班牙	http://svo.laeff.inta.es/
意大利虚拟天文台（DRACO）	意大利	http://vobs.astro.it
印度虚拟天文台（VO-India）	印度	http://vo.iucaa.ernet.in/%7Evoi/

China-VO 获得了自然科学基金委的资助，加速了它的发展速度。于 2004 年，国内科学家提出了完整的 China-VO 的体系结构[11]，如图 1.1 所示。

图 1.1 中国虚拟天文台体系结构

整体体系结构分为四层，从下至上依次为构造层、资源层、汇集层和用户层。

构造层包括各种数据资源、计算资源、网络资源和存储资源等。各种数据资源在虚拟天文台这样一个数据密集型在线研究平台中占有非常关键的作用，是 VO 成功运作的基础和前提。它主要包括星表、星图、光谱、时序数据、计数测量数据、模拟数据、多媒体数据、天文文献等。

资源层是以开放的网格服务架构（简称 OGSA）[12]为基础，配合其他网格系统服务工具，利用标准的数据模型和服务模型，通过抽象化实现统一的数据访问和计算访问以及网格系统管理等功能。系统管理主要涉及作业管理、安全管理、资源状态管理、数据管理等。

汇集层提供具有天文特色的各种服务，如数据处理、数据挖掘、统计分析、可视化等应用服务。

用户层是整个体系的最高层，包括 VO 客户端和 VO 门户，直接与虚拟天文台用户接触，用户层的基本职能是用户任务提交和处理结果返回，主要功能包括用户登录、身份认证、VO 资源浏览、任务编制和提交、结果显示、数据下载等。

China-VO 的体系结构建立在 OGSA 的基础之上。物理上，整个系统是分布式的，在网络环境下实现的；逻辑上，通过网格操作系统的管理，它是一个统一的整体。

在上述系统架构基础之上，以虚拟天文台科学目标为指导，China-VO 经过几年的努力，通过不断设计实现小型工具软件和中型应用程序已逐渐向一个天文应用平台靠近，开发出了各种各样的天文应用工具，如：2005 年发布的 VO_IMPAT[13]，它用于进行快速的星空位置查找和主要星表的联合查询，支持数字图像的可视化。2006 年发布的 SkyMouse 是一个桌面天文信息搜索工具，它可以通过鼠标选取屏幕上的词组，从多个天文信息数据库中查找对应名称的天体数据、图像和相关文献[14]。该工具简单实用，它可以让天文学家在阅读文献的时候快速浏览某个感兴趣的天体的数据信息。2007 年又完成了 FitHAS[15]和 VO-DAS[16]的初步开发。FitHAS 是一个批量读取 FITS 文件头并导入数据库的工具，该工具对于整理和管理 FITS 文件有很大的帮助。VO-DAS 是一个异地异构海量数据的统一访问服务，在下节中将会详细地予以介绍。

1.2 VO 数据访问与 VO-DAS

1.2.1 VO 数据访问

在各国天文数据中心存储的数据主要有三个特点：

1） 数据的分布性：天文数据存储在世界各地的数据中心，而不是集中统一存放在某一个数据结点上。这种分布存放的特征使得天文学家对它们的访问变得比较复杂，特别是当需要对来自不同数据库的数据进行交叉证认时，对数据的收集、数据格式调整、数据存放和数据处理将花费较长的时间。

2） 数据的异构性：目前天文数据的管理没有通用的标准，因此，天文数据库的存储方式、命名方式、单位和元数据标准都有很大的差异。当数据使用者使用来自不同来源的数据资源的时候，必须熟悉这些不同数据库的操作方法，毫无疑问，这对用户来说，是比较耗时耗力的过程。

3） 海量数据的访问：天文学发展至今，已经产生了数量巨大的天文观测数据，天文学的发展需要应用这些海量数据完成更加精确或新的发现。但是，目前尚没有统一解决海量数据访问的方案。如果采用将所需数据备份到本地机器上，当研究多个不同项目的海量数据的时候，使用这种方式的成本将会比较高。

基于天文数据存储的上述特点以及世界各国对天文资料管理方式存在的巨大差异，实现对异地异构数据资源的统一访问就成为虚拟天文台的基本功能，即定义全球统一的互操作标准和分布式数据的统一访问机制，以此来屏蔽数据源在数据格式、存储格式、主机环境、访问形式等诸多方面的异构性和复杂性。数据访问与互操作的实现在虚拟天文台中体现为一系列的服务。这些服务存在于数据源和上层应用之间，为上层应用提供统一各数据访问服务，同时进行数据格式转换，以此来增强数据的互操作能力，其原理好比沙漏模型[5]。如图 1.2 所示：

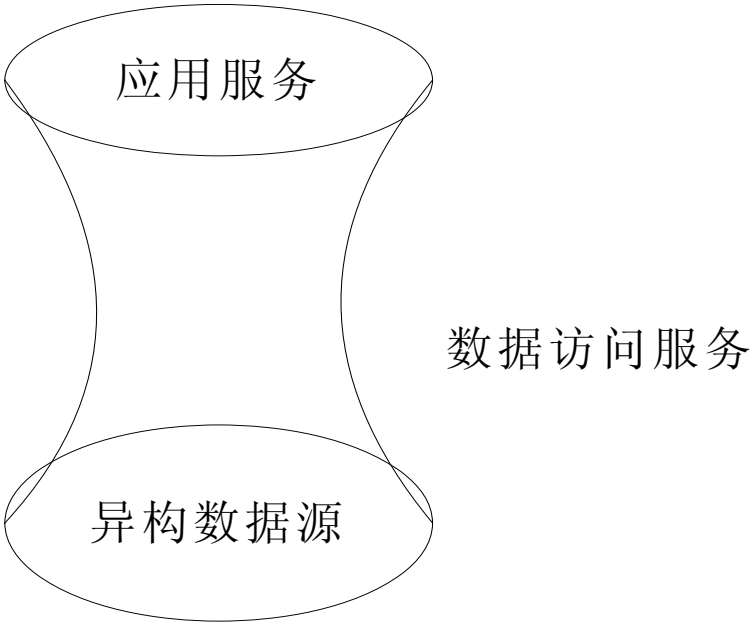

图 1.2 沙漏模型

这种访问机制不仅仅提供给天文学家获取数据和分析结果的便利，它还提供给用户的应用，使得程序也可以通过一个统一的访问接口来查询获取数据，从而实现全球范围内主要天文研究资源，不同数据集，不同数据服务间的互操作，特别是异地数据资源的统一、透明地访问与获取

1.2.2 中国虚拟天文台数据访问服务（VO-DAS）

中国虚拟天文台于 2006 年 5 月份提出了的数据访问服务（VO-DAS）的设计方案，主要目标是实现分布保存的异构数据的统一访问功能。它的设计主要实现如下一些功能：

1） 统一访问异地异构的天文数据库。数据用户不必了解数据资源的具体物理位置以

及数据资源的具体组织形式，通过统一的访问接口就可以获得他们期望的数据。

2） 支持天文数据的交叉证认或联合查询。系统提供一种机制把分散在不同地方的数据资源从逻辑上联系起来，实现联合查询或交叉证认。

3） 能够访问不同类型的数据资源，包括星表数据、图像数据和光谱数据。

4） 支持自动发现资源的功能。系统提供资源注册机制，天文数据一旦注册到系统中，就成为其可用的资源，系统就可为使用者找到需要的天文资源。

5） 能够访问海量的数据。系统支持一次访问上百万条的数据记录。

6） 支持多种结果数据存储格式。用户查询的结果数据可以保存成 VOTable、ASCII、CSV 等多种文件格式，并可以通过 FTP 自动传输到用户的服务器上，最大程度地满足使用者的要求。

VO-DAS 的这些功能很好的解决了虚拟天文台国内外数据访问的难题。同时，它还将成为中国虚拟天文台框架下天文数据共享的平台。

VO-DAS 的实现是一个建立在网格技术基础之上的异地异构天文数据访问平台。以 Globus Toolkit[17]网格系统为基础环境，利用 OGSA-DAI[18]提供的基本数据访问服务，结合国际虚拟天文台联盟（IVOA）制定的数据访问服务规范和本项目的实际需求，以 Web Service[19]和网格服务的形式实现对异地异构数据库系统、文件系统数据的访问功能。VO-DAS 体系结构[20]如图 1.3 所示。

它主要分成两个部分：DAS 和 Data Node。DAS 是 VO-DAS 的核心模块，它起到承上启下的调度功能。它是一个基于 Web 服务资源架构（简称 WSRF）[21]的任务管理服务。一方面它是一个用户作业调度和管理服务，另一方面它管理 Data Node，保证用户的数据查询请求找到正确的 Data Node，找到正确的数据资源。Data Node 是一个基于 OGSA-DAI WSRF 中间件的数据资源节点。数据（星表，图像或光谱）都通过 Data Node 提供给 DAS 服务。数据的拥有者不需要了解 DAS 和 Data Node 之间的通讯方法，只要按照要求将自己的数据配置到一个就近的 Data Node 之下即可发布到 DAS 服务上。

VO-DAS 服务器提供给客户端访问的接口有四类：资源元数据接口（RMI）、数据查询接口（DQI）、数据存取接口（DAI）和管理接口（MI）。RMI 用于获得 VO-DAS 发现的所有资源元数据。它提供如表 1.2 所描述的方法。DQI 用于发送数据查询请求，其接口方法如表 1.3 所述。DAI 用于访问查询任务的相关数据，它的相关方法见表 1.4 的描述。MI 是一个管理的接口，它的方法如表 1.5 所示。

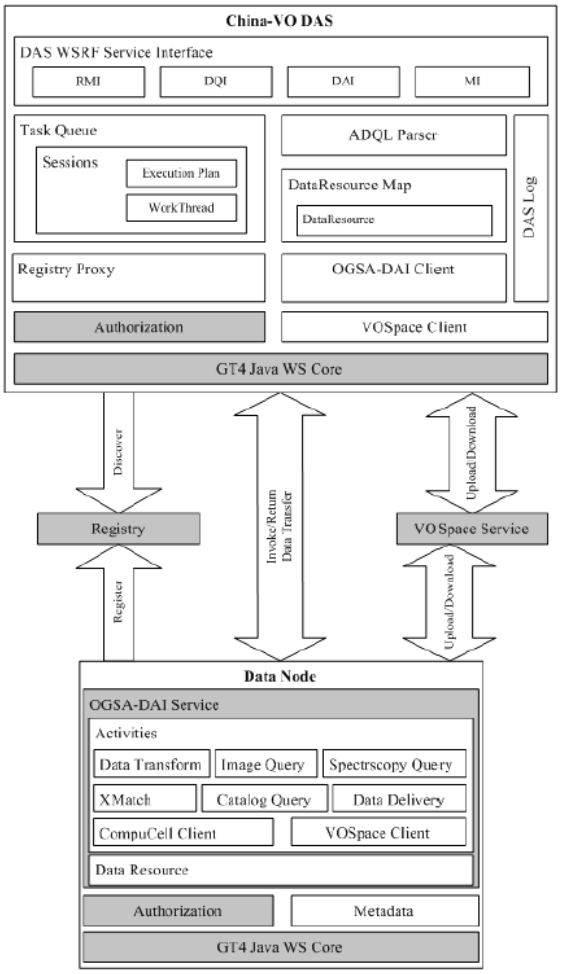

图 1.3 VO-DAS 体系结构图

1.3 VO-DAS 客户端系统研究的价值与意义

客户端（Client）或称为用户端，是指与服务器相对应，为客户提供本地服务的程序。一般安装在普通的客户机上，需要与服务端互相配合运行。对用户来说，较重要的一点就是客户端形式，即用户与客户端交互的方式。不同的交互方式将直接影响到用户对系统的使用效率。介于服务器与用户之间设计出的客户端既能发挥出服务器的潜能又能满足用户的需求是至关重要的。因此本论文基于VO-DAS服务设计与实现的三种客户端:GUI客户端、命令行客户端和Web客户端的目的就在如此。GUI客户端简单易用，适合初学者；命令行客户端以命令行的方式请求服务，适合高级用户；Web客户端直观方便，用户使用它无需任何环境设置。

表 1.2 RMI 接口定义

接口名称	参数	返回值	说明
GetAllResource		String	获得系统中 VOTable 形式的所有数据资源的元数据
GetMetaTables	String	String	获得指定资源名称下的表元数据
GetMetaColumn	Strin，String	String	获得指定资源名称指定表名下的列元数据

表 1.3 DQI 接口定义

接口名称	参数	返回值	说明
SynQuery	String（2）	Dataset	同步查询数据，查询结果直接返回
AsynQuery	String， Integer，String	Dataset	异步查询数据，查询结果存放在指定的位置

表 1.4 DAI 接口定义

接口名称	参数	返回值	说明
GetTargetURL	session	URL	获得查询结果文件的 URL
GetQueryResult	session	dataset	获得查询结果数据

表 1.5 MI 接口定义

接口名称	参数	返回值	说明
StartSession		session	开始一个新的 session
GetStatus	session	String	获得任务的当前状态
GetStartTime	session	Date	查询任务开始的时间
GetSubmitTime	session	Date	查询任务提交的时间
GetEndTime	session	Date	查询任务结束的时间
DestorySession	session		释放一个 session

1.3.1 数据访问客户端的现状

虚拟天文台发展至今，各国虚拟天文台都在使用已经比较稳定、成熟的虚拟天文台协议和比较成熟商用化的网络技术，发展了很多不同表现形式的虚拟天文台应用产品，为天文学家提供数据访问和管理的平台。

美国虚拟天文台开发的 Open SkyQuery[22]，它使用标准的查询语言 ADQL（Astronomical Data Query Language）[23]和天文数据节点 SkyNode 标准[24]，对外提供统一的 Web 服务接口，向用户提供了 Web 客户端的数据访问门户。如图 1.4 所示，用户可以采用简单查询和高级查询两种方式访问数据。由于它使用 Web Service 作为提供访问数据服务的模式，数据传输效率比较低。允许用户一次最多访问 5000 条记录，而且只能以同步的方式查询。

由 Sloan Digital Sky Server（SDSS）提供的 casjob 服务[25]，如图 1.5 所示，它提供了访问 Sloan 数据的客户端，允许用户注册登陆，并自动分配一个称为 MyDB 的虚拟数据库来保存用户查询的结果，它可以采用异步查询的方式即用户可以连续提交查询任务并且允许查询者断开连接，其访问的数据量是 500M。

由法国斯特拉斯堡天文数据中心（CDS）开发的桌面应用程序 ALADIN[26]如图 1.6，它用于可视化天文数据联合查询，通过网络获得多个波段的天体图片，并分层显示在用户界面上。同时 Aladin 使用了 IVOA 的 PLASTIC 协议[27]作为与其它应用程序协作的协议。

上述的天文数据访问客户端向用户提供的数据访问平台存在诸多不足，如 Open SkyQuery 提供给用户的访问方式受限，查询的数据量低，不提供异步查询。Casjob 分配给每个用户的空间有限，不提供异地数据查询等等。

1.3.2 VO-DAS 客户端的意义

中国虚拟天文台为了解决异地异构海量数据资源访问的问题，设计出了虚拟天文台数据访问服务（VO-DAS），VO-DAS 对外为客户端的网格应用产品提供了一些简单、直观的 Web 服务访问接口： RMI、DQI、DAI 和 MI。但是这些接口仅仅只限于程序使用，最终用户通过这些接口直接访问 VO-DAS 会是一种繁琐的工作。因此，如何根据 VO-DAS 定义的这些 Web 服务接口，设计出网格应用产品的客户端，使天文学家或数据用户通过客户端能够访问 VO-DAS，非常方便地获取自己想要的数据，是 VO-DAS 亟待解决的问题。另外一方面，如上节所示，各国虚拟天文台都在致力于开发出各具特色的数据服务和客户端，来服务于更多的天文学家，使科学家能够实际感受到信息化技术和服务给科学研究带来的真正效益。

China-VO 同样也不例外，，本文自主开发的客户端系统就是基于 VO-DAS 强大的数据访问能力，不仅借鉴国内外开发的经验，而且还弥补了数据访问客户端现状存在的一些不足，从而

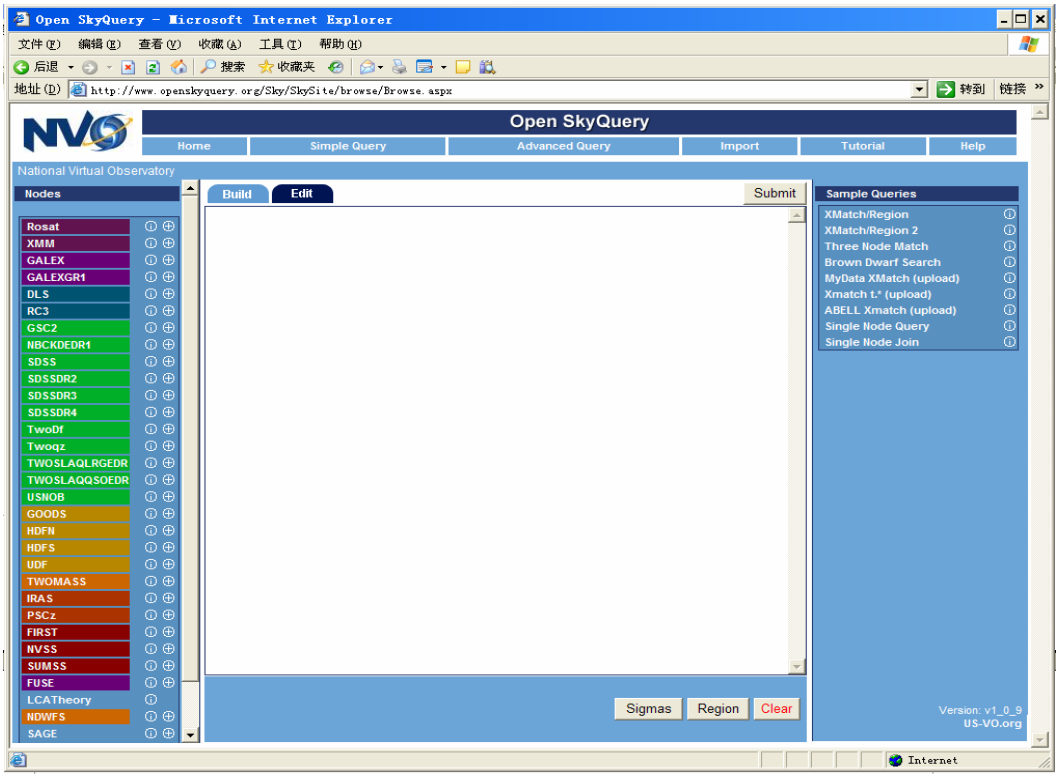

图 1.4 Open SkyQuery 天文数据访问客户端

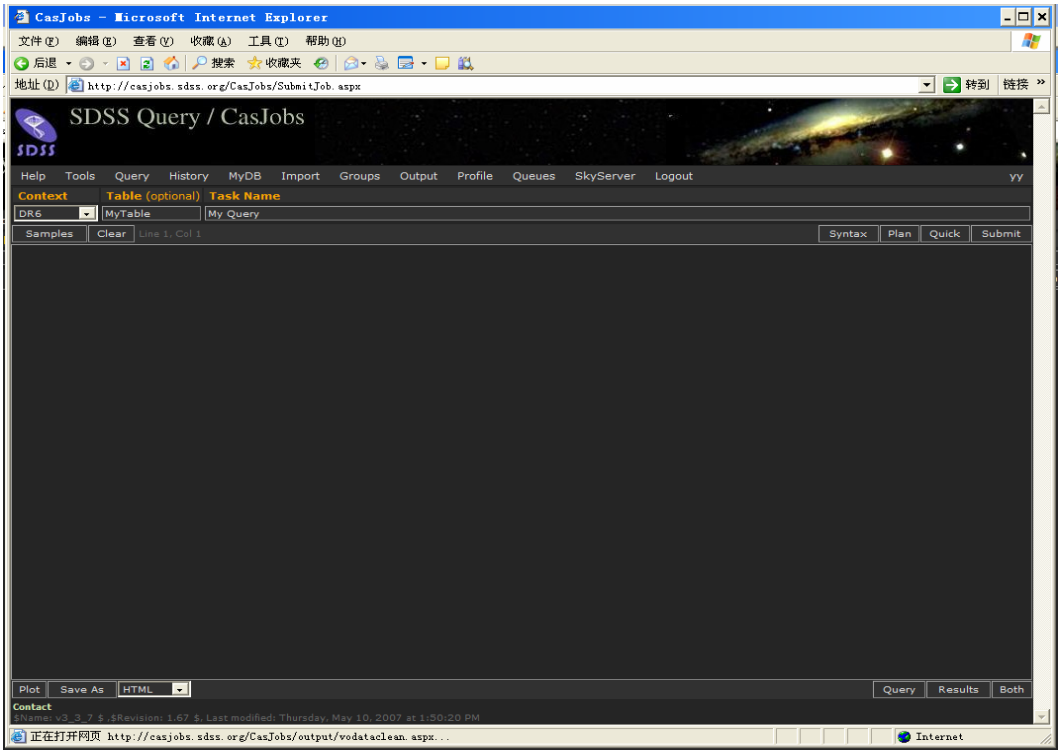

图 1.5 SDSS 的 casjob 客户端

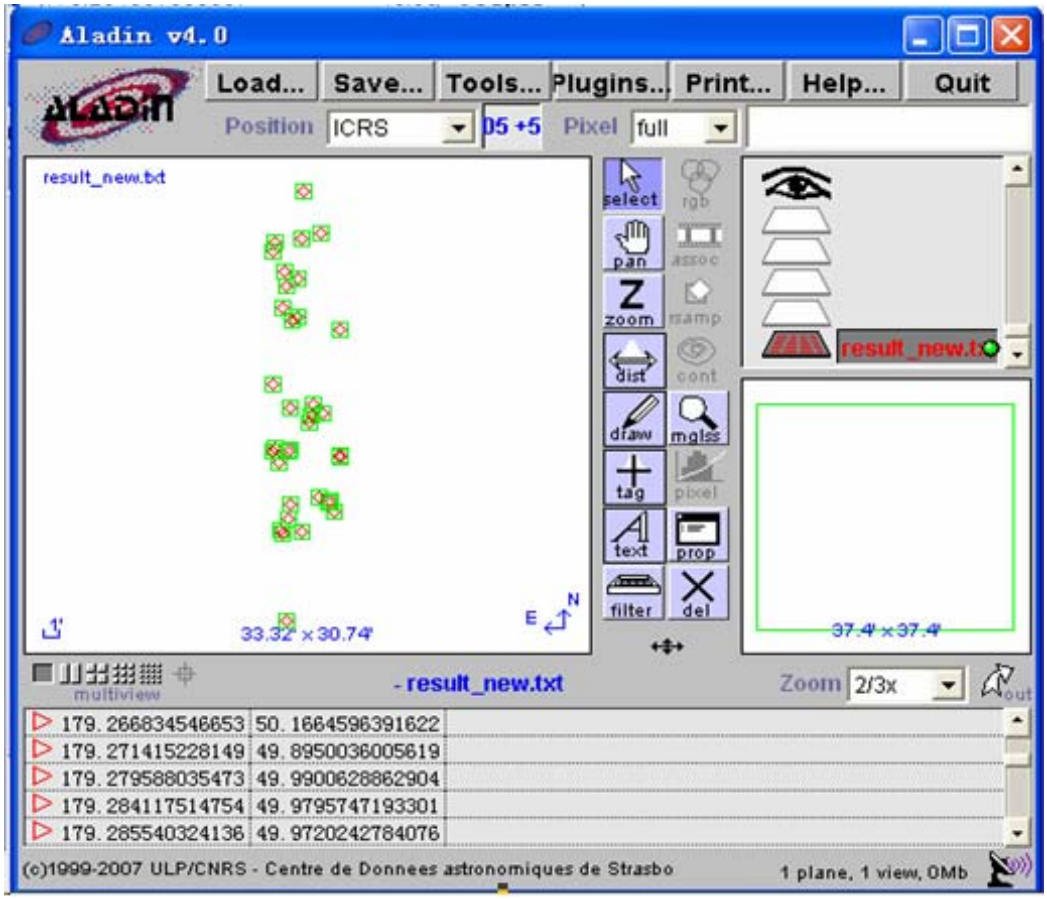

图 1.6 Aladin 应用程序客户端

设计出了 VO-DAS 的数据访问客户端软件。

如果只是开发现有模式的工具来满足天文学家复杂研究的需求，这样就会伴随工具数量增多，工具功能复用的耦合度下降，势必会影响人力、物力资源的高效利用，而且天文学家借助工具的工作效率也难以实质性地跨跃新台阶。因此，开发可复用性高和交互性好的天文服务工具来满足天文学家与日俱增的需求是虚拟天文台的任务。籍如此，我们所实现的平台将尽可能地扩大它的用户群，最大限度地发挥它的潜力和使用效率，使得它既能满足初级用户的访问，也能适应高级用户的查询；既能方便实现小数据量的访问，也能容易循环获取大数据的查询；既要节省开发设计成本，也要具有良好的互操作性，这些将成为 VO-DAS 网格应用设计的关键。

立足 VO-DAS 的接口服务，我们设计并实现了一套客户端系统，即 VO-DAS 的 GUI 客户端、命令行客户端和 Web 客户端。它们分别以不同的方式响应用户对 VO-DAS 的请求，完成对分布存放异构数据资源的访问。其中，GUI 是图形界面交互方式，它易学易用，适合数据访问频率不是很大的初级数据用户；命令行客户端以命令行来处理数据用户的查询，命

令行还可集成到用户自己的程序中，它能满足高级用户频繁的数据访问需求；Web 客户端以浏览器的方式响应用户的请求，这种形式的客户端尚在开发之中，本论文将不予以介绍。这样，数据用户通过客户端就可实现对 VO-DAS 接口的调用，从而访问分布存储的异构天文数据资源。这三种客户端分别以不同的形式将 VO-DAS 的所有功能外化给用户，并为用户提供友好的操作方式， 包括：

1) 客户端能够方便地获取系统中的元数据信息，包括数据资源、表元数据、表列元数据及其详细描述信息。
2) 提供用户可选择的同步查询和异步查询两种请求方式。
3) 能方便、直观地获取数据存取的任务信息。
4) 易于管理查询任务的相关信息。

另外，GUI 还支持与其它 VO 工具进行互操作的能力。即多种类型的天文服务能够在一个统一的环境下为了一项共同的研究而相互合作，需要它们之间具有良好的互操作性，也即来源不同的、功能不同的天文工具服务能够相互“理解”对方的数据和操作。

VO-DAS 实现的客户端系统为数据用户提供了一个分布式异构海量天文数据库的统一访问方法，它将 VO-DAS 对各国天文数据整合的资源，以标准、透明和易用的方式提供给国内外天文学家。GUI 同时还整合了其它各国虚拟天文台的 VO 应用，从而为进一步的数据可视化工作、数据挖掘工作提供了支持。另外，VO-DAS 系统不仅是中国虚拟天文台的主要数据访问服务环境，还将作为 LAMOST 以及未来国内其它天文观测项目数据发布平台的基础。这对推动中国虚拟天文台与各国 VO 工具的应用进行大规模和不同方式的数据访问和分析工作具有很重要的现实意义。

本文在后面的各个章节安排如下：第二章中我们将简单介绍本论文涉及的相关标准和技术。第三章将详细介绍 GUI 客户端的需求分析、设计与实现过程。第四章我们将介绍命令行客户端的开发过程。第五章我们将详细介绍 VO-DAS 系统的集成与部署，重点说明数据节点配置工具的设计与实现，简单说明系统的集成过程。第六章以科学应用范例来说明本论文工作的应用。最后，在第七章中做出总结和对未来的工作做出展望。

参 考 文 献：

[1] 崔辰州. 不断走向实用的虚拟天文台. http://www.china-vo.org/cn/events/cvo07/talks/ivo-CCZ.ppt.

[2] 崔辰州. 中国虚拟天文台系统设计. PhD thesis,中国科学院研究生院, 2003.

[3] Brunner R., Djorgovski S., Szalay A. Towards a National Virtual Observatory. Virtual Observatory of the Future. Michigan. 2001: p.343-372.

[4] IVOA, http://www.ivoa.net.

[5] ASTROGRID, http://www.astrogrid.org.

[6] SkyQuery, http://www.skyquery.net.

[7] OpenSkyQuery, http://www.openskyquery.net.

[8] CDS, http://cdsweb.u-strasbg.fr/.

[9] China-VO, http://www.china-vo.org.

[10] ZhaoY.-H., LAMOST project and its scientific goals. Publications of the Yunnan Observatory , December 1999:p1-7.

[11] Cui C.-Z. and Zhao Y.-H.. Architecture of Chinese Virtual Observatory. Astronomical Research and Technology, Publications of National Astronomical Observatories of China. 2004(Vol.1,No.2): p140-151 .

[12] 肖侬. 网格体系结构 OGSA. http://www.chinagrid.net/grid/paperppt/bigfile/paperppt/XiaoN/Arch2.ppt .

[13] Wang D. and Zhao Y.-H.. VO-IMPAT: Image Processing and Analysis Toolkit for the Virtual Observatory of China. Astronomical Research and Technology, Publications of National Astronomical Observatories of China. 2006 (Vol.3,No.3),p:295-303.

[14] Cui C.-Z., Sun H.-P., and Zhao Y.-H.. SkyMouse, An Integrated On-line Astronomical Information Access System. The Virtual Observatory in Action: New Science, New Technology, and Next Generation Facilities, 26th meeting of the IAU, Special Session 3, 17018, 21-22 August,2006 in Prague,Czech Republic ,SPS3,#55.

[15] Cui C.-Z., Li W., Yu C., Xu Z. & Zhao Y.-H.. Search and Location of FITS Data Files. Astronomical Research and Technology, Publications of National Astronomical Observatories of China, 2007, in press

[16] 刘超,田海俊,高丹,杨阳,路勇,崔辰州,赵永恒. 异地异构天文数据资源的统一访问.天文研究与技术, 2008, in press.

[17] Globus Toolkit, http://www.globus.org.

[18] OGSA-DAI,http://www.ogsadai.org.uk

[19] Web Service, http://www.w3.org/.

[20] 刘超. 基于虚拟天文台的数据挖掘技术及其在银河晕结构研究中的应用. PhD thesis,中国科学院研究生院,2008.

[21] WSRF, http://www.globus.org/wsrf.

[22] Open SkyQuery, http://www.openskyquery.net.

[23] ADQL, http://www.ivoa.net/Documents/WD/ADQL/ADQL-20050602.pdf.

[24] SkyNode, http://www.ivoa.net/Documents/WD/SNI/SkyNodeInterface-20050602.pdf.

[25] casjob, http://casjobs.sdss.org/CasJobs/.

[26] Bonnarel F.,Fernique P.,Bienayme O,et al. The AALADIN interactive sky atlas, A&AS, 2000(143):p33-40.

[27] Taylor J.,et al. Binding Applications Together with PLASTIC, Astronomical Data Analysis Software and Systems XVI ASP Conf.Ser., 2007:p511.

第二章 相关标准与关键技术

2.1 IVOA 相关标准

2.1.1 ADQL

ADQL 是针对天文数据访问特点而设计的数据库查询语言。它扩充 Region 实现锥形检索，能够进行多表联合的交叉证认，支持常用的数学函数等功能。天文数据访问门户或客户端程序发送 ADQL 查询语句到天文数据结点的查询接口上，由其内部的 ADQL 解析器来将该查询语句转换为关系数据库可识别的 SQL 标准语句。ADQL 具有两种表现形式：字符串形式（ADQL/s）和 XML 形式（ADQL/x）。前者基于 SQL92 并符合 ADQL 的语法，后者是遵循 XSD 的 XML 文本。两者可以在不丢失任何信息的情况下进行相互转换。如图 2.1 是 ADQL/s 查询语句的示例。

```
SELECT   a.objid, a.ra, a.dec
FROM   SDSSDR6:PhotoPrimary a
WHERE   Region('CIRCLE J2000 181.3 -0.76   6.5')
```

图 2.1 ADQL/s 语句表现形式

与此等价功能的是如图 2.2 所示的 ADQL/x 表现形式。它们表示查询 SDSSDR6 的 Photoprimary 星表中符合条件原点坐标 J2000、赤经 181.3 度、赤纬-0.76 度、半径 6.5 度的小锥形天区内的 objid、ra 和 dec 值。

2.1.2 VOTable

VOTable[1]是 IVOA 制定的基于 XML 的数据格式编码标准，用来解决大数据量的传输和网格计算，提高数据存储的灵活性和有效性的表格描述形式。它的文档部分分成两个部分，一部分是元数据的定义，另一部分是数据。在 VOTable 里详尽地描述天文星表的数据内容以及元数据，其中对于星表元数据描述的内容包括表格自身的描述以及表格中每列

```
<?xml version="1.0" encoding="utf-8"?>
<Select xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ivoa.net/xml/ADQL/v0.9">
  <SelectionList>
    <Item xsi:type="columnReferenceType" Table="a" Name="objid" />
    <Item xsi:type="columnReferenceType" Table="a" Name="ra" />
  </SelectionList>
  <From>
    <Table xsi:type="archiveTableType" Archive="SDSSDR6
Name="Photoprimary" Alias="a" />
  </From>
  <Where>
    <Condition xsi:type="regionSearchType">
      <Region xmlns:q1="http://www.ivoa.net/xml/STC/STCregion/v1.10"
xsi:type="q1:circleType" unit="deg">
        <q1:Center>181.3 -0.76</q1:Center>
        <q1:Radius>6.5</q1:Radius>
      </Region>
    </Condition>
  </Where>
</Select>
```

图 2.2 ADQL/x 表现形式

```
<?xml version="1.0"?>
<VOTABLE version="1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="http://www.ivoa.net/xml/VOTable/VOTable/v1.1">
  <COOSYS ID="J2000" equinox="J2000." epoch="J2000." system="eq_FK5"/>
  <RESOURCE name="myFavouriteGalaxies">
    <TABLE name="results">
      <DESCRIPTION>Velocities and Distance estimations</DESCRIPTION>
      <PARAM name="Telescope" datatype="float" ucd="phys.size;instr.tel"
             unit="m" value="3.6"/>
      <FIELD name="RA"   ID="col1" ucd="pos.eq.ra;meta.main" ref="J2000"
             datatype="float" width="6" precision="2" unit="deg"/>
      <FIELD name="Dec"  ID="col2" "pos.eq.dec;meta.main" ref="J2000"
             datatype="float" width="6" precision="2" unit="deg"/>
      <FIELD name="Name" ID="col3" ucd="meta.id;meta.main"
             datatype="char" arraysize="8*"/>
      <FIELD name="RVel" ID="col4" ucd="src.veloc.hc" datatype="int"
             width="5" unit="km/s"/>
      <FIELD name="e_RVel" ID="col5" ucd="stat.error;src.veloc.hc"
             datatype="int" width="3" unit="km/s"/>
      <FIELD name="R" ID="col6" ucd="phys.distance" datatype="float"
             width="4" precision="1" unit="Mpc">
        <DESCRIPTION>Distance of Galaxy, assuming H=75km/s/Mpc</DESCRIPTION>
      </FIELD>
      <DATA>
        <TABLEDATA>
        <TR>
          <TD>010.68</TD><TD>+41.27</TD><TD>N  224</TD><TD>-297</TD><TD>5</TD><TD>0.7</TD>
        </TR>
        <TR>
          <TD>287.43</TD><TD>-63.85</TD><TD>N 6744</TD><TD>839</TD><TD>6</TD><TD>10.4</TD>
        </TR>
        <TR>
          <TD>023.48</TD><TD>+30.66</TD><TD>N  598</TD><TD>-182</TD><TD>3</TD><TD>0.7</TD>
        </TR>
        </TABLEDATA>
      </DATA>
    </TABLE>
  </RESOURCE>
</VOTABLE>
```

图 2.3. VOTable 文档实例

元素的特性诸如字段名、单位、数据类型、精度、宽度和统一内容描述等等。如图 2.3 所示描述了一个简单的 VOTable 的文档。

2.1.3 PLASTIC 协议

PLASTIC(Platform for Astronomy Tool InterConnection)是一种用于天文桌面应用工具之间的通讯协议。它主要用于解决 VO 工具之间的互操作问题，例如使对方完成装载 VOTable、高亮度地显示数据行的内容子集和加载特定天区的图像等功能。PLASTIC 包含两个重要的概念：PLASTIC HUB 和 PLASTIC MESSAGE。HUB 用于接收双方或多方应用的注册并负责消息路由和转发功能。MESSAGE 是与平台无关传输消息的载体，它包括两组消息：核心消息和天文应用消息。应用程序可以根据需要对消息系统进行扩展。如图 2.4 是 PLASTIC 的一组核心消息协议。

Message ID	Args	Returns	Description
ivo://votech.org/test/echo	text: String	response: String	Echos text (used for troubleshooting and pinging an application).
ivo://votech.org/info/getIVORN	void	id: String	Returns the tool's registerable ID e.g. ivo://org.astrogrid/ar
ivo://votech.org/info/getName	void	name: String	Get a human-readable name e.g. Topcat
ivo://votech.org/info/getDescription	void	name: String	Get a human-readable description of the application.
ivo://votech.org/info/getVersion	void	version: String	Get the version of PLASTIC supported.
ivo://votech.org/info/getIconURL	void	url: String	Get a URL to an icon for the application.
ivo://votech.org/hub/event/HubStopping	void	void	Sent by the hub before it shuts down
ivo://votech.org/hub/event/ApplicationRegistered	id: String	void	Sent by the Hub when an application registers
ivo://votech.org/hub/event/ApplicationUnregistered	id: String	void	Sent by the Hub when an application registers

图 2.4. PLASTIC 核心消息

2.2 网格服务

网格[2]是利用互联网把地理上广泛分布的各种资源（包括计算资源、存储资源、带宽资源、软件资源、数据资源、信息资源、知识资源等）连成一个逻辑整体，就像一台超级

计算机一样，为用户提供一体化信息和应用服务（计算、存储、访问等），虚拟组织最终实现在这个虚拟环境下进行资源共享和协同工作，彻底消除资源“孤岛”，最充分的实现信息共享。网格技术的特点：

1. 资源共享，消除资源孤岛：网格能够提供资源共享，它能消除信息孤岛、实现应用程序的互连互通。网格与计算机网络不同，计算机网络实现的是一种硬件的连通，而网格能实现应用层面的连通。
2. 协同工作：网格第二个特点是协同工作，很多网格结点可以共同处理一个项目
3. 通用开放标准，非集中控制，非平凡服务质量：这是 Ian Foster 最近提出的网格检验标准。网格是基于国际的开放技术标准，这跟以前很多行业、部门或者公司推出的软件产品不一样。
4. 动态功能，高度可扩展性：网格可以提供动态的服务，能够适应变化。同时网格并非限制性的，它实现了高度的可扩展性。

网格之所以能有以上所说的种种优势特征，是由网格的体系结构赋予它的。网格体系结构是关于如何建造网格的技术，包括对网格基本组成部分和各部分功能的定义和描述，网格各部分相互关系与集成方法的规定，网格有效运行机制的刻画。图 2.5 为网格的沙漏体系结构[3]。

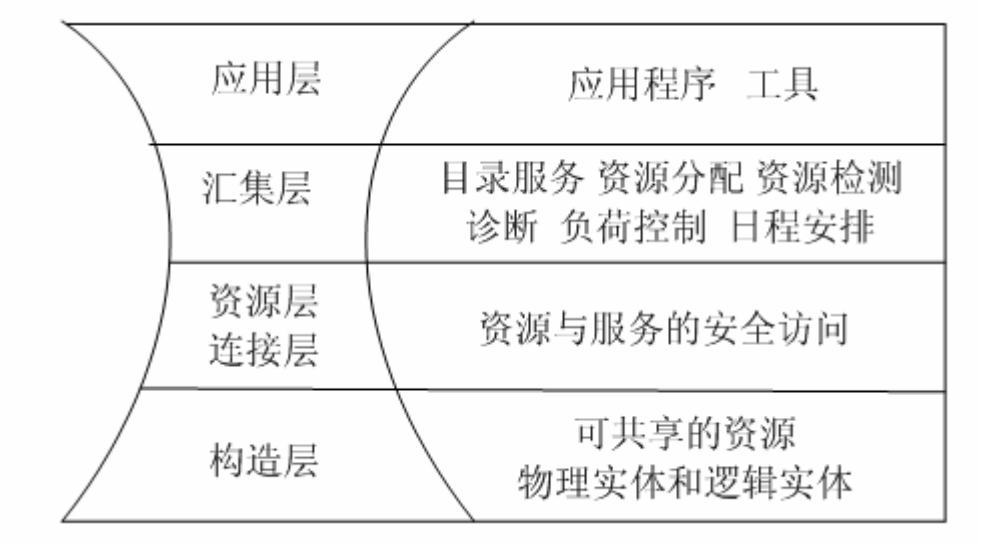

图 2.5 网格沙漏体系结构

该结构主要包括以下五个层次：

构造层(Fabric)：控制局部的资源。由物理或逻辑实体组成，目的是为上层提供共享的

资源。常用的物理资源包括计算资源、存储系统、目录、网络资源等；逻辑资源包括分布式文件系统、分布计算池、计算机群等。构造层组件的功能受高层需求影响，基本功能包括资源查询和资源管理的 QoS 保证。

连接层(Connectivity)：支持便利安全的通信。该层定义了网格中安全通信与认证授权控制的核心协议。资源间的数据交换和授权认证、安全控制都在这一层控制实现。该层组件提供单点登录、代理委托、同本地安全策略的整合和基于用户的信任策略等功能。

资源层(Resource)：共享单一资源。该层建立在连接层的通信和认证协议之上，满足安全会话、资源初始化、资源运行状况监测、资源使用状况统计等需求，通过调用构造层函数来访问和控制局部资源。

汇集层(Collective)：协调各种资源。该层将资源层提交的受控资源汇集在一起，供虚拟组织的应用程序共享和调用。该层组件可以实现各种共享行为，包括目录服务、资源协同、资源监测诊断、数据复制、负荷控制、账户管理等功能。

应用层(Application)：为网格上用户的应用程序层。应用层是在虚拟组织环境中存在的。应用程序通过各层的应用程序编程接口（API）调用相应的服务，再通过服务调动网格上的资源来完成任务。为便于网格应用程序的开发，需要构建支持网格计算的大型函数库。

OGSA（Open Grid Services Architecture）被称为是下一代的网格体系结构，它是在“五层沙漏结构”的基础上，结合最新的 Web Service 技术提出来的。OGSA 最突出的思想就是以“服务”为中心。在 OGSA 框架中，将一切都抽象为服务，包括计算机、程序、数据、仪器设备等。这种观念，有利于通过统一的标 准接口来管理和使用网格。OGSA 包括两大关键技术即网格技术和 Web Service 技术。Web Service 提供了一种基于服务的框架结构，但是，Web Service 面对的一般都是永久服务，而在网格应用环境中，大量的是临时性的短暂服务，比如一个计算任务的执行等。考虑到网格环境的具体特点，OGSA 在原来 Web Service 服务概念的基础上，提出了“网格服务（Grid Service）”的概念，用于解决服务发现、动态服务创建、服务生命周期管理等与临时服务有关的问题。基于网格服务的概念，OGSA 将整个网格看作是“网格服务”的集合，但是这个集合不是一成不变的，是可以扩展的，这反映了网格的动态特性。网格服务通过定义接口来完成不同的功能，服务数据是关于网格服务实例的信息，因此网格服务可以简单地表示为“网格服务＝接口/行为＋服务数据”。

Globus Alliance[4]是网格中间件的提供者和网格应用的推动者，在网格中间件早期版本的 Globus Toolkit 3.0 中采用了 OGSA 架构，2005 年推出的 Globus Toolkit 4.中又全面支持 WSRF 框架。WSRF，或者称为 Web 服务资源框架，是一个在 Web 环境中设计处理有状态资源的一个规范，它提出了提供持久数据的方式。作为一个框架，它有一系列的处理细节模块组成，分别是：

- WS 资源:定义了 Web services 如何能够被用来表示多资源实例。
- WS-资源属性(WSRF-RP):指定了同 WS 资源中定义的属性进行交互的方式。
- WS-资源生命周期(WSRF-RL):定义了如何来管理和销毁 WS-资源的生命周期。
- WS-服务组(WSRF-SG):指定了如何来聚合 WS-资源。
- WS-基本错误(WSRF-BF):定义了被 WSRF 服务抛出的 SOAP 格式的错误。

2.3 OGSA-DAI

开放网格服务架构 - 数据访问与集成（Open Grid Services Architecture - Data Access and Integration，OGSA-DAI）是一种中间件，其设计目标是提供一种简便的方法，在网格环境中实现数据的访问和集成。它具有如下功能特点：支持不同的数据资源，包括关系型和 XML 数据库，通过 Web 服务在网格上发布出来；这些类型的数据资源能被查询和更改；使用 XSLT 实现数据格式的转换；支持直接将查询结果数据返回给客户端，也可以通过 URL、FTP、GridFTP 等传输协议将数据传输到指定的位置。它的体系结构如图 2.6 所示。data layer(数据层)：数据层是通过 OGSA-DAI 发布的数据资源，包含：关系数据库(MySQL、Oracle、DB2 等)、XML 数据库和文件。

business logic layer(业务逻辑层)：该层封装了 OGSA-DAI 的核心功能，它执行客户端的请求并返回执行结果，管理数据传输和发布以及数据源的链接、管理、交互等。

presentation layer(表示层)：该层通过 Web 服务接口封装 OGSA-DAI 向网格展示所有的服务。

client layer(客户层) ：OGSA-DAI 支持与 WSRF、WS-I 兼容的客户端的访问。通过客户端工具可以实现对底层数据源的检索、更新等操作。

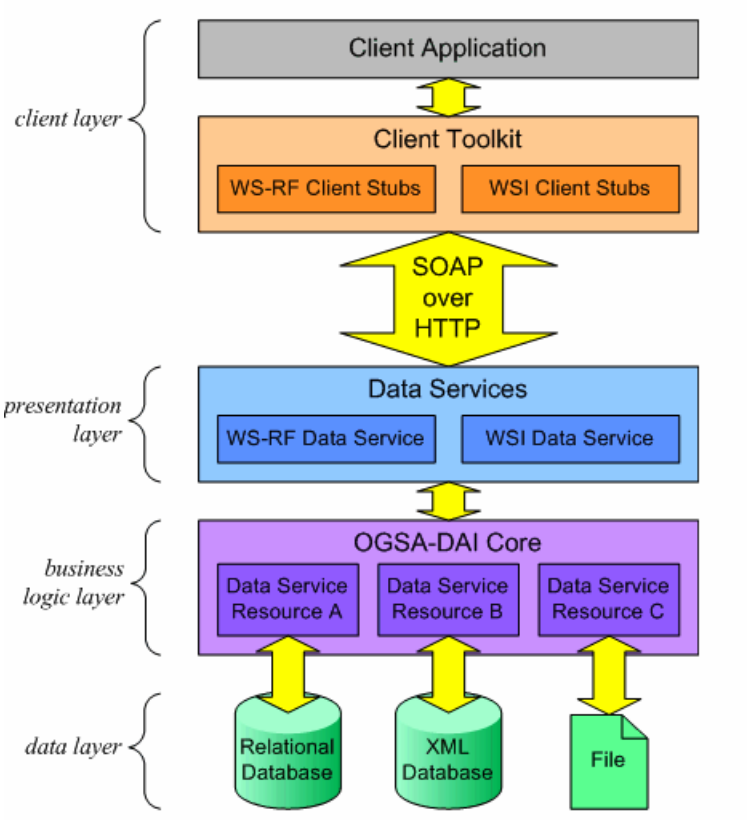

图 2.6 OGSA-DAI 的体系结构

参 考 文 献

[1] VOTable, http://www.ivoa.net/Documents/REC/VOTable/VOTable-20040811.html.

[2] Miguel L.BoteLorenzo, Yannis A.Dimitriadis. Grid Characteristics and Uses: A Grid Definition. http://www.chinagrid.net/dvnews/upload/2005_03/05031923425425.pdf.

[3] 樊宁. 网格体系结构概述. http://developer.51cto.com/art/200611/35461.htm.

[4] Globus Alliance, http://www.globus.org.

第三章 GUI 客户端系统的设计与实现

本章将描述 GUI 客户端[1]的设计与实现。VO-DAS 作为异地异构的数据访问服务被实现以后，需要有与之相应的客户端来与用户进行交互，从而达到访问服务的目的，进而获取用户查询的结果。我们开发的 VO-DAS 的 GUI 客户端能够满足用户的这一需求，并且为初学者提供了人性化的人机交互界面。本章将对 GUI 客户端按照软件设计过程的思路详细介绍它的需求分析、系统设计以及软件实现。

3.1 GUI 客户端需求分析

VO-DAS 对外提供简单、直观的访问操作接口，这些接口实现了对异地异构海量天文数据资源的访问。但这些接口仅限于程序操作，最终用户很难来直接访问它们。因此就需要基于这些 Web 服务接口，实现 VO-DAS 的网格应用产品，使用户能够通过这个网格应用产品即客户端直接地访问 VO-DAS，从而获取自己想要的数据。同时，我们还必须考虑到设计什么形式的客户端，才能够提高 VO-DAS 的应用效果，这也将长远地影响到作为 VO-DAS 重要基础的大型项目 LAMOST 观测数据的使用效率问题。因此，我们设计了 GUI 客户端，它的实现有如下几个目的：

1) 所有的操作将以可视化的方式来与服务器进行交互。大部分天文学家对 VO-DAS 及其相关的技术不太熟悉，可视化的方式使用户不需专业培训就能够使用客户端来对 VO-DAS 进行操作。
2) 适合低频率的数据访问。用户须手动地对服务器进行每一次数据查询任务的提交工作，频繁地作业提交将比较耗力。
3) 适合小数据量的查询。对于同步方式的查询，用户等待返回的结果数据量较大时将会受到本地机器内存的限制。
4) 实现与其它 VO 工具的交互协同操作，使它们在一个环境中协同实现比较复杂的天文研究任务。

3.2 GUI 客户端的设计

3.2.1 系统的总体结构

GUI 客户端以可视化的形式响应用户对服务器请求的每一个操作。它按功能模块可分为：界面模块、连接模块、元数据处理、监控模块、表示模块、异步查询、同步查询、数据处理和退出处理模块，如图 3.1 所示。

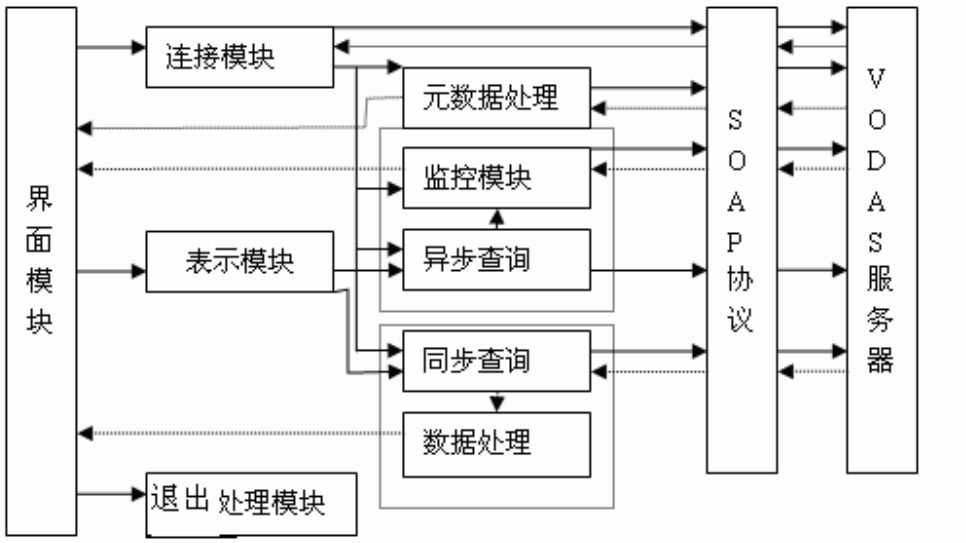

图 3.1 VO-DAS GUI 客户端设计

图 3.1 中实线箭头表示客户端发送请求的流程，虚线箭头表示客户端接收服务器消息的流程。

界面模块负责显示从服务器端获取数据信息包括元数据和监控信息，处理用户输入的查询请求并将查询结果呈现给用户。

连接模块接收用户在界面输入的服务器地址参数并转换成 WSRF 的 URI 地址信息，连接到指定的 DAS 服务器。

元数据处理模块负责获取所连接服务器上的元数据信息，将结果信息以树形结构形式展现给用户，作为用户编辑 ADQL 查询语句的参考信息。

表示模块、异步查询和同步查询分别负责处理用户输入的请求数据，然后根据查询请求方式作同步查询或异步查询。

监控模块负责循环监控异步提交查询任务的状态信息，并根据此信息对任务做相应的

处理。

数据处理模块负责处理用户同步查询返回的结果数据。这些数据可以由用户保存在本地机器上，也可通过客户端集成的 PLASTIC 将它们发送到其它应用程序作进一步地分析处理或研究。

退出处理模块负责应用程序关闭时将服务器上正在运行的任务或已运行完成的任务信息保存在本地机器上，便于下次启动进行初始化处理。

SOAP 协议是客户端与服务器交互的通讯协议。客户端向服务器发起的信息以及服务器返回给客户端的信息均由 SOAP 协议来进行通讯。

3.2.2 系统工作的一般流程

GUI 客户端既可以采用同步方式又可以采用异步方式进行查询。本节根据图 3.1 的框架流程图，使用异步方式与同步方式，说明客户端工作的一般流程。

异步方式：

1. 客户端向服务器发出建立连接的请求，服务器端做出连接回应，连接成功。
2. 连接初始化：
   i. 调用服务器的 RMI 接口 GetAllResource、GetMetaTables 和 GetMetaColumn，从服务器端获取 VO-DAS 系统中的元数据，在客户端进行相应地处理后以树形结构的方式呈现给用户；
   ii. 读取本地机器上次保存的任务信息，可视化其历史信息和运行任务信息，后者添加至监控队列，并启动客户端监控器对其进行监控。
3. 客户端请求异步查询。用户参考元数据信息编辑查询语句、命名查询任务、指定数据保存的格式和地址，根据用户输入的信息分析查询模式，如果是异步查询，则调用 DQI 的 asynQuery 异步接口，提交查询请求。
4. 客户端监控查询任务。查询任务提交后被追加至监控队列，此任务的其它相关信息可视化给客户端，包括任务 ID、任务名、调用服务器端 MI 的 GetStartTime、GetSubmitTime、GetEndTime 和 getStatus 接口的值，并且根据 getStatus 接口的值更新客户端的状态信息来实施监控。同时也可请求调用 MI 的 destorySession 接口销毁正在执行的任务。循环 3。
5. 对于查询状态完成的任务，监控器将任务追加至历史队列。用户可以从结果数据保

存的地址处下载该数据。

同步方式：

前两步与异步方式相同。

3．客户端请求同步查询。用户参考元数据信息编辑查询语句、指定数据保存的格式，根据用户输入的信息分析查询模式，如果是同步查询，调用 DQI 的 synQuery 接口，提交查询请求。

4．用户等待查询结果数据返回给客户端。

5．结果数据处理。结果数据返回后，用户可以根据自己的需要对其进行处理：或保存在本地机器上，或通过客户端交互式操作的方式将数据直接发送到其它 VO 工具进行数据挖掘或分析处理。

从以上工作流程中可以知道，GUI 客户端的同步查询和异步查询的区别主要体现在：

1）查询请求方式：异步方式允许用户连续提交查询请求，同步方式则需等待结果数据返回后才能提交下一个同步请求。

2）结果数据处理：异步查询的结果数据保存在用户指定的地址处，用户可以通过 FTP 或 GridFTP 下载其数据；同步查询的结果数据直接返回给客户端，由用户选择是否做进一步地处理。

这两种查询的方式，都用 ADQL 查询语言来描述查询的任务。数据保存格式分别可以是 ASCII、VOTable、FITS、GZIP。

3.2.3 查询结果数据处理

由上节中可知，两种查询请求导致两种不同的结果数据处理方式。异步查询的数据处理方式相对比较简单，它的结果直接保存在 FTP 或 GridFTP。下面着重介绍同步方式的数据处理方式。

同步查询的结果数据以用户指定的格式直接返回给用户，此时用户可以选择保存在本地磁盘上以便下次使用时再次装载，也可以将结果数据直接发送到工具如 TOPCAT 或 ALADIN，这个功能是通过集成 IVOA 的消息通讯机制 PLASTIC 协议来实现的，这大大提高了 VO-DAS GUI 客户端与其它 VO 工具的互操作性。如图 3.2 所示是 VO-DAS 的 GUI 客户端与 TOPCAT 工具交互的过程。

VO-DAS 与 TOPCAT 的互操作过程通过 PLASTIC 协议中的 PLASTIC HUB 和

PLASTIC MESSAGE 来实现。HUB 首先接受双方应用程序的注册，应用的一方以 MESSAGE 为消息载体向 HUB 发出请求，此时 HUB 解析 MESSAGE 中的目的地，并根据

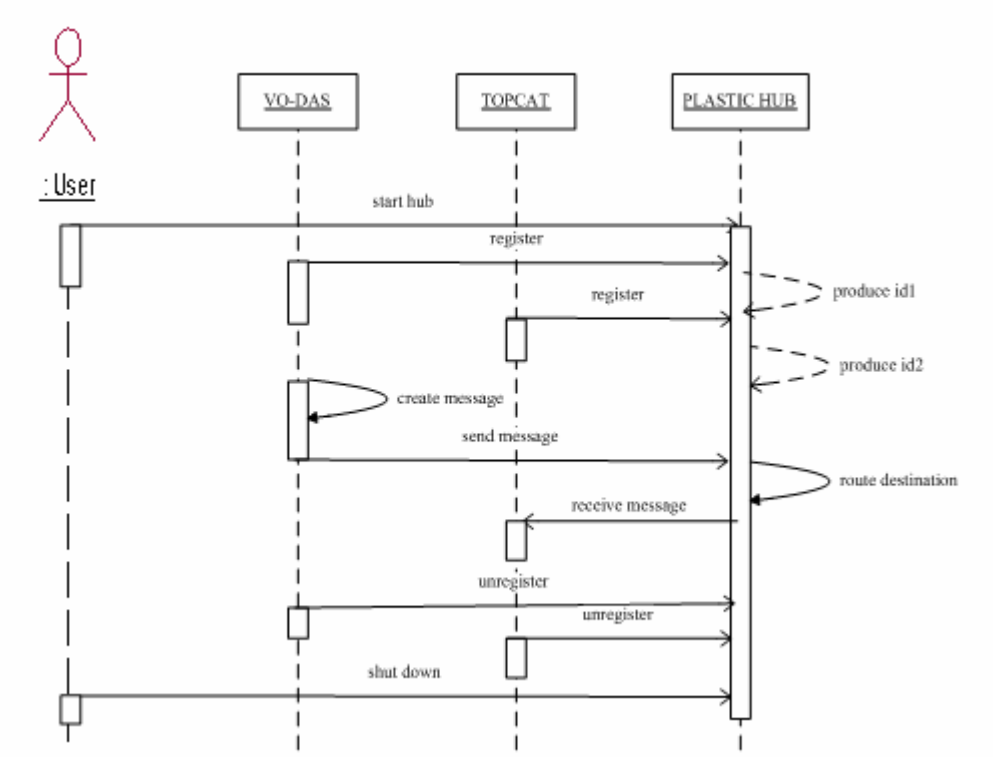

图 3.2. VO-DAS 和 TOPCAT 基于 PLASTIC 进行交互式操作示意图

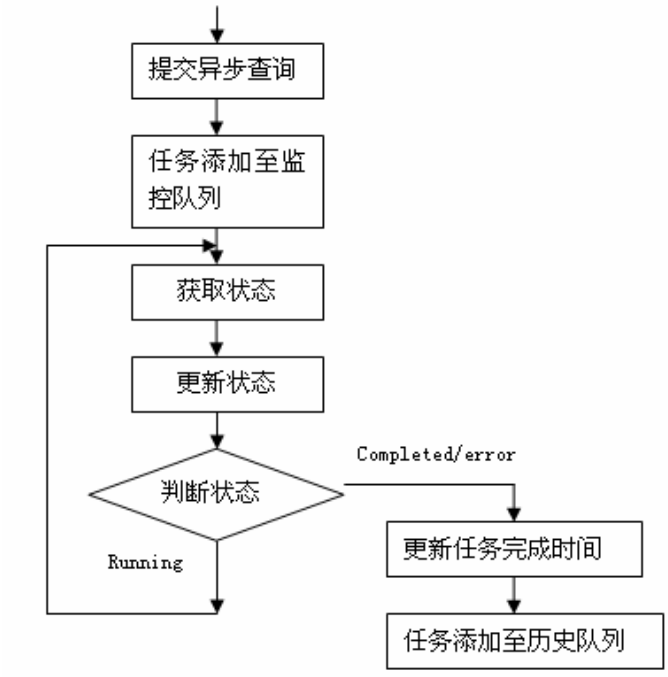

图 3.3. 监控器的流程控制图

注册在其中的 ID 号进行路由，将请求的信息按照请求要求的方式传输给另一方的应用。交互过程完毕，对于外置的 HUB，由用户负责关闭 HUB，对于内置 HUB，则由应用程序关闭时一起关闭其交互功能。

3.2.4 监控模块

异步查询由于具有连续提交任务的特点，在服务器端同一时间内会有多个任务同时运行或等待。对于一个友好界面的客户端来说，需要一种机制来监控服务器上执行的任务并呈现给用户，以便用户根据任务执行的情况做出相应的操作。监控模块设计目的便是如此，它的流程控制示意图如图 3.3 所示。它循环从服务器上获取每个 Job 的状态信息，判断这个信息的内容，继而做出不同处理方式。具体实现过程中， 由主程序在开始运行时开辟一个新的线程，由这个线程循环监控服务器的任务状态。

3.2.5 获取元数据

系统连接成功后，客户端就从 VO-DAS 获取数据资源元数据，这个过程描述如下：

1. 客户端向 VO-DAS 调用 GetAllResource 接口询问数据资源的列表。
2. VO-DAS 根据自己的记录返回给客户端一个数据资源列表的 XML 字符串。
3. 客户程序解析返回的字符串，提取出列表资源的名字和描述信息。
4. 客户端向 VO-DAS 调用 GetMetaTable 接口询问某个数据资源名字的表信息。
5. VO-DAS 返回指定数据源包含所有表元数据列表的 XML 字符串
6. 客户程序解析返回的字符串，提取表元数据的名字和描述信息
7. 客户端向 VO-DAS 调用 GetMetaColumn 接口询问某个表的详细信息
8. VO-DAS 返回指定表的各个列的名称、类型、单位等描述信息的 XML 字符串
9. 客户程序解析返回的字符串，提取出列表元数据的名字和描述信息。

通过层层递进的方式，获取到了元数据名字和与之相关的描述两部分信息。程序将把元数据名字以纵深的树形结构显示在用户界面，其它的描述信息则以文本的形式与名字绑定，显示在一个文本框中。用户在编辑查询语句的时可参考这些元数据信息了解哪些数据可用以及这些数据所代表的含义。

3.3 GUI 客户端的实现

系统的开发平台采用 Eclipse 集成开发环境和面向对象的程序设计语言 JAVA。经过对 GUI 客户端进行面向对象分析和面向对象设计后，我们实现了以下几个主要的类：

Class main_frame：应用程序的入口和出口类。 应用程序启动时，首先执行这个类。此类实现了一个称作doShutDownWork（）方法，它为Java程序添加退出事件的响应，即在此方法中插入Runtime.getRuntime().addShutdownHook(new Thread(){…})，参数中的线程实现将监控队列中的Job任务以及历史信息以XML形式写入到本地磁盘。因此，当应用程序退出时，这个方法就执行磁盘写操作。

Class interfaceDesign：实现应用程序的主界面类。它主要包含的方法如表3.1所示：

表 3.1 interfaceDesign 类的实现

方法明称	说明
JMenuBar createMenuBar()	创建菜单栏
JToolBar createToolBar()	创建工具栏
JTree createTree(String[] a)	创建显示元数据结点的树
JTabbedPane createShowTabbedPane()	创建显示元数据描述信息的标签
JTabbedPane createQueryTabbedPane()	创建ADQL编辑器的标签
JDialog connectDialog(Main_frame frame)	创建连接服务器窗口

Class synResultView：实现显示同步查询的操作界面。

Class Job：实现与 Job 任务相关的属性描述或方法调用描述，如表 3.2 所示。

Class JoblistManager：实现控制器中显示 joblist 或 historyList 可视化的 tooltip 信息，加强用户获取信息的能力。

Class jobMonitor：任务监控线程。在其类的 run 方法中通过循环获取 joblist 中每个任务在服务器上的运行状态信息来实现监控的过程。

synThread：同步查询线程。同步查询的结果数据需要一直等待，为避免主线程一直忙状态，程序的实现时开辟这个子线程来实现同步查询，并接收查询结果。

Class interopateWithPlastic：实现同步结果数据与其它VO工具的互操作。类中实现的主要代码行如下所示：

```
hub = uk.ac.starlink.plastic.PlasticUtils.getLocalHub();  //获取外部本地Hub
```

```
URI id = hub.registerNoCallBack("VO-DAS");  //注册“VO-DAS”，返回ID号
URI message = new URI("ivo://votech.org/fits/image/loadFromURL"); //创建消息
List l = new ArrayList();  //创建数组列表
l.add(Thread.currentThread().getContextClassLoader().getResource("")+ "result.vot");
//向列表中添加即将发送的数据文件
hub.request(id, message, l);  //请求HUB发送数据
hub.unregister(id);  //消息发送完毕，反注册
```

上述代码描述了 VO-DAS 通过 PLASTIC 向其它 VO 工具发送消息的过程。

表 3.2 Job 类的实现

属性名或方法名称	说明
String jobID	job 的 ID 号
String taskNme	任务名
String queryString	查询语句
String Status	状态信息
Date submitTime	任务提交时间
Date startTime	开始时间
Date finishTime	结束时间
String DASURL	DAS 服务器地址
String DataURL	异步操作结果数据保存地址
Int Format	数据保存或返回的格式
String eprDirectory	应用程序关闭时任务保存地址
EndpointReferenceType EPR	Job 的 session
int queryType	查询提交类型
void AsynchRun()	实现调用同步查询方法
String synRun()	实现调用异步查询方法
Void finish(DASClient dasClient)	任务完成的处理方法
getJobState(DASClient dasClient)	调用获取状态方法

经由系统的分析和设计，所实现的 GUI 客户端的主界面如图 3.4 所示：

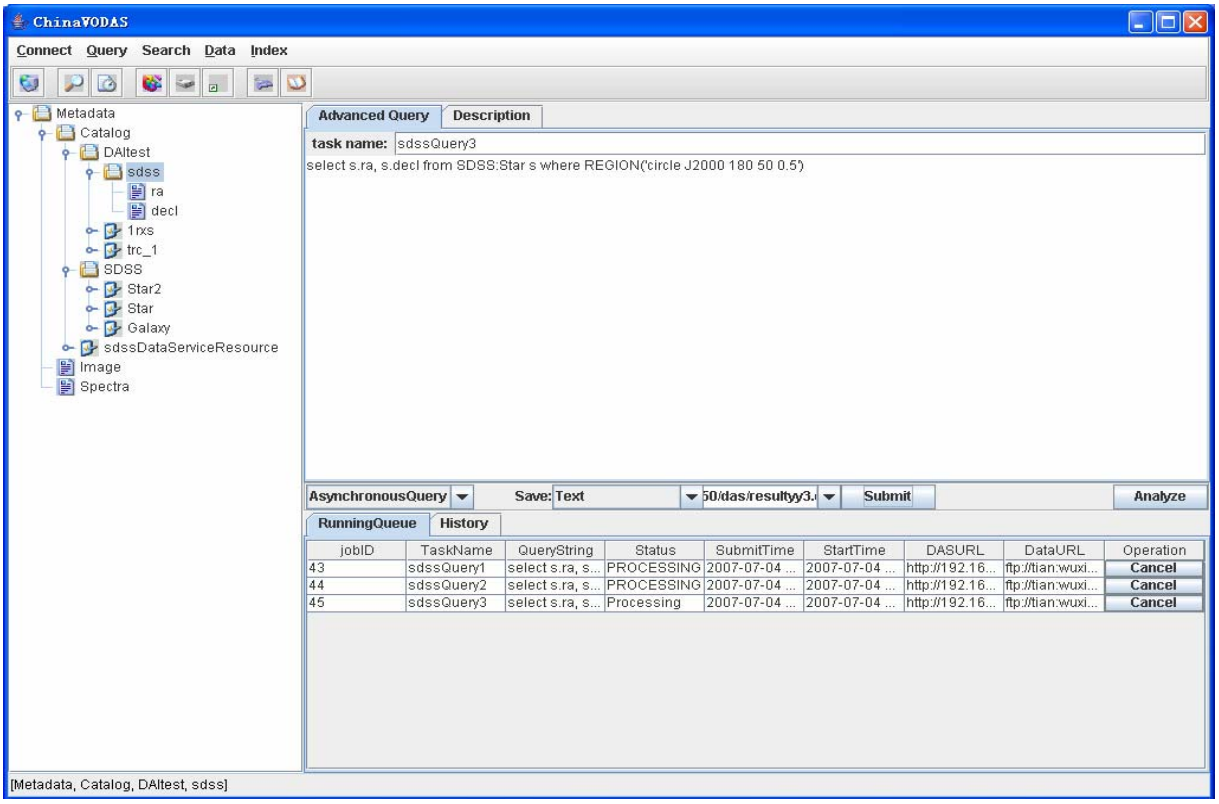

图 3.4.VO-DAS GUI 客户端

图 3.4 中左边部分是连接服务器后从中获取的元数据名称，它以树形结构显示。图中显示了星表类资源 DAItest，资源所包含的表格 sdss、1rxs 和 trc_1，每个表格下的所有列（如 sdss 中的 ra 和 decl）。这种层次关系的显示方式方便用户参考它来编辑 ADQL 语句。当用户点击左边元数据树形结点时，右上部分的“Description”标签区域中会详细显示该元数据的详细信息，包括它所表示的含义、制作人等。另外，右上部分的“Advanced Query”标签是查询语句编辑器，提交查询任务之前，需在此编辑 ADQL 语句。中间部分的下拉框及其按钮分别表示提交查询的方式（AsynchronousQuery/synchronousQuery）、数据保存的格式（Text、VOTable、ASCII、FITS 和 GZIP）、数据保存地址和任务提交按钮。右下部分的两个标签分别显示 JOB 在不同状态下的信息。“RunningQueue”记录处于服务器端正在运行或等的 JOB 信息；“History”记录在服务器端执行完毕或发生错误的 JOB 信息。

3.4 本章小结

本部分所实现的 GUI 客户端是 VO-DAS 客户端的一种形式。它主要遵循初级用户简单易用的原则，用面向对象的分析和设计方法，实现了基于 VO-DAS WEB 服务接口的网格应用产品。它的性能和可用性设计的好坏将直接关系到 VO-DAS 的使用效率，甚至是 China-VO 的用户群。本章介绍了 GUI 客户端的设计与实现过程。其中客户端的设计包括系统设计、主要功能模块设计和用户界面设计等等。

在系统的实现过程中，采用了 Eclipse 集成开发环境。它是开源软件工具，为实现修改的自动化提供良好的支持。另外，它还具有一种相当强大的代码格式化功能，这些特性大大提高了开发人员的工作效率。用 JAVA 语言作为主要编程语言，它具有体系结构中立的优点，在不同的软硬件平台上移植非常方便。同时， JAVA 支持多线程，在多处理器上运行时负载平衡尤为突出。另外，JAVA 是可扩展的，可与其它语言编写的现有程序库连接，满足了系统进一步扩展的需求。

GUI 的客户端可以满足初级用户非常方便地访问 VO-DAS 的服务，从而实现异步查询、同步查询、交叉证认和与其它 VO 工具的互操作。

参 考 文 献：

[1] 杨阳, 刘超, 田海俊, 崔辰州, 赵永恒. VO 数据访问服务客户端系统的设计与实现. 天文研究与技术, in press, 2008

第四章 命令行客户端的设计与实现

我们将在本章详细介绍针对 VO-DAS 开发的另外一种客户端：命令行客户端。与 GUI 客户端类似，它可实现对 VO-DAS 的数据访问请求，但是它以命令行的方式响应用户的请求。它能够满足高级用户对数据的频繁访问请求。

4.1 命令行客户端的需求分析

由 3.1 我们可知 VO-DAS 客户端设计的必要性，GUI 客户端的实现只是针对初级用户对 VO-DAS 小数据量的访问。由 1.2 节所知各地存储的天文资源具有海量数据访问的特点，特别循环获取某些海量资源的时候，GUI 的客户端就会受到一些操作限制。如何最大限度地发挥 VO-DAS 的潜力和提高其使用效率，是 VO-DAS 客户端设计的目标之一。经过长期的调研和分析， VO-DAS 的命令行客户端可以很好地实现这一目标。由它来提供一组访问 VO-DAS 的基本操作命令，用户不需要图形终端就可以通过这些命令完成数据的查询。因此，它为用户访问 VO-DAS 提供了一种灵活便捷的方式。同时，命令行方式还可以集成到用户自己的程序里面，无论是 Python 还是 FORTRON，都能轻易地通过这组命令完成频繁的数据访问。这类客户端适合访问数据比较频繁或者需要使用自己编写的程序完成数据访问任务的用户。

4.2 命令行客户端的设计

4.2.1 系统的总体结构

命令行客户端以命令行的方式给用户提供访问异地异构数据的查询。它的设计分为两大部分，一部分是实现 Linux 或 Windows 环境下的 shell 或 bat 命令，另一部分是实现后台调用服务器接口的一组 Java 程序。命令行客户端的总体结构按照功能模块划分，如图 4.1。

界面负责 Linux 或 Windows 下的 Shell 或 Bat 命令的编辑、提交请求并将结果返回给用户。Shell 和 Bat 命令模块分析用户从界面输入的命令请求并调用后台相应的接口。连接模块用于通过 SOAP 协议向服务器发起连接请求。同步查询和数据处理分别负责以同步的

方式提交查询，将查询返回的结果以用户指定的格式呈现给用户。异步查询、获取状态、销毁任务和获取 URL 模块分别负责向服务器提交异步查询,发起获取查询状态的请求并返回状态值，撤销正在服务器端运行的任务，获取查询结果数据保存的地址 URL。元数据处理模块获取并处理系统中的可用元数据，以 XML 文本格式显示给用户。帮助模块负责显示用户如何使用客户端的相关信息。上述模块中除帮助模块外，其余的功能模块均要求用户从界面部分发起命令、连接服务器后才能完成相关的功能。

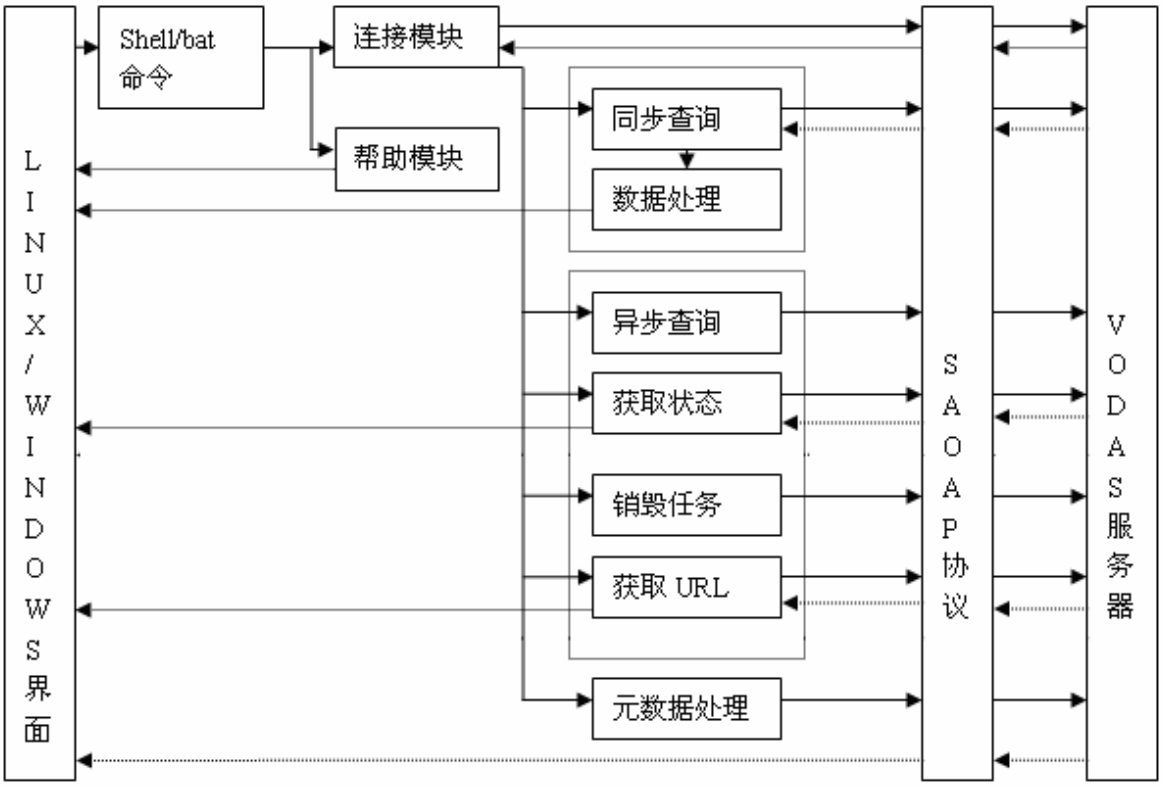

图 4.1.命令行客户端的设计， 实线箭头表示客户端发送请求的流程，
虚线箭头表示客户端接收服务器消息的流程

4.2.2 工作的一般流程

命令行客户端同样可以采用同步和异步方式进行查询，但是工作的流程与 GUI 不完全一样。下面根据图 4.1 的设计来说明命令行客户端工作的一般流程。

异步查询：

1. 客户端请求异步查询。用户在 Windows 或 Linux 客户端以命令行的方式发起异步查询请求。
2. 客户端向服务器端发起连接请求，连接成功并调用后台 DQI 的异步查询 asynQuery

接口，提交请求。循环 1。

3．调用服务器端 MI 的 getStatus 接口获取服务器端的查询任务状态或调用 MI 的 destoryJob 接口销毁正在运行的查询。

4．调用服务器提供 DAI 的 getTargetURL 接口获取查询结果数据保存地址并下载数据。对于请求状态为完成的任务，用户可以下载结果数据。

同步查询：

1．客户端请求同步查询。用户在 Windows 或 Linux 客户端以命令行的方式发起同步查询请求。

2．客户端向服务器端发起连接请求，连接成功并调用后台 DQI 的同步查询 synQuery 接口，提交请求。

3．用户一直等待，直到查询结果返回给客户端。

同步查询和异步查询请求方式与 GUI 客户端同，但是同步查询的结果数据仅仅返回给客户端，不作进一步的数据处理工作。

4.2.3 命令模块

命令行的客户端用户提交各项查询请求与该模块直接交互。这个模块分别用 Shell 脚本和批处理环境实现了一套在 Linux 和 Windows 平台下的操作命令，每条命令的基本实现流程如图 4.2 所示。

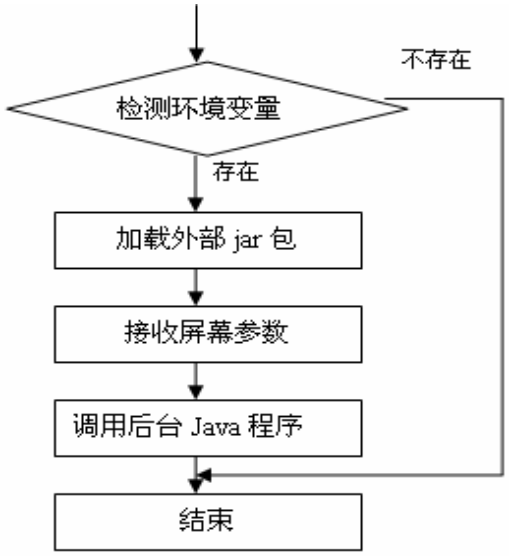

图 4.2 命令调用流程实现

4.2.4 后台调用模块

该模块主要用 Java 程序实现如何接收并判断通过命令传来的参数，然后按照服务器接口的标准进行转换，并启动调用接口的功能。

如图 4.1 所示，与 GUI 客户端类似，命令行客户端用户对各项查询的请求除帮助功能外都要先经过连接成功方可进行下一步的操作。由于命令行客户端的特殊性，在具体实现时将会有所区别。首先，用户向服务器端进行的每一次请求，都要进行连接操作各一次。因为每一次前台命令调用后台程序时，都经由一个主函数进行入口，重新开辟一个新的线程。其次，连接服务器的地址等配置信息会存放在一个配置文件里，由程序每次实现读入操作，而不需要用户频繁地进行手动输入。命令行客户端的元数据获取后呈现给用户的方式不同，它首先调用服务端 RMI 的接口，然后应用程序按照 XML 的组织文本形式展现给用户。其流程图如图 4.3 所示。

4.3 系统实现

4.3.1 后台程序实现

命令行客户端首先用面向对象的 Java 程序实现了一组支持客户端的接口来完成各项数据查询操作。在其之上通过 Linux 的 shell 脚本和 Windows 的批处理环境分别封装了各个接口，实现了适用于不同操作系统平台上的一组命令。在用 Java 具体实现的过程中，主要创建了如下几个类：

Class Commands_main：用于接收命令行发送过来的参数并判断下一步该调用哪个类完成相应的操作。

Class Connect：读取本地机器上服务器地址的配置信息，由客户端向服务器发起连接操作。

Class MetaData：主要负责系统中元数据的一系列操作和处理，它包含多个方法如 void ShowMeta()（实现元数据的 XML 格式化）、String[][] getResource()（得到资源元数据）、String[][] gettable(获取表元数据)、String[][] getColumn(获取表列元数据)和 Document parseXMLDocument(xml 字符串文档化)

Class SynQ：提交同步查询请求，并处理返回的结果数据，然后呈现给用户

Class AsynQ：提交异步查询请求。它同时包含与异步操作有关的 dataURL( )、jobStatus( )和 destory( )方法

Class Help：提供命令行客户端所有命令的用法。

4.3.2 前台命令实现

与用户最终交互的命令是通过 Linux 的 shell 脚本和 Windows 的批处理环境分别封装了后台的各个接口，下面分别以实现异步查询的 asyn 命令为例来说明前台命令的两种实现方式，图 4.4 是批处理环境实现 asyn 命令的主要源代码，如图 4.5 是用 shell 脚本实现 asyn 命令的主要源代码。它们的表达语法不同，但实现功能一样。首先，判断客户端的环境变量 JAVA_HOME 和 DAS_client_HOME 是否存在，如果存在，继而加载后台实现 JAVA 程序所需的类库，然后，从标准输入接收命令和参数，调用后台 JAVA 主程序。

4.3.3 用户命令接口

通过上述脚本或批处理的一系列封装，我们实现了扩展名为 sh 和 bat 的命令集，下面就对用户使用的每一个命令接口的输入输出进行详细的描述。

1） md 命令

i. 功能：

获取系统中指定的可用元数据信息

ii. 参数：

表 4.1 md 命令输入输出参数表

Input/output	参数类型及名称	说明
Input	String ResourceName	输入所要获取的资源名称
	String TableName	输入所要获取资源名称下的表名称
	String ColumnName	输入所要获取资源名称下表名中指定的列名
	None	无条件获取系统中所有可用的元数据信息
Output	String MetaData	格式化 XML 的字符串结果

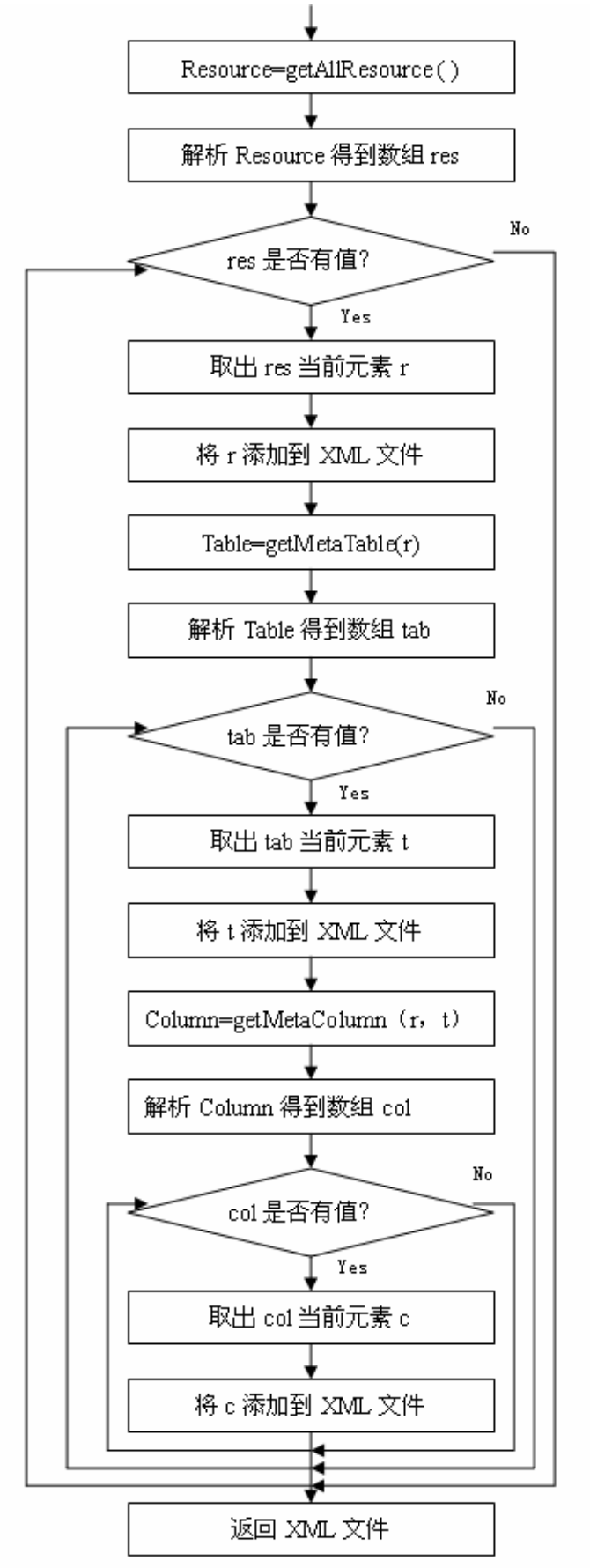

图 4.3 命令行客户端获取元数据流程

iii. 用法：

md [ResouceName [TableName [ColumnName]]]

2） syn 命令

i. 功能：

提交一个同步查询任务，查询结果以字符串形式输出到标准输出上

```
@echo off
if "%JAVA_HOME%" == "" goto nogl
goto jh
:nojh
    echo Error: JAVA_HOME not set
    goto end
:jh
if "%DAS_Client_HOME%" == "" goto nodasclient
    goto dasclient
:nodasclient
    echo Error: DAS_Client_HOME not set
    goto end
:dasclient
set CMD_LINE_ARGS=
:setupArgs
if %1a==a goto setCMD
set CMD_LINE_ARGS=%CMD_LINE_ARGS% %1
shift
goto setupArgs
:setCMD
if not "%VO-DAS_LIB%" =="" goto run
set VO-DAS_LIB = %DAS_Client_HOME%\lib\org_china_vo_das_engine_DAS.jar;...
set OGSA-DAI_HOME = %DAS_Client_HOME%\OGSA-DAI
set OGSA-DAI_LIB = %OGSA-DAI_HOME%\lib\exist.jar;...
set OGSA-DAI_ThirdParty = %OGSA-DAI_HOME%\thirdparty\activation.jar;...
set CLASSPATH = %CLASSPATH%;%VO-DAS_LIB%;%OGSA-DAI_ThirdParty%;%OGSA-DAI_LIB%
:run
rem echo %CMD_LINE_ARGS%
set _RUNJAVA=java -cp
if not "%JAVA_HOME%" == "" set _RUNJAVA="%JAVA_HOME%\bin\java"
%_RUNJAVA% DAS.Commands_main asyn %CMD_LINE_ARGS%
:end
```

图 4.4 批处理环境实现的 asyn 命令

```
####### MAIN BODY ##########
if [ ! -d "$JAVA_HOME" ] ; then
  echo "Error: JAVA_HOME invalid or not set: $JAVA_HOME" 1>&2
  exit 1
fi
if [ ! -d "$DAS_Client_HOME" ] ; then
  echo "Error: DAS_Client_HOME invalid or not set: $DAS_Client_HOME" 1>&2
  exit 1
fi
# echo "$@"
if [ "X$LOCALCLASSPATH" = "X" ] ; then
    VO_DAS_LIB = $DAS_Client_HOME/lib/org_china_vo_das_engine_DAS.jar:...

    OGSA_DAI_HOME = ${DAS_Client_HOME}/OGSA_DAI

    OGSA_DAI_LIB = ${OGSA_DAI_HOME}/lib/exist.jar:...
    OGSA_DAI_ThirdParty = $OGSA_DAI_HOME/thirdparty/activation.jar:...
    LOCALCLASSPATH=$VO_DAS_LIB # :$OGSA-DAI_LIB:$OGSA-DAI_ThirdParty
    # echo $LOCALCLASSPATH
fi
### SETUP OTHER VARIABLES ####
# echo "$@"
# echo "Please wait a minute..."
java -cp $CLASSPATH:$LOCALCLASSPATH DAS.Commands_main asyn "$@"
# echo "The process is over!"
```

图 4.5 shell 脚本实现 asyn 命令

ii. 参数：

表 4.2 syn 命令输入输出参数表

Input/Output	参数类型及名称	说明
Input	String adql	用 ADQL 语言描述的查询任务
	File adqlFile	指定 ADQL 语言描述查询任务的文件
	Int format	查询结果返回的格式
Output	String ResultData	输出到标准输出上的查询结果字符串

iii. 用法：

syn <adql |-f adqlFile> <format>

3） asyn 命令：

i. 功能：

提交一个异步查询任务，查询结果保存在指定的存储服务器上

ii. 参数：

表 4.3 asyn 命令输入输出参数表

Input/Output	参数类型及名称	说明
Input	String adql	用 ADQL 语言描述的查询任务
	File adqlFile	指定 ADQL 语言描述查询任务的文件
	Int format	查询结果返回的格式，可为 0、1、2、3
	URL StorageAddress	结果数据存储的地址
	File SessionFile	保存任务的 session 文件名
Output	None	

iii. 用法：

asyn <adql | -f adql> <format> <StorageAdress> <SessionFile>

4） jobstatus 命令

i. 功能：

查询已提交异步查询任务的执行状态

ii. 参数：

表 4.4 jobstatus 命令输入输出参数表

Input/Output	参数类型及名称	说明
Input	File SessionFile	保存任务 session 的文件名
Output	String status	任务状态：running 、error、completed、

iii. 用法：

jobstatus <SessionFile>

5） dataurl 命令

i. 功能：

获取异步查询任务结果数据保存地址

ii. 参数：

表 4.5 dataurl 命令输入输出参数表

Input/Output	参数类型及名称	说明
Input	File SessionFile	保存任务 session 的文件名
Output	URL url	查询结果数据保存的地址

iii. 用法：

dataurl <SessionFile >

6） destory 命令

i. 功能：

销毁正在服务器上执行的任务

ii. 参数：

表 4.6 destory 命令输入输出参数表

Input/Output	参数类型及名称	说明
Input	File SessionFile	保存任务 session 的文件名
Output	Node	

iii. 用法：

destroy <SessionFile>

另外，上述所有命令如果带-help 参数运行，将输出每条命令详细的使用帮助信息。

4.4 命令行客户端的应用

现在我们以在 Linux 环境中的 asyn 命令为例，来说明命令行的基本操作使用过程。如图 4.6 所示。

首先用户可以在提示符下输入：

asyn.sh -help

获取同步查询命令 asyn 的用法，如图 4.5 的上部显示了 asyn 命令的使用格式和参数以及每

个参数所代表的含义。然后就可以编辑 asyn 命令行提交同步查询任务：

asyn.sh –f adqlSample2 0 ftp://vodas:vodasvodas@159.226.169.222/test/ vodas.dat Session

其中，查询语句 ADQL 放在文件 adqlSample2 中，以 tab 键隔开的文本形式(用 0 表示)的结果数据存于 *ftp://vodas:vodasvodas@159.226.169.222/test* 目录下 *vodas.dat* 文件中，同时，将查询提交时服务器返回唯一标识查询任务的 session 内容保存在文件 Session 中。查询提交一段时间后，使用命令：

jobstatus session

查询当前任务运行的状态，如图所示命令返回的结果为“COMPLETED”，表明执行完成，用户即可从上述 ftp 地址下载该数据文件 vodas.dat。

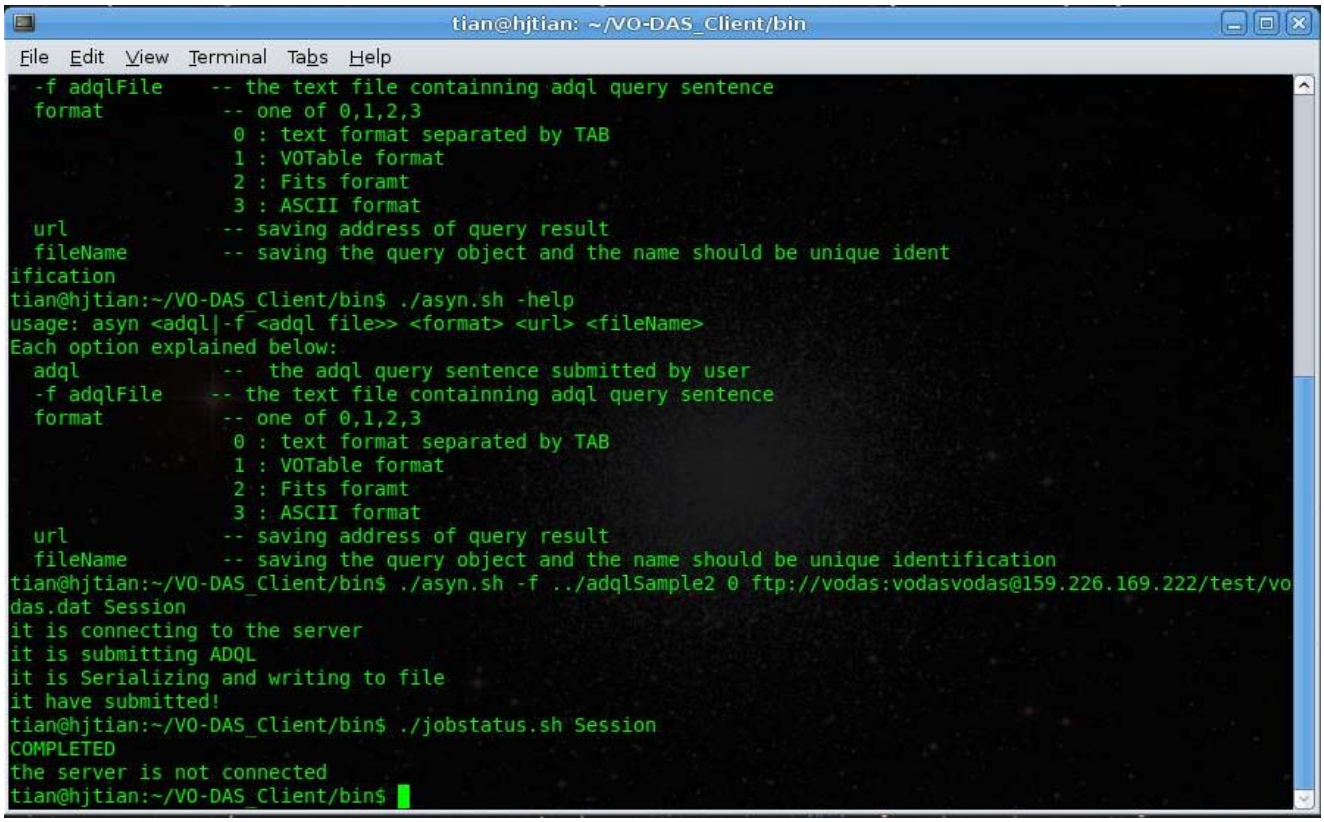

图 4.6 命令行客户端

4.5 本章小结

本部分所实现的命令行客户端是不同于GUI的VO-DAS的另外一种客户端。它以命令行的方式向用户提供了访问VO-DAS的平台。它的实现分为两大部分，后台用JAVA程序封

装了VO-DAS提供的Web服务接口，前台部分用Shell脚本和批处理在JAVA程序之上又进行了再次封装，从而形成了可在Linux和Windows系统上运行的一套命令。本章主要介绍了命令行客户端的总体结构、工作的流程、命令模块和后台调用模块的设计与实现。

命令行客户端用户不需要图形界面的操作，只要通过命令行即可完成用户的查询请求。命令行方式还可以集成到用户自己的程序中去，满足了高级用户对数据的海量处理和分析。

第五章 VO-DAS 系统集成与部署

本章将详细介绍我们设计的VO-DAS数据结点配置工具的实现以及VO-DAS系统的部署。数据结点配置工具设计的目的在于对拥有大量天文观测数据的用户使用此工具能够方便地将这些数据投放到 VO-DAS 系统中，实现数据的共享。作为 VO-DAS 系统投入使用前的一个必备工作，我们还介绍了 VO-DAS 系统部署工作。

5.1 数据结点配置工具的设计与实现

5.1.1 数据结点配置工具需求分析

在前面两章中我们已经提及到，为了最大限度地发挥 VO-DAS 的潜力和使用效率，我们已经实现了VO-DAS的两种客户端，即GUI客户端和命令行客户端。以此来扩大VO-DAS应用层面上的用户群。更进一步地说，这些天文学家能否从 VO-DAS 中进行新的研究和发现，还取决于系统中可供天文学家存取使用数据的价值及其数量。因此，这就需要一种方法将天文学家通过大型望远镜观测到的手头上闲置的有利用价值的海量数据能够方便地将它们与科学界同仁进行共享，共同致力于新的科学研究。在 VO-DAS 系统中，Data Node 服务解决了系统数据共享这一难题。但是天文学家面临的又一难题是生成 Data Node 服务所需的专业技术以及工作量以显得非常棘手。

首先，我们来了解一下它的共享实现过程，如图 5.1[1]所示。Data Node 是 VO-DAS 系统中所有访问数据的来源，由它来提供数据资源的服务。如果用户手头有一批观测数据，希望这批数据能投放到 VO-DAS 中去供别人共享，这时数据用户必须配置 Data Node 服务。在 Data Node 上我们使用 OGSA-DAI 网格中间件来封装各种不同类型的数据库或文件，给程序提供统一的访问接口。因此，用户首先要安装 OGSA-DAI，然后在 OGSA-DAI 的基础上手动配置自己数据服务，包括部署数据服务（Data Service）、数据服务资源（Data Service Resource ）和通过数据服务发布数据服务资源等一系列的操作过程。然后，根据已配置好的资源手动生成相应的元数据文件（metadata.xml），最后将它注册到 Registry 服务中。这时 Data Node 基本生成，VO-DAS 启动时会检测到新配置的这些数据。可以看到，生成 Data Node 这一系列繁琐的操作过程对于不太熟悉相关技术的天文学家来说，毫无疑问是比较困

难的。因此，我们实现的数据结点配置工具旨在将复杂的操作过程以图形界面的方式呈现给客户，使用户能够轻易地将拥有的数据源配置和共享到 VO-DAS 系统中。

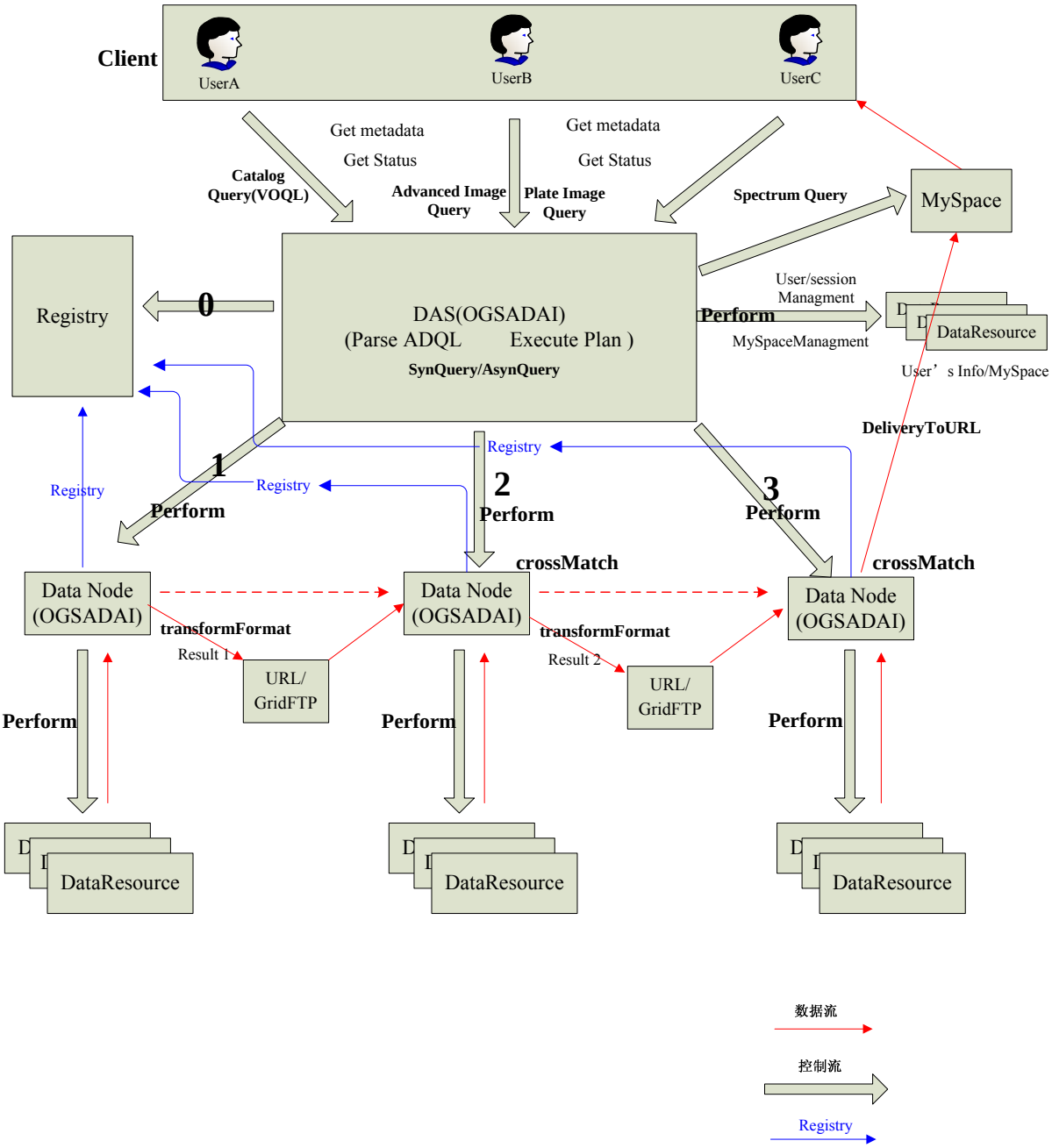

图 5.1 VO-DAS 的结构图

5.1.2 系统设计的总体结构

根据对系统进行功能需求分析，它由界面模块、数组资源部署、数据访问处理、元数据配置和帮助信息功能模块组成，如图 5.2 所示：

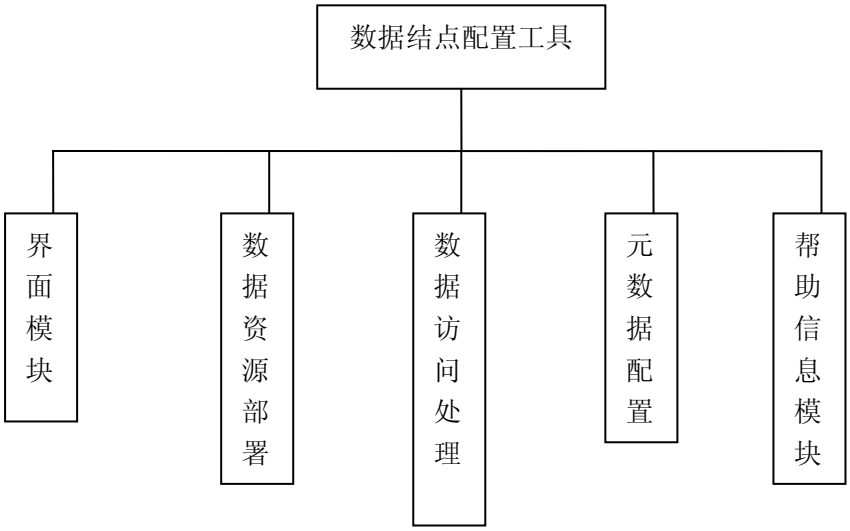

图 5.2 数据结点配置工具功能模块图

界面模块：负责展示用户功能操作模块、显示系统数据以及接收用户对信息的修改。

数据资源部署：负责将用户的数据通过 OGSA-DAI 网格中间件部署成数据资源服务。

数据访问处理模块：主要根据用户部署资源的信息读取本地或远程的物理数据库信息，并以用户易读的树形结构方式呈现给用户。

元数据配置模块：负责将数据库中的元数据结构信息进行配置，并以 XML 文件存储在本地机器上。

帮助信息模块：主要为用户使用此工具提供一些帮助信息。

5.1.3 系统工作流程

数据结点配置工具的工作流程：

1. 应用程序启动时，系统首先进行初始化，检查 OGSA-DAI 中先前的资源服务配置信息，根据此信息读取数据库中的元数据内容，可视化给客户端。
2. 程序启动后，用户根据元数据的显示内容了解系统服务中存在哪些数据库或表。如果显示的元数据内容中有用户所需的数据，就可对它们进行配置。如果没有，此时用户就可以将自己的数据资源部署到 OGSA-DAI 中。
3. 资源部署工作一旦完成，程序自动地读取已部署资源的物理元数据并进行数据处理后呈现给用户。
4. 对于已经部署到数据结点中的数据，用户就可以进行元数据的配置工作了，

包括修改和确认等。

5. 数据的配置工作完成后，系统自动生成一个元数据配置文件 metadata.xml，这个配置文件包含了用户提供给系统使用数据资源的所有元数据信息。

最终由用户将这个文件的 URL 手动注册到 Registry（注册服务）中。通过这一系列操作过程，用户配置的 Data Node 上数据就能够被 VO-DAS 系统所识别，最终用户随意的查询到自己想要的数据。

5.1.4 数据库访问

JDBC 是 Java 数据库连接应用程序接口（Java Database Connectivity API）的简称，Sun 提供的一套数据库编程接口 API 函数，由 Java 语言编写的类、界面组成。用 JDBC 写的程序能够自动地将 SQL 语句传送给相应的数据库管理系统。不但如此，使用 Java 编写的应用程序可以在任何支持 Java 的平台上运行，不必在不同的平台上编写不同的应用。Java 和 JDBC 的结合可以让开发人员在开发数据库应用程序时真正实现“Write Once，Run Everywhere！”

JDBC 的体系结构如图 5.3 所示。图中 JDBC API 的作用就是屏蔽不同的数据库驱动程序之间的差别，使得程序设计人员有一个标准的、纯 Java 的数据库程序设计接口，为在 Java 中访问任意类型的数据库提供技术支持。驱动程序管理器（Driver Manager）为应用程序装载数据库驱动程序。数据库驱动程序是与具体的数据库相关的，用于向数据库提交 SQL 请求。

VO-DAS 系统的数据结点配置工具是客户端/服务器的桌面应用富客户端，数据库一般在服务器端。因此 JDBC 的驱动程序部署在客户端。如图 5.4 所示。

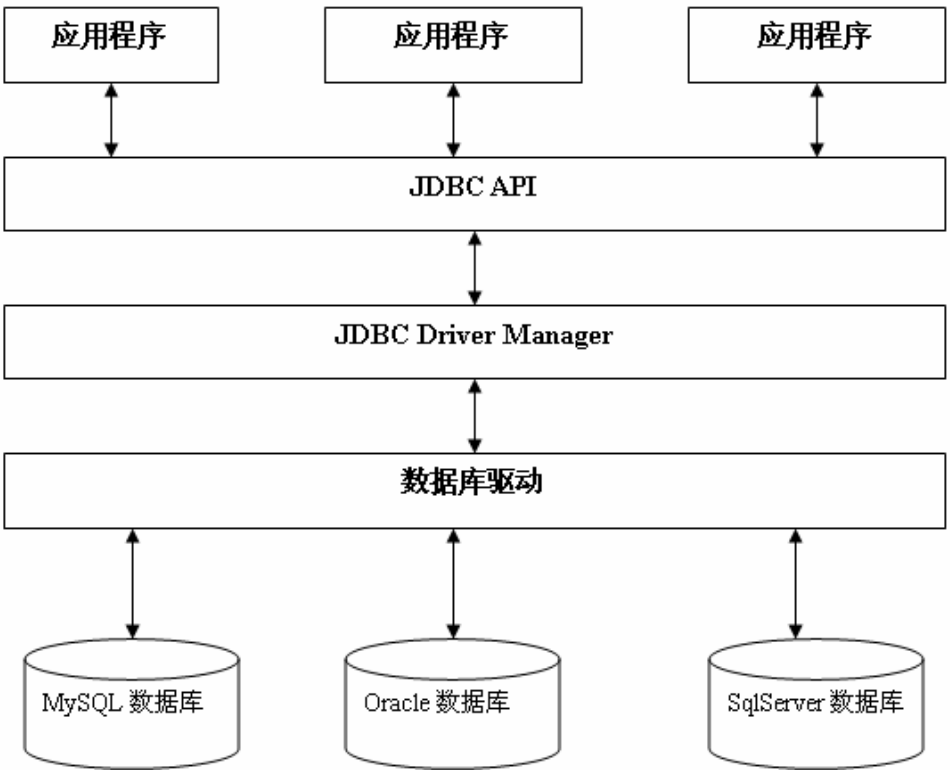

图 5.3 JDBC 的体系结构

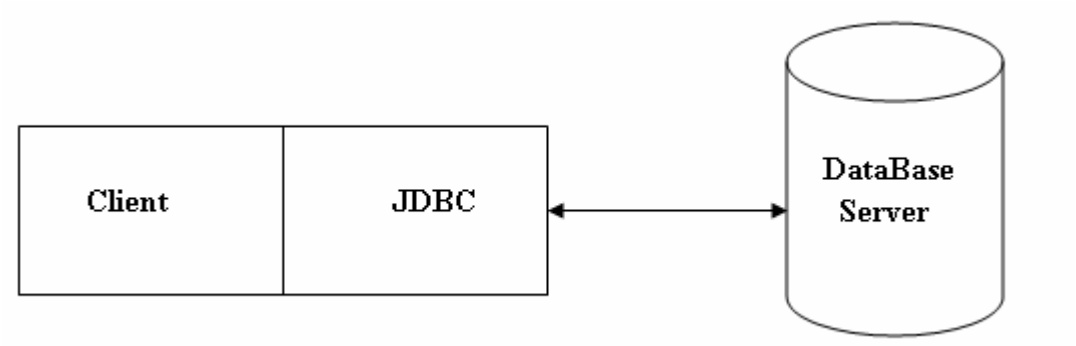

图 5.4 客户端/服务器应用

在具体实现过程中，我们需要使用到 JDBC 的几个基本类，这些类能够加载一个驱动程序，连接数据库，生成并执行一条 SQL 语句并检查结果。另外，结点配置工具主要是获取数据库表格的元数据，因此，还需要获取数据库结构以及它的表格的附加信息，如：数据库表的名字、列名以及类型等。以下是实现这些功能的类的接口。

Java.sql.DriverManager -- 加载 JDBC 驱动程序并管理数据库连接

Java.sql.Connection -- 连接数据库

Java.sql.Statement -- 管理连接中的 SQL 语句

Java.sql.ResultSet -- 访问执行语句的结果

Java.sql.DatabaseMetaData -- 获得数据库中的元数据信息

Java.sql.ResultSetMetaData -- 查询数据库返回结果集中的元数据信息

以下代码根据上述接口类，实现了驱动程序的加载，根据 dbURL（数据库存放地址）、dbUser（用户名）、dbPassword（密码）连接数据库，并获取数据库中每个表名的元数据信息。

```
Class.forName("org.gjt.mm.mysql.Driver").newInstance();
Connection conn = DriverManager.getConnection(dbURL, dbUser, dbPassword);
DatabaseMetaData meta1 = conn.getMetaData();
ResultSet mrs = meta1.getTables(null, null, null,new String[] { "TABLE" });
```

此时，mrs 变量中就获取了指定数据库下所有的表名。从而可以进一步获取每个表中的结构信息，如：列名、宽度、类型等等，代码实现如下：

```
String sql = "select * from " + tableNames.get(k) + " limit 1";
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery(sql);
ResultSetMetaData meta = rs.getMetaData();
```

这样，通过 JDBC 就可以获取到数据结点中所有的元数据信息，由客户端程序作进一步地处理并以易读的树形结构呈现给用户。

5.1.5 资源部署

Ant是Apache软件基金会JAKARTA目录中的一个子项目，用于简单或复杂Java工程的自动化构建、部署工具。它具有以下的优点。

1）跨平台性，Ant 是纯 Java 语言编写的，具有很好的跨平台性。

2）操作简单，Ant 是由一个内置任务和可选任务组成的。

3）Ant 运行时需要一个 XML 文件(构建文件)；通过调用 target 树，就可以执行各种 task。每个 task 实现了特定接口对象。由于 Ant 构建文件是 XML 格式的文件，所以容易维护和书写，而且结构很清晰。

4）Ant 可以集成到开发环境中。由于 Ant 的跨平台性和操作简单的特点，它很容易集

成到一些开发环境中去。

Ant 的构建文件是一个处理工程构建或部署给定阶段的目标的集合。这个文件包含有很多标签及其各个标签的属性，其中每个构建文件只存在一个 Project 项目标签，项目标签下可有一个或多个 Target 标签。OGSA-DAI 安装完成后，在其根目录下会自动生成构建文件。OGSA-DAI 的数据资源部署就通过这个构建文件来完成。在进行部署数据资源时，将会使用到这个构建文件中如下 Target 标签和它的相关属性及值。

deployService –Ddai.container=/path/to/Web/services/container –Ddai.service.name=service/name

将用户指定的服务名称部署到指定的容器中。

deployResource –Ddai.container=/path/to/Web/services/container –Ddai.resource.file=DAI-SERVICE-RESOURCE-FILE

将用户指定的资源文件部署到指定的容器中，这个资源文件记录了用户配置资源的一些相关信息，包括数据资源类型、产品、URI、ID 以及数据资源访问的用户名、密码和数据驱动类名等。

exposeResource –Ddai.container=/path/to/Web/services/container –Ddai.service.name=service/name –Ddai.resource.id=ResourceID

通过数据服务来发布数据服务资源。其资源部署实现的流程图 5.5 所示。

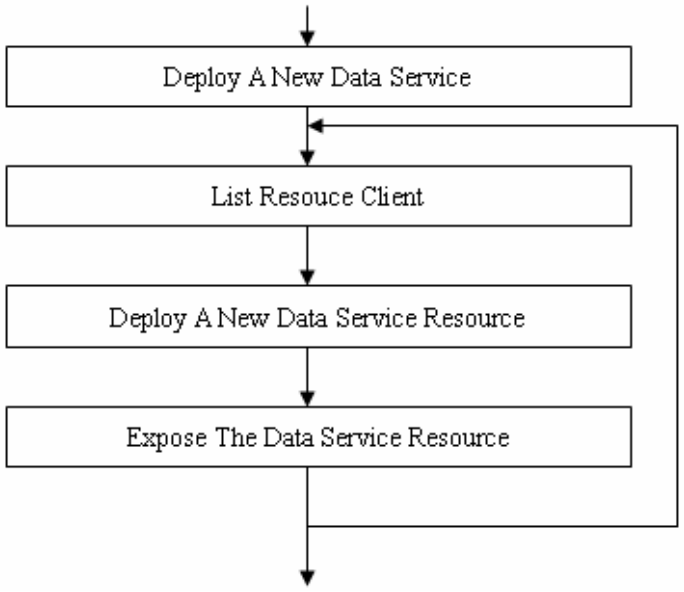

图 5.5 OGSA-DAI 资源部署流程

OGSA-DAI 使用构建文件部署资源可以通过两种方式来进行：命令行和图形用户界面。命令行方式对于不熟悉计算机操作系统的数据用户来说比较难于操作。图形用户界面方式相对来说比较简单。因此，在实现数据结点配置工具时，我们将 Ant 集成到所使用的 eclipse 开发环境中，实现编程调用 ant 构建文件的图形用户界面功能。这样就屏蔽掉了 Ant 命令的使用细节。

以 Java 编程调用 Ant 的程序代码如下代码所示。

```
File buildFile = new
File(main_frame.mReadResource.getEnv().get("OGSADAI_HOME")+"\\build.xml");
Project p = new Project();
 p.addBuildListener(consoleLogger);
 p.fireBuildStarted();
 p.init();
 ProjectHelper helper = ProjectHelper.getProjectHelper();
 helper.parse(p, buildFile);
  p.executeTarget("guiDeployService");
```

首先，它使用 OGSA-DAI 的构建文件创建一个文件对象，然后创建一个新的 Project 对象。Project 是 Ant 用于表示带有其所有目标、任务和属性的 Ant 项目的 Java 类。它初始化此项目；这将导致该项目执行某一内部安装程序。然后，定位缺省的 ProjectHelper，并使用该 ProjectHelper 分析此构建文件，并将来自构建文件的信息填充到项目中。接下来，通过传递目标名称，例如使用 “guiDeployService”来执行此构建文件中的 guiDeployService 目标。但是即使所有这些操作都成功完成，仍看不到任何输出，因此添加一个 BuildLogger 作为侦听器，这样，它就可以接收构建过程中发出的事件通知。对于这种情况，注册最简单的记录器和 DefaultLogger，并将输出消息直接输出到标准输出，将错误消息直接输出到标准错误。要做到这一点，在创建项目之后初始化项目之前添加以下代码行：

```
DefaultLogger consoleLogger = new DefaultLogger();
consoleLogger.setErrorPrintStream(System.err);
consoleLogger.setOutputPrintStream(System.out);
consoleLogger.setMessageOutputLevel(Project.MSG_INFO);
p.addBuildListener(consoleLogger);
```

5.1.6 元数据配置

元数据配置就是用户根据元数据区中显示的数据结构信息对其进行编辑和修改。元数据区中显示的数据以树形结构方式表示数据资源、数据表及其列的结构层次关系。对元数据的配置信息包括资源的名字（name）和描述信息（description）、资源表的 name 和 description 以及表列的 name、单位（unit）、type、description 和统一内容描述（UCD）等信息。这些信息最终以 metadata.xml 文件名存在本地机器的 web 服务容器中，这个文件的配置信息如图 5.6 所示。

```
<?xml version="1.0" ?>
<VOTABLE version="1.1" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:noNamespaceSchemaLocation="http://www.ivoa.net/xml/VOTable/VOTable/v1.1">
 <COOSYS ID="J2000" equinox="J2000." epoch="J2000." system="eq_FK5" />
<RESOURCE name="DataServiceResourceDAItest">
 <TABLE name="sdss">
    <DESCRIPTION>DAItest_sdss_DESCRIPTION</DESCRIPTION>
    <FIELD name="ra" ID="col1" ucd="pos.eq.ra;meta.main" ref="J2000" datatype="float" width="6" precision="2" unit="deg" />
    <FIELD name="decl" ID="col2" ucd="pos.eq.dec;meta.main" ref="J2000" datatype="float" width="6" precision="2" unit="deg" />
 </TABLE>
<TABLE name="1rxs">
    <DESCRIPTION>DAItest_rosat_Description</DESCRIPTION>
    <FIELD name="ra" ID="col1" ucd="pos.eq.ra;meta.main" ref="J2000" datatype="double" width="9" precision="5" unit="deg" />
    <FIELD name="decl" ID="col2" ucd="pos.eq.dec;meta.main" ref="J2000" datatype="double" width="9" precision="5" unit="deg" />
    <FIELD name="PosErr" ID="col3" ucd="pos.eq.PosErr;meta.main" ref="J2000" datatype="tinyint" width="2" precision="0" unit="deg"/>
    <FIELD name="Count" ID="col4" ucd="pos.eq.Count;meta.main" ref="J2000" datatype="double" width="9" precision="3" unit="deg" />
</TABLE>
<TABLE name="trc_1">
    <DESCRIPTION>DAItest_trc_1_Description</DESCRIPTION>
    <FIELD name="RAdeg" ID="col1" ucd="pos.eq.ra;meta.main" ref="J2000" datatype="double" width="12" precision="8" unit="deg" />
    <FIELD name="DEdeg" ID="col2" ucd="pos.eq.dec;meta.main" ref="J2000" datatype="double" width="12" precision="8" unit="deg" />
    <FIELD name="BT" ID="col3" ucd="pos.eq.BT;meta.main" ref="J2000" datatype="float" width="6" precision="3" unit="deg" />
    <FIELD name="VT" ID="col4" ucd="pos.eq.VT;meta.main" ref="J2000" datatype="float" width="6" precision="3" unit="deg" />
</TABLE>
</RESOURCE>
</VOTABLE>
```

图 5.6 配置工具生成的 metadata.xml 文件格式

从这个配置文件中，我们可知这个数据结点配置包含 DataServiceResourceDAItest 的资源，其下有三个表格 sdss、1rxs 和 trc_1，每个表下又包含了一系列的字段及其属性的描述。这个配置文件生成以后，用户通过将此文件的 URL 注册到 Regsitry 中，就完成了实现数据结点的生成。VO-DAS 系统启动时就可发现这些数据，用户通过 VO-DAS 的客户端就可以了解到这些元数据的信息以及查询到自己想要的数据。

5.1.7 实现工具及用户界面

配置工具的具体实现使用 JAVA 语言作为主要编程语言，我们使用了一个基于 Java 的开放源代码构建使用工具 Ant，它具有丰富的功能库和高度的可扩展性，使用 ANT 能够方

便地集成 OGSA-DAI 基于 ant 构建文件的部署过程。工具开发采用 ECLIPSE 集成开发环境，它是开源软件工具，内置增量式 JAVA 编译器，可以让 ECLIPSE 使用项目需求的 ANT，符合项目开发需求，同时，ECLIPSE 为实现修改的自动化提供良好的支持，另外，具有一种相当强大的代码格式化功能，这些特性大大提高了开发人员的工作效率。

以上述工具实现的桌面应用程序在启动时，就会出现如图 5.7 所示的资源部署界面。左下标签为“metadata”用于显示部署资源中的元数据信息，右下是用户编辑和配置元数据结构信息的区域。在标签为“Deploy Resource”部分是资源部署区，每个按钮分别实现了集成 OGSA-DAI 的部署过程，其界面如图 5.8 所示。

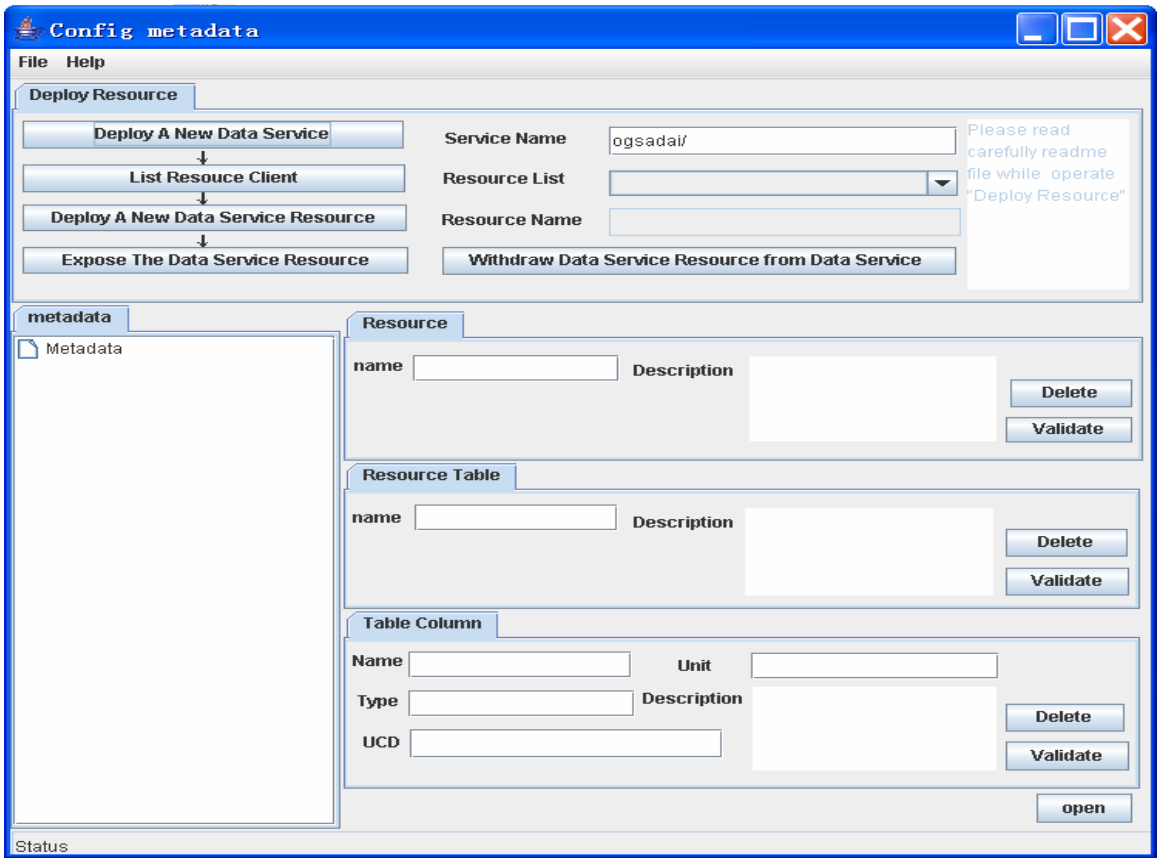

图 5.7 配置工具用户界面

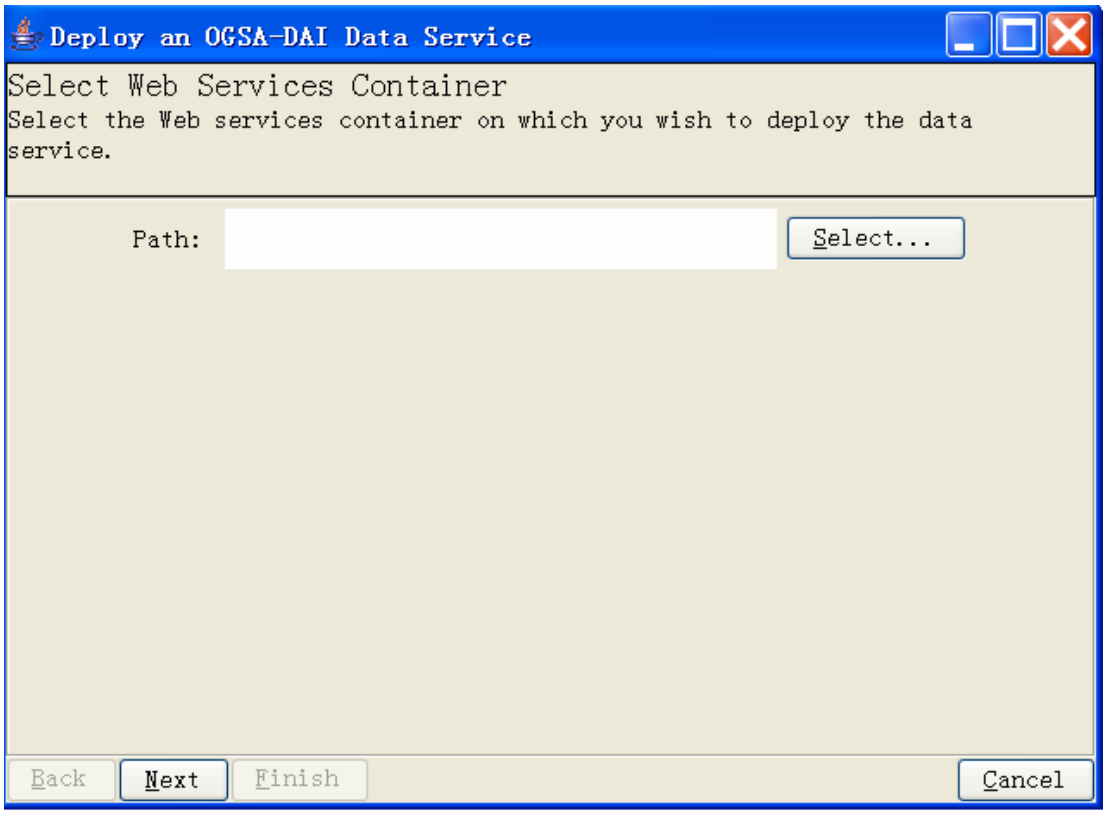

图 5.8 数据资源部署界面

5.2 VO-DAS 系统的集成

5.2.1 VO-DAS 系统的集成环境

前面我们已经详细地介绍了 VO-DAS 系统与最终用户交互、提供查询界面的客户端以及对数据实施共享的 Data Node，这仅仅是 VO-DAS 系统的组成部分。VO-DAS 系统不是一个单一的服务，而是由一组相互联系的服务组成一个完整的数据服务系统。作为 VO-DAS 系统最基本的服务或组件主要有 5 个部分：VO-DAS 服务(VO-DAS Server)、数据结点(Data Node)、VO 注册服务(VO Registry)、存储服务(Storage Server)和客户端（Client）。如表 5.1 所示。其中 VO-DAS Server 是整个系统的核心，负责 DAS 系统的任务调度和作业执行状态的监控，既及时地响应客户端的要求，又负责在调度过程中正确地对 Data Node 进行数据存取；Data Node 是系统中的数据源层，包括星表、图像、光谱等形式的数据库服务器，它使用 OGSA-DAI 来实现数据的管理，对不同类型的天文数据资源进行封装，为应用提供统一的访问接口，对于数据的数据即资源元数据(Resource Metadata)采用 VOTable 予以描

述；VO Registry 向 VO 应用提供数据和服务等可用资源的发现机制；Storage Server 则实现

表 5.1 VO-DAS 系统的组成

组件名称	说明
VO-DAS Server	在一个 VO-DAS 系统内至少有一个 VO-DAS 服务器，它直接和客户程序进行交互，接受客户端的数据查询请求，并替客户端完成网格环境下的查询
DataNode	用于封装数据资源成为符合 WSRF 的网格服务，它会注册到 VO Registry 上面
VO Registry	虚拟天文台注册服务器，可以是一个共享的组件。所有可用的 DataNode 必须在此注册才能够被发现
Storage Server	支持 FTP 和 GridFTP 的存储服务，用来存放查询的结果数据
Client	系统至少提供一个客户端给最终用户用来访问 VO-DAS 服务器，从而提交查询任务等操作

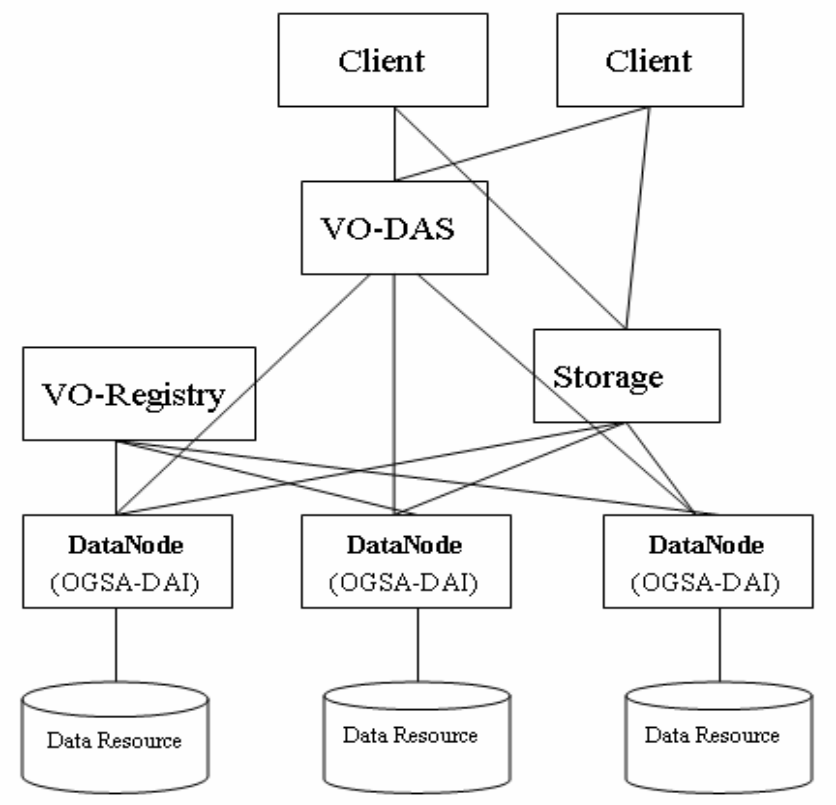

图 5.9 VO-DAS 系统的组件关联图

了网络化数据存储能力服务。这些服务之间依靠通信协议进行协作。它们之间的关联见图 5.9 所示。

合理地部署 VO-DAS 系统中既相对独立又相互联系的服务，使其成为一个逻辑的整体，为用户的请求做出响应，是数据访问必不可少的环节。以上所述的各种服务在部署时对环境具有一定的依赖程度，各种服务只有搭建在理想的环境中，才能真正实现服务的功能，与其他服务进行协作。如表 5.2 所示，表明了 VO-DAS 系统各个部分在部署时对环境的要求。

表 5.2 VO-DAS 的系统部署环境

服务名称		部署环境
VO-DAS Server		Java: SUN JDK Container: Apache Tomcat Web Service: Globus Toolkits Java WS Core Middleware: OGSA-DAI WSRF Database: MySQL/eXist/DB2/Oracle/PostgreSQL/SQL Server Others: Apache Ant
Data Node		Java: SUN JDK Container: Apache Tomcat Web Service: Globus Toolkits Java WS Core Middleware: OGSA-DAI WSRF Database: MySQL/eXist/DB2/Oracle/PostgreSQL/SQL Server Others: Apache Ant
VO Registry		推荐采用 AstroGrid 项目提供的 Registry 系统
Storage Server		目前使用 ftp 服务，推荐采用 AstroGrid 项目提供的 VOSpace 系统
Client	GUI 客户端	Java: SUN JDK、Others:PLASTIC
	命令行客户端	Java: SUN JDK、Others: Apache Ant

表中 “部署环境”列是相对应的各个服务器上的软件要求，它们之间的安装也有相互依赖的先后关系，读者可参照其它的相关资料。

5.2.2 VO-DAS 系统的部署

配置好各个服务器上的环境以后，就可以将各个服务分别搭建在相对应的服务器上，使他们进行相互协作与通信。以下是 VO-DAS 系统服务部署说明。

(1) VO-DAS Server 主要是对 DAS 服务的 gar 包及其配置文件 das.xml 的部署。其中，gar 包是通过 GLOBUS 中 bin 目录下的 globus-deploy-gar 命令将它部署在 GLOBUS 的容器中，DAS 的配置文件描述系统的静态参数，如 DAS 系统运行时的工作线程数、临时文件路径以及用户数据库的路径等。在开机时通过脚本来实现自启动 GLOBUS 的容器和自动配置 das 配置文件的参数。

(2) Data Node 的部署仅需运行数据资源结点配置工具的 jar 包，通过它的图形界面方式来实现天文数据资源的发布过程，其结果在本地生成元数据描述文件(metadata.xml)。

(3)将上述过程中生成的 metadata.xml 文件的 URI 通过 ASTROGRID Registry 系统注册到 VO Registry，在启动 Registry 服务的过程中由此来读取 Data Node 上的元数据。

(4) Storage Server 支持 AstroGrid 项目提供的 VOSpace 系统，同时允许用户指定 FTP 或 GridFTP 服务。

(5) 最后，对 GUI 客户端的用户来说，只要运行 Client 的 jar 包程序，实现与指定 VO-DAS Server 的连接，用户即可对 VO-DAS 进行数据访问了。命令行的客户端在本地机器需配置好 JAVA 环境变量和命令存放的目录，即可用命令的方式实现对 VO-DAS 的数据访问。

上述步骤中提及的程序包、配置文件以及更详细的文档都等可以通过中国虚拟天文台 VO-DAS 网站(das.china-vo.org)得到。

5.3 本章小结

数据结点配置工具为数据用户将观测的数据共享给 VO-DAS 提供了便捷的方式，使 VO-DAS 从另外一个角度发挥它的潜力和使用效率。工具能够以图形界面的方式完成 OGSA-DAI 数据资源的部署和已部署资源元数据的自动生成。本章介绍了结点配置工具的设计与实现过程。其中包括系统的总体结构、工作流程、数据库访问、元数据配置等。

合理地部署 VO-DAS 系统中既相对独立又相互联系的服务，使其成为一个逻辑的整体，为用户的请求做出响应，是数据访问必不可少的环节。VO-DAS 系统的集成部分则详

细介绍了 VO-DAS 系统的组件及其关联、系统集成的环境以及部署的步骤等。最终使相互独立的服务之间依靠通信协议能够协作、合理地响应用户的请求。

参 考 文 献

[1]田海俊. 虚拟天文台数据访问服务(VO-DAS)之任务调度研究及 VO-DAS 的应用. master thesis,华中师范大学, 2007.

第六章 科学应用范例

前面几章我们已经介绍了 VO-DAS 系统的 GUI 和命令行客户端以及数据结点配置工具的设计与实现，并且对 VO-DAS 系统的整个部署过程也做了详细的描述。本章我们就以“通过星流的运动轨迹研究银河系的引力势”为科学范例，说明用户使用 VO-DAS 系统在系统部署、结点配置以及科学应用的整体过程。

6.1 科学范例目的

我们科学范例的目的是通过星流的运动轨迹估算出银河系的引力势。恒星的运动轨道决定了在银河系的引力势。因此，通过星流的运动轨迹就可以估算出银河系的引力势。在这个的轨道上，我们选取一类距离最容易确定的恒星即 BHB 星（蓝水平分支星）来估算引力势。首先，确定 BHB 星所在的 A 型星区域，如图 6.1[1]所示。由于 A 型星中的 BHB 星会受到 BS 星（蓝离散星）的干扰，然后就需要分离出 BS 星来得到 BHB 星，如图 6.2[1]所示。将这些 BHB 星用工具将其可视化即可得到 BHB 星的运动轨迹，从而可以进一步估算出银河系的引力势。

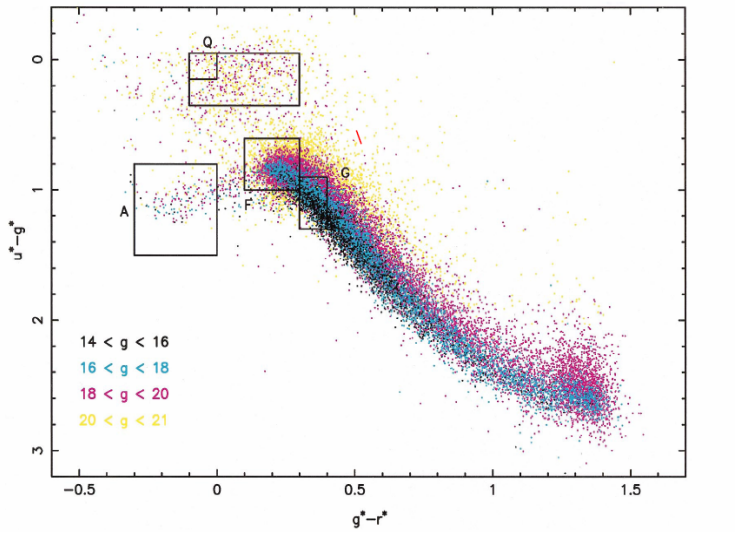

图 6.1 天赤道上 SDSS 测试数据的双色图。此图来自 BRAIN Yanny et al.(2000) FIG.1.

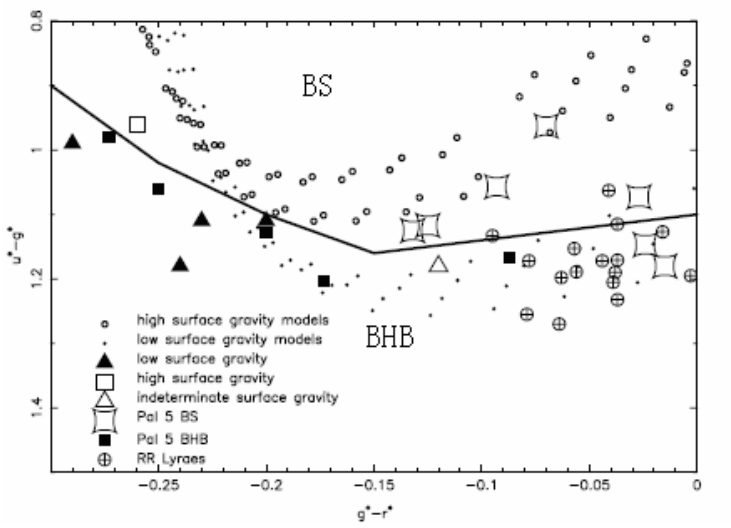

图 6.2 已测表面重力的恒星及其相关恒星模型双色图。

此图来自 BRAIN Yanny et al.(2000) FIG.10.

6.2 科学范例操作步骤

此范例中我们选取 SDSS 的 Star2 为样本数据，并给出一个完整的应用思路。

第一步：部署 VO-DAS 系统的各个组成部件：VO-DAS 服务、数据结点、VO 注册服务、存储服务和客户端，使其正确地响应用户对 VO-DAS 系统的请求。

第二步：利用数据结点配置工具把 SDSS 的 Star2 数据通过一系列的部署过程将数据发布出来，并对其元数据进行配置，最终生成元数据文件 metadata.xml。将此文件手动注册到 Registry 服务，VO-DAS 服务启动即可检测，Star2 的数据结点已经生成。

第三步：构造查询 Star2 表中 A 型星数据的 ADQL 表达式，如图 6.1 所示。通过 VO-DAS 的 GUI 客户端向服务器发送此请求，并返回查询结果，如图 6.2 所示。

第四步：确保 TOPCAT 是运行状态，启动它内置 PLASTIC HUB 并且注册到 HUB。将返回给客户端的结果直接发送到 TOPCAT。

```
SELECT s.glon,s.glat,s.g,s.u,s.r
FROM SDSS:Star2 s
WHERE s.g>17.2 and s.g<20.2 and s.g- s.r>=-0.3 and
s.g-s.r<=0 and s.u-s.g>=0.8 and s.u-s.g<=1.2.
```

图 6.3　科学范例 ADQL 查询语句

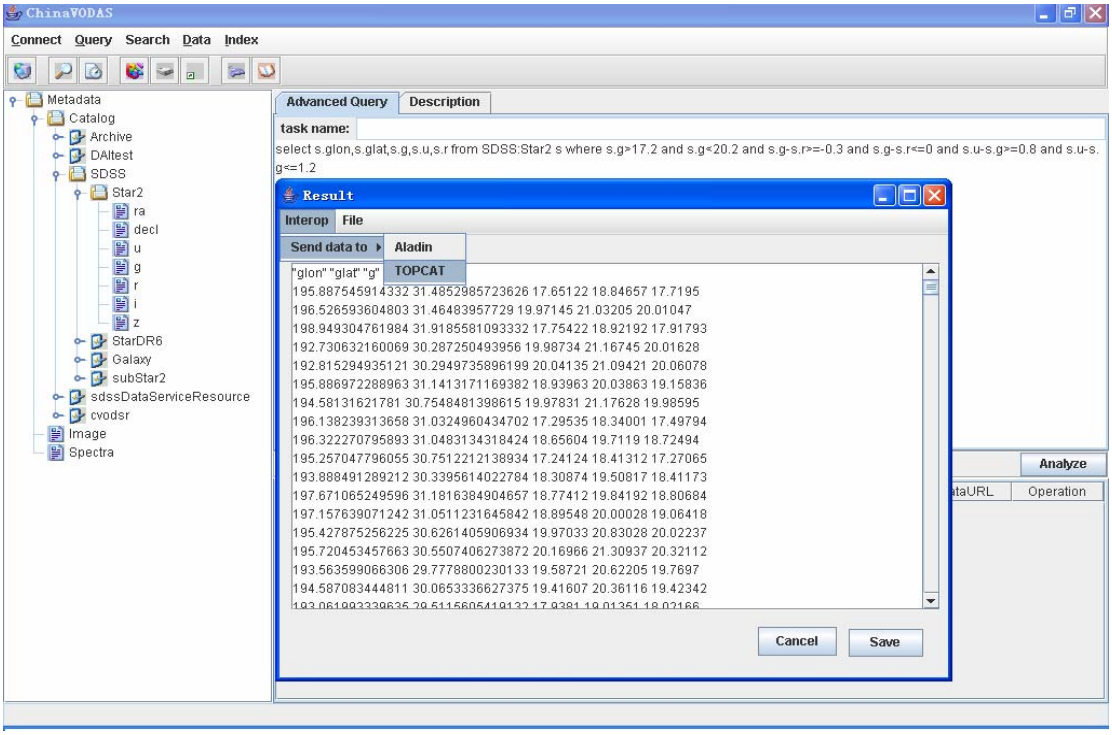

图 6.4 VO-DAS GUI 客户端同步数据查询

第五步：在 TOPCAT 中构建筛选 BHB 星的表达式，如下所示，得到新的结果数据。

$$u\text{-}4*g+3*r>=1.8 \;||\; u\text{-}2*g+r>=1.3 \;||\; u\text{-}0.67*g\text{-}0.33*r>=1.1$$

第六步：求出 BHB 星的距离，进一步得到它在直角坐标系中的位置。

第七步：对第六步的数据利用 TOPCAT 工具画出 3D 图，从而估算出 BHB 星的运动轨迹，如图 6.3 所示，图中的窄带状部分就是我们目标轨迹。科学家根据这个轨迹就可以进一步地分析出银河系的引力势。在此步，数据用户还可可视化数据的球面图、散点图和密度图找出更为准确的轨迹范围。

6.3 结论

科学范例应用实现的过程中，我们用到了 VO-DAS 系统和 TOPCAT 工具。利用 VO-DAS 的 GUI 客户端从 Sloan 的 Star2 数据中查找出 A 型星数据，然后发送到 TOPCAT 用来筛选 BHB 星，并对其进行可视化。利用 VO-DAS 的命令行客户端发送请求的过程类似，但是要利用异步查询，结果数据需要手动加载到 TOPCAT 中。科学应用范例充分体现了 VO-DAS 系统可以把数据资源以透明、标准的服务方式提供给国内外天文学家。使天文

学家可以利用系统对观测的数据直接进行访问，并且通过互操作方式将结果数据作进一步地处理和分析。此系统不仅是中国虚拟天文台的主要数据访问服务环境，还将作为LAMOST 以及未来国内其它天文观测项目数据发布平台的基础。同时对推动中国虚拟天文台与各国 VO 工具的应用进行大规模和不同方式的数据访问和分析工作具有重要的现实意义。

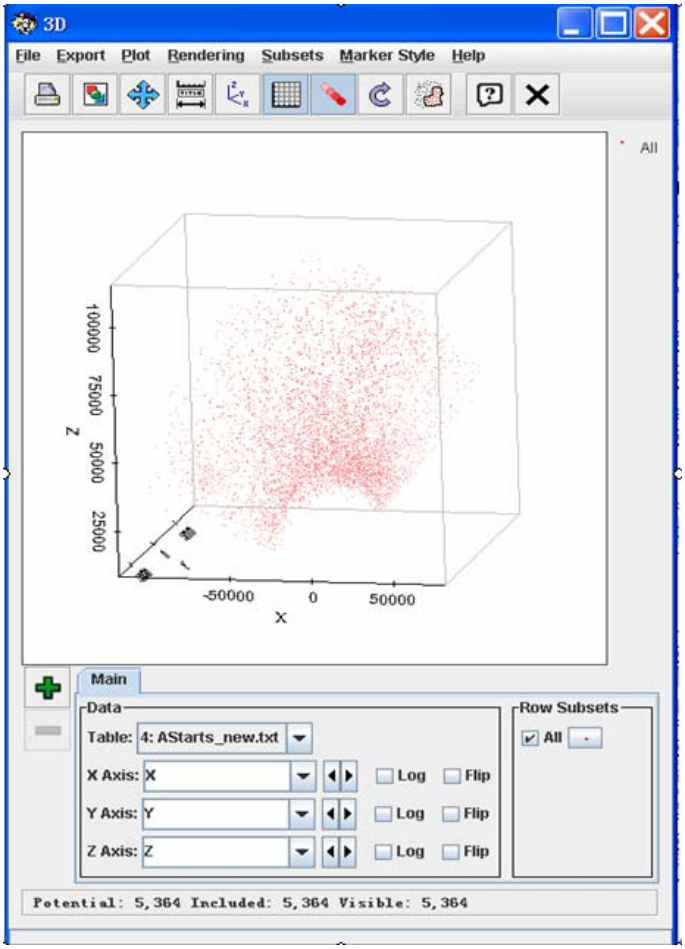

图 6.5 科学范例的 BHB 的运动轨迹

参 考 文 献：

[1]BRIAN YANNY，HEIDI JONEWBERG，STEVE KENT et al. IDENTIFICATION OF A-COLORED STARS AND STRUCTURE IN THE HALO OF THE MILKY WAY FROM SLOAN DIGITAL SKY SURVEY COMMISSIONING DATA. THE ASTROPHYSICAL JOURNAL, 2000(540):p825-841.

第七章 总结与展望

本论文基于 VO-DAS 提供的 Web 服务接口，实现了 VO-DAS 的两种客户端，即 GUI 客户端和命令行客户端。GUI 客户端以图形界面的方式访问 VO-DAS；命令行客户端以命令行的方式来完成用户对 VO-DAS 的请求。这两种客户端各具特色，分别以不同的方式完成对 VO-DAS 的访问请求，大大提高了 VO-DAS 的使用效率。另外，对 VO-DAS 的数据服务 Data Node，本文设计出了数据结点配置工具，为数据用户共享有价值的海量数据提供便捷的方式，这从另一个方面发挥了 VO-DAS 的潜力。然后概要地介绍了访问 VO-DAS 一个必不可少的环节，即 VO-DAS 系统集成，它将 VO-DAS 的各个不同的组件通过恰当的方式合理地组建在一起，为完成用户的请求而协同工作。最后，以科学范例说明了 VO-DAS 系统的应用。论文的主要内容分布如下：

第一章介绍了虚拟天文台产生的背景与国内外发展的现状， VO 数据访问和虚拟天文台数据访问服务（VO-DAS），并阐述了 VO-DAS 客户端系统研究的价值与意义。

第二章概述了虚拟天文台的相关标准和关键技术。其中包括 ADQL、VOTable、PLASTIC、网格服务和 OGSA-DAI。

第三章着重介绍了我们自主开发的 GUI 客户端系统的总体结构、系统工作的一般流程、查询结果数据处理、监控模块和获取元数据过程等，同时阐述了 GUI 客户端的实现细节。

第四章描述了我们设计的 VO-DAS 的另一种客户端：命令行客户端。分别介绍了设计的总体结构、工作一般流程、命令模块、后台调用模块以及它们的实现过程。

第五章阐述了数据结点配置工具的设计与实现，包括系统设计的总体结构、工作流程、数据库访问技术、资源部署实现、元数据配置以及实现工具与用户界面部分。同时，还描述了如何来集成 VO-DAS 的各个部分的组件，最终为用户的请求做出正确地响应。

第六章介绍了利用 VO-DAS 系统来做科学研究的应用范例：用星流的运动轨迹估算出银河系的引力势。

本论文着重介绍的 GUI 客户端、命令行客户端和数据结点配置工具将最大限度地发挥出通过 VO-DAS 系统访问异地异构海量数据进行科学研究的潜力。VO-DAS 系统作为虚拟天文台研究的课题之一，它的开发是一个逐步完善的过程。目前，图像和光谱查询功能有待完善，Web 客户端正在开发之中。于 2008 年 6 月份，VO-DAS 系统将发行 1.0 版本。后

续版本中，系统日志管理、用户帐号管理和安全管理，存储服务（MySpace）和注册服务（Registry）将会进一步改进。另外，整个系统还尚不能满足用户同时对同一结点进行多个大数据量查询任务的长期稳定性需求，受服务器硬件因素的限制，系统的负载能力也非常有限。总之，系统目前存在的这些功能与性能、软件与硬件方面的需求，将是我们日后研究工作的重点。另外，随着 VO-DAS 系统的发布与后续用户在使用过程中不断反馈给我们意见，这也将会成为 VO-DAS 系统功能完善和性能提高的一个助推器。

发表文章目录

[1] 杨阳, 刘超, 田海俊, 崔辰州, 赵永恒. VO 数据访问服务客户端系统的设计与实现. 天文研究与技术, in press, 2008.

[2] 刘超, 田海俊, 高丹, 杨阳, 路勇, 崔辰州, 赵永恒. 异地异构天文数据资源的统一访问. 天文研究与技术，in press, 2008.

致 谢

今年已经迎来了在国家天文台生活的第三个春天，此时已是大地回春，到处一片生机盎然！再回首，翘望曾经走过的历程，那一幕一幕，那点点滴滴，甚是让我感动至深！在这里，首先，我应该感谢给了我帮助最大的老师或师兄：我的导师赵永恒研究员、崔辰州副研究员以及我的师兄刘超博士。

我真诚地感谢赵永恒研究员。赵老师知识渊博、治学严谨，他待人宽厚、平易近人。赵老师虽然日常工作非常繁忙，但他仍能从百忙之中抽出时间悉心解答学生的疑难问题。他的宽博学识、坦荡的胸襟和风范无时无刻不在影响着我，这将成为我前进的主动力。

我也要由衷地感谢崔辰州博士。作为课题负责人，他总是勤奋工作、兢兢业业。我在课题组的每一步工作都是在他的指导下完成的。从开始的选题、资料收集、开题、研究和撰写论文的每一个环节，无不得到崔辰州博士的悉心指导和帮助。他性格开朗、平易近人，这几年，不管是有关工作还是在生活的话题，我们之间的沟通都是非常的愉快。

我还要由衷地感谢我的师兄刘超博士。他在我的课题研究中给予过我很多的帮助，提出了许多宝贵的意见。他治学严谨、一丝不苟。每次跟他请教或讨论一些问题时，总会让我豁然开朗。他为人随和、待人热忱。他的治学和为人将成为我学习榜样！

在天文台学习的三年学习生活中，还有很多老师和同学都给过我热情的帮助和无私的指导。

感谢张彦霞副研究员，张老师工作诚诚恳恳、治学态度非常严谨。待人热情和细心。从我收到张老师通信的第一封邮件时，我就从字里行间感受到了张老师为人师表的魅力。实际生活接触中，她也给了我许多工作和生活指导。

我要感谢 LAMOST 实验室全体的老师和同学，特别是陈英和袁辉老师、罗阿理副研究员、王丹博士、高丹、田海俊、罗宇、路勇、何勃亮、薛元、刘建、王凤飞、宋轶含、严太生、孙士卫、吴悦。

感谢杜红荣老师和艾华老师。杜老师和蔼可亲，感谢她热情细致的帮助。

感谢兴隆党支部书记李金增老师的悉心教导和关心。

真诚地感谢我的父母亲默默地支持，感谢大伯和四伯一直以来给予我的鼓励和关心。感谢姑姑对我无微不至的关怀。感谢我的男友李福东这三年里对我的关心和理解。

感谢所有关心、照顾和帮助过我的亲人、朋友、老师和同学。

最后，我要将此论文献给养我育我、曾无时无刻不在惦记着我的祖母。她虽已逝 6 年，但她的为人处世之道以及对我至深至沉的疼爱，将会一直烙印在我心间，永不褪色。