分类号：TP393 密级：公开

研 究 生 学 位 论 文

论文题目(中文) 中国虚拟天文台资源管理系统的设计

论文题目(外文) China-VO Resoure Management System Design

研 究 生 姓 名 李 长 华

学 科 、 专 业 计算机软件与理论

研 究 方 向

学 位 级 别 硕 士

导师姓名、职称 **李 廉** 教授

论 文 工 作
起 止 年 月 2004 年 3 月至 2005 年 4 月

论文提交日期 2005 年 4 月

论文答辩日期 2005 年 5 月

学位授予日期 2005 年 6 月

校址：甘肃省兰州市

中国虚拟天文台资源管理系统的设计

论文摘要（中文）

摘要：计算机与互联网技术的飞速发展，网格、语义网等新 IT 技术的涌现，使得海量、分布式、多波段天文数据的无缝融合和处理成为可能。在这样的背景下，旨在将世界范围内主要天文研究资源无缝透明地整合在一起的虚拟天文台（VO）设想应运而生,并很快得到世界各国天文界的重视。以中国科学院国家天文台（NAOC）为代表的中国天文界提出了中国虚拟天文台（China-VO）计划。

以开放网格服务架构（OGSA）为代表的网格技术旨在消除互联网上的“信息孤岛”，实现数据资源、计算资源、存储资源等各种网络资源的全面共享。OGSA 为 China-VO 的建设提供了重要的网格平台，其架构是 China-VO 体系结构的基础。整体上看，China-VO 由构造层、资源层、汇集层和用户层构成。China-VO 采用面向服务的设计理念，整个服务体系包括应用服务提供者、数据服务提供者和 VO 注册三种角色。

由于 China-VO 采用了 OGSA 的体系结构，是以目前流行的 GT3 作为底层实现来架构我们的实现平台，为此，在系统中，一切都是服务，资源管理本身是由一系列的服务组成，如现在的资源注册服务，资源监控服务，资源与服务查询服务等都是资源管理系统的组成部分。

互操作性是 VO 对资源与服务的基本要求，是 VO 功能开发的基础。在资源管理系统中，VO 定义了一套自己的资源元数据标准，及许多相关的其它技术标准。从而使得中国虚拟天文台能与世界各国的天文台进行无缝透明的访问。

资源注册与发现是资源管理的核心环节。在 VO 中，我们实现了集中与分布的层次注册体系，及资源的局部管理与全局发现机制。提供了友好的界面，为资源提供者提供资源，而在对资源的管理上，实现了资源向服务的转化。在服务的分配上，我们实现了基于负载平衡的服务分配机制。

实时的了解 VO 系统中各节点的资源使用状态，以及时更新系统的服务分配策略，实现整个系统的负载平衡，是 VO 系统平稳运行的关键。为此，开发了一个节点资源监控服务，以实时反应系统的资源状态信息，为基于负载平衡的服务分配提供了条件。

关键字：中国虚拟天文台，网格，OGSA，资源服务化，资源注册与发现，资源监控

论文摘要（英文）

Abstract

With the emergency of many new techniques, such as initiative optic, self-adaptive optic, optical interference,the capability of data collecting in astronomy is enhanced greatly. On the other hand, the fast development and application of High Performance Computing and Internet technology,Grid and semanteme network technology,etc., makes it possible to locate and access large-scale astronomical datasets all around the planet. Under this background, the VO imagine arise.

Grid technology, with its preventative Open Grid Service Architecture (OGSA), aims to remove the "isolated islands" of information on the Internet, and to enable dynamic sharing of all kinds of network resources, including data resource, computing resource, storage resource and so on. OGSA provides an important technical platform for the China-VO. The system architecture of the China-VO is strongly based on that. The China-VO will adopt service-oriented conception. The whole service system functions as three roles, i.e. application service provider, data service provider and VO registry.

In VO environment, interoperability is the basic requirement for resources and services, and the base for developing high-level applications. By defining and adopting feasible and scalable VO data model, data exchange standard and metadata standard, heterogeneity transparency of resources access can be reached.

Resource registration and discovery is a core of resource management. In VO environment, we have realized a level registry system with local management and the overall discovery mechanism. Moreover, we developed the friendly interface for resource provider. Meanwhile,we have realized the transformation from resource to grid service and balanced-load distribute mechanism successfully.

It is the key to VO environment that real-time understanding the status of nodes in order to update allocation tactics and make the whole system balanced.So, I have developed a monitor service to reflect the systematic resource status in real time.

Key words: China-VO; Grid; OGSA; resource registry and resources discovery; resource monitor and allocation.

目 录

第一章 中国虚拟天文台概述

1.1 虚拟天文台概念

四百年前伽利略首次把望远镜指向太空，使人类摆脱了仅能用肉眼直接观测太空的历史，为从哥白尼开始的天文学革命提供了大量的科学证据[1]。到一百五十年前，由于照相技术和光谱技术在天文观测中的应用，用人眼作为唯一的天文探测器的时代结束，诞生了天文学的新分支——天体物理学，并发展成为现代天文学的主流。

五十多年前，在第二次世界大战中得到蓬勃发展的无线电技术使得人类的视野跃出了可见光的波段，发展成为射电天文学。之后不久随着宇航时代的到来，空间天文学诞生，天文观测不再局限于地面。人类对宇宙的观测范围扩展到了γ射线、X射线、紫外和红外波段。天文学开始进入全波段天文学时代。

从20世纪90年代开始，天文学又经历着革命性的变化。这一变化是由前所未有的技术进步所推动的，即望远镜的设计和制造技术、大尺寸探测器阵列的设计和制造技术、高性能计算技术和互联网技术。

随着众多先进的地面与空间天文设备的投入使用，特别是大规模CCD探测器的使用，使得观测数据量急速增长。例如目前哈勃空间望远镜（HST）每天大约产生5GB的数据；我国正在建造的大天区面积多目标光纤光谱望远镜（LAMOST）也将每天产生3GB的数据[2]；而美国计划建造的“大口径综合巡天望远镜（LSST）”，又称为“暗物质望远镜”，每天的观测数据将达到18TB的量级[3]!

除了数据量的快速增长外，天文观测的方式也发生着变化。当前天文学的观测方式主要有两种类型：定点观测和巡天观测。

定点观测就是为了完成特定的研究课题而利用观测设备对预先选定的天体进行观测，数量一般不多，很少能超过一百个。

巡天观测是对整个天区或者很大面积的天区按照预先设定的观测计划进行观测。

如果说利用γ射线巡天、X射线巡天、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、统计分析工具和数据挖掘工具，这就相当于拥有了虚拟的各种探测设备。这样，由虚拟的数字天空、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（Virtual Observatory， VO）。由此可见，虚拟天文台是在互联网时代里天文学发展的必然产物。

简言之，虚拟天文台就是利用先进的信息技术，将世界各地的天文资源无缝透明的联结起来的数据密集型的天文研究环境。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[4]。虚拟天文台将促进我们对许多决定宇宙演化的天体物理过程的理解。它会用更经济的投资产生新的和更好的科学。虚拟天文台将作为一个协调性的和操作性的机构来促进新型的工具、协议和合作方面的发展，以充分实现现代天文数据库的科学潜能，从而将成为“天文学发现”的推进器。

1.2 虚拟天文台的科学目标

目前，天文学家确定的虚拟天文台的主要科学目标是：

1）多观测参数高维空间的探索：将各个巡天数据统一到虚拟天文台中，将会有更广泛而复杂的应用。这些数据能提供全天在十多个不同波段上的信息，在多维空间里展示整个天空的真实面貌。

2）稀有天体与新型天体的发现：目前通过巡天来寻找稀有天体（如高红移类星体、褐矮星等）的项目正在蓬勃发展。假如某种有趣的天体或现象出现的概率是百万分之一或一亿分之一，那么就需要几百万或几亿个样本才有可能发现。这样，在海量数据中进行彻底的宇宙探索来寻找稀有的未知类型天体便具有更加诱人的前景。因此，虚拟天文台将会利用其独有的数据资源和计算资源促进新的天文发现。

3）新兴的科学领域：虚拟天文台对任何要求融合各类数据来研究天文现象的课题都具有重要的影响。虚拟天文台的出现会大大促进多波段天文学的发展，推动各种各样令人兴奋的科学探索，帮助统计天文学的兴起，从而使天文学研究在数量和质量上得到充分地提高。

4）数据挖掘技术：数据挖掘技术在虚拟天文台中的应用，将使任何地方的天文学

家在不依赖于大望远镜的情况下就可以做出一流的工作，而这种研究方式完全不同于传统的天文学研究。运用数据挖掘技术可以有效地解决天文学中的“数据雪崩”问题，这对天文学发展是至关重要的。

虚拟天文台的发展壮大和普及将会使得实测天文的研究模式再次发生重大变化，从巡天研究模式升级为 VO 研究模式。虚拟天文台将提供一种集成的天文研究环境，以实现天文观测，数据收集，数据处理的自动化，以大大简化天文学家的工作，提高工作效率，我们可以用”三替”来形象的说明虚拟天文台的实现目标，一替天文学家找数据，二替天文学家处理数据，三替天文学家可视化数据。同时，虚拟天文台可成为天文教育与科学普及的良好平台。

1.3 虚拟天文台的研究现状

美国首先提出建立虚拟天文台的计划后，欧洲、英国、德国、日本、加拿大、俄罗斯等国家也相继提出了类似的计划。虽然这些计划来自不同的国家，有着不同的天文和技术背景，但是他们之间有许多非常重要的共同点：每个项目都在寻求数据密集型天文研究的出路，都在力图挖掘现有以及未来海量天文数据的潜力。

每个项目都制定了有自己特色的科学目标，都有自己的技术优势和兴趣。从国际天文界的眼光来看，这些项目的共同目标就是建立一个国际虚拟天文台（IVO）。如果把这些项目联合起来，共同迎接虚拟天文台所面临的科学与技术挑战，这对 IVO 的建立是很有好处的。

因为IVO必须是一个完整的，不同部分能进行互操作的系统。为了实现不同部分之间的互操作性，就必须对许多问题在国际范围内达成一致和认可，这需要一种机制来实现。正是出于此目的，AstroGrid、AVO和NVO利用 2002 年 6 月在德国Garching举行“迈向国际虚拟天文台（Towards an International Virtual Observatory）”国际会议[5]的机会提出了成立国际虚拟天文台联盟（IVOA）[6]的倡议，他们的倡议得到了与会代表的支持。IVOA当即表示成立。 目前，IVOA在虚拟天文台的各个领域都成立了相应的工作组，以负责某个方面标准的制定。在IVOA的协调与促进下，虚拟天文台正在获得飞速的发展。

第二章 网格及主要相关技术

2.1 网格的概念

网格计算是伴随着互联网技术而迅速发展起来的，专门针对复杂科学计算的新型计算模式。这种计算模式是利用互联网把分散在不同地理位置的电脑组织成一个“虚拟的超级计算机”，其中每一台参与计算的计算机就是一个“节点”，而整个计算是由成千上万个“节点”组成的“一张网格”， 所以这种计算方式叫网格计算。这样组织起来的“虚拟的超级计算机”有两个优势，一个是数据处理能力超强；另一个是能充分利用网上的闲置处理能力。简单地讲，网格是把整个网络整合成一台巨大的超级计算机，实现计算资源、存储资源、数据资源、信息资源、知识资源、专家资源的全面共享。

清华大学李三立院士将网格与信息高速公路作了比较，他说：“将先进计算基础设施（网格）与信息高速公路相比较，可以说，信息高速公路是信息传输和获取的信息基础设施；而先进计算基础设施则是信息处理的信息基础设施。虽然，国内外都有不断把信息高速公路扩充频带宽度、改进路由器性能的计划；但是，国外科学家认为：真正的下一代信息基础设施是先进计算基础设施，它将使以计算机为主体的信息处理发生根本性的变化。”[7]

中科院计算所李国杰院士认为：“网格不同于国外正在搞的Internet 2 或下一代Internet (NGI)，网格可以称作是第三代 Internet，其主要特点是不仅仅包括计算机和网页，而且包括各种信息资源，例如数据库、软件以及各种信息获取设备等，它们都连接成一个整体，整个网络如同一台巨大无比的计算机，向每个用户提供一体化的服务。”[8]

网格技术的产生、发展必须具备以下三个基本条件：计算资源的广域分布、网络技术（特别是互联网）以及不断增长的对资源共享的需求。在计算机技术发展的早期阶段，只有很少数量的大型计算机，它们通常被安装在相互独立的计算中心内，多个计算机用户通过使用终端来共享一台大型机的资源，但却不能同时共享多台大型机的计算资源。

随着网络技术的发展，多台大型计算机可以在局域网内互连，用户通过网络便可以同时使用多台计算机的资源。而互联网的飞速发展与普及使得网格计算技术的产生成为可能。下图显示了计算资源共享的发展过程[9]。

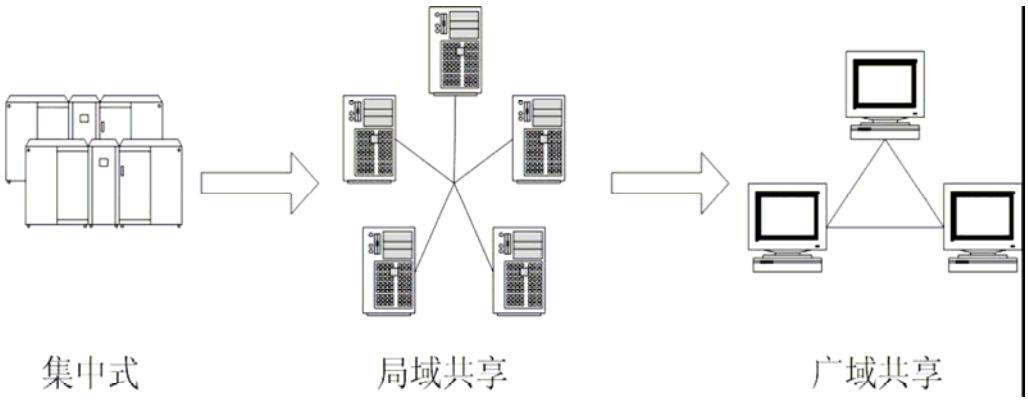


图2-1：资源共享发展过程

从上世纪60年代末开始研究计算机分组交换技术到今天，互联网已经走过两代历程。第一代是20世纪70～80年代，主要的成就是把分布在世界各地的计算机用TCP/IP协议连接起来，主要的应用是电子邮件。第二代是20世纪90年代，主要成就是把成千上万网站上的网页连接起来，主要的应用是Web信息浏览以及电子商务等信息服务。目前正处于从第二代互联网向第三代互联网过渡的转型期。第三代互联网也就是网格（Information Service Grid），其主要特点是不仅仅包括计算机和网页、而且包括各种信息资源，例如数据库、软件以及各种信息获取设备等，它们都连接成一个整体，整个网络如同一台巨大无比的计算机，向每个用户提供一体化的服务。简单地讲，传统互联网实现了计算机硬件的连通，Web实现了网页的连通，而网格试图实现互联网上所有资源的全面连通。网格追求的最终目标是能够做到服务点播和一步到位的服务，把整个互联网整合成一台巨大的超级计算机，实现计算资源、存储资源、数据资源、信息资源、知识资源、专家资源的全面共享。

网格技术要解决的信息共享不是一般的文件交换与信息浏览，而是要把所有个人与单位连接成一个虚拟组织（Virtual Organization），实现在动态变化环境中有灵活控制的协作式信息资源共享。网格与Web最大的区别是一体化，即用户看到的不是数不清的门类繁多的网站，而是单一的入口和单一系统映像。

现有的Web信息服务器就好像Internet世界上一个个孤立的小岛。虽然这些“小岛”之间暂时还有充足的带宽资源可用，但大量的信息还是被“锁”在各个小岛的中央数据库里，各 “孤岛”之间并不能按照用户的指令进行有意义的交流。解决这一问题的最佳途径是建立跨越Web的信息分布和集成应用程序逻辑——网格。

网格的兴起将改变传统的Client/Server（C/S）和Client/Cluster结构，形成新的Pervasive/Grid体系结构。客户端是各种各样的上网设备，而连在网上的各种服务器将组成单一的逻辑上的网格。

网格的本质特征表现在应用上。网格的服务包括文件消息、计算、信息内容、事务处理和知识服务等，因此网格可大致分为数据网格、计算网格、信息网格与知识网格等。

全球网格研究的领军人物、美国Argonne国家实验室的资深科学家、美国Globus 项目的领导人Ian Foster 曾在1998 年出版的《The Grid: Blueprint for a New Computing Infrastructure》[10]一书中这样描述网格：“网格是构筑在互联网上的一组新兴技术，它将高速互联网、高性能计算机、大型数据库、传感器、远程设备等融为一体，为科技人员和普通老百姓提供更多的资源、功能和交互性。互联网主要为人们提供电子邮件、网页浏览等通信功能，而网格功能则更多更强，让人们透明地使用计算、存储等其他资源。”

2000 年，Ian Foster 在《The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration》这篇论文中把网格进一步描述为“在动态变化的多个虚拟机构间共享资源和协同解决问题。”但是，人们仍然就什么是网格而争论不休。2002 年7 月，Ian Foster 在《What is the Grid? A Three Point Checklist》[11]一文中，限定网格必须同时满足三个要求：

1）在非集中控制的环境中协同使用资源；

2）使用标准的、开放的和通用的协议和接口；

3）提供非平凡的服务。

总而言之，网格不外乎是要利用互联网把分散在不同地理位置的电脑组织成一台“虚拟的超级计算机”， 实现计算资源、存储资源、数据资源、信息资源、软件资源、存储资源、通信资源、知识资源、专家资源等的全面共享。其中每一台参与的计算机就

是一个节点，就像摆放在围棋棋盘上的棋子一样，而棋盘上纵横交错的线条对应于现实世界的网络，所以整个系统就叫做“网格”了。在网格上做计算，就像下围棋一样，不是单个棋子完成的，而是所有棋子互相配合形成合力完成的。传统互联网实现了计算机硬件的连通，Web 实现了网页的连通，而网格试图实现互联网上所有资源的全面连通。下面再看看与网格紧密相关的几个概念：

1).资源:在网格中所指的资源是个非常广泛的概念，它包括从程序，文件和数据到计算机，传感器，网络等所有软硬件基础设施，网格资源是随时间动态变化的，资源本身是异构的和多样的。

2).共享:共享是网格的主要目的之一。但在网格中，共享不只是交换文件，而是强调对计算机软件，数据以及其它所有资源的直接访问，而且，这种共享还必须是高度受控制的，是有条件的。虚拟组织就是基于这样的一些共享规则，由一些人或团体形成的集合体。网格中的共享又是非常灵活的，主要有三种形式的共享，即客户端与服务器（c/s）的共享，端到端(p2p)的共享以及代理（proxy）的共享。另外，网格中的共享是一种随时间变化的动态的共享，这也是网格中的资源具有动态性的缘故。

3).虚拟组织：在网格系统中，资源的共享是有条件的，而且，共享关系也是动态变化的，因此， 为了协调好动态变化的共享资源，就提出了虚拟组织的概念，所谓虚拟组织，就是指遵守资源共享规则的一组个体或机构，它是组织网格资源的单位。虚拟组织与实际组织的关系如图 2-2 所示：

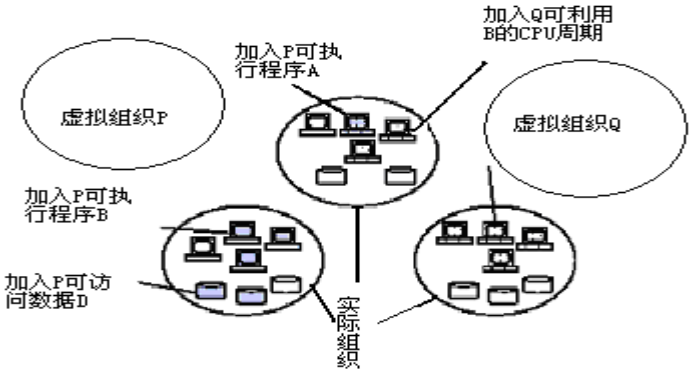


图 2-2：一个实际组织通过共享所有实际组织控制的部分或全部资源而参与到一个或多个虚拟组织中。图中，我们画出了三个实际组织（圆圈）和两个虚拟组织：P 和 Q.

4).网格服务：网格服务是一种特殊的Web Service，该服务提供了一组接口，这些接口的定义明确并且遵守特定的惯例，解决服务发现，动态服务创建，生命周期管理，通知等问题。简而言之，网格服务=接口/行为+服务数据，而在OGSA中，一切都看作是网格服务。一个高级的网格服务又由多个子服务组成。 在OGSA中，网格服务分为两种类型，一为基本服务，一为临时服务。基本服务是永久性的服务，而临时服务是由基本服务派生出来的，它们都具有一定的生命周期。

2.2 网格的体系结构

2.2.1 层次体系结构(五层沙漏模型)[12]

以“协议”为中心，同时强调服务与 API（Application Programming Interface）和 SDK（Software Development Kits）的重要性。五层沙漏结构根据各组成部分与共享资源的距离，将功能分散在五个不同的层次，从底层到上面依次是：构造层、连接层、资源层、汇聚层和应用层，越向下层就越接近于物理的共享资源；而且各部分的协议数量是不同的，对于其最核心的部分，要能够实现上层协议向核心协议的映射，同时实现核心协议向下层其他各种协议的映射，因此核心协议形成了协议层次结构中的瓶颈，在五层结构中，资源层和连接层共同组成这一核心。

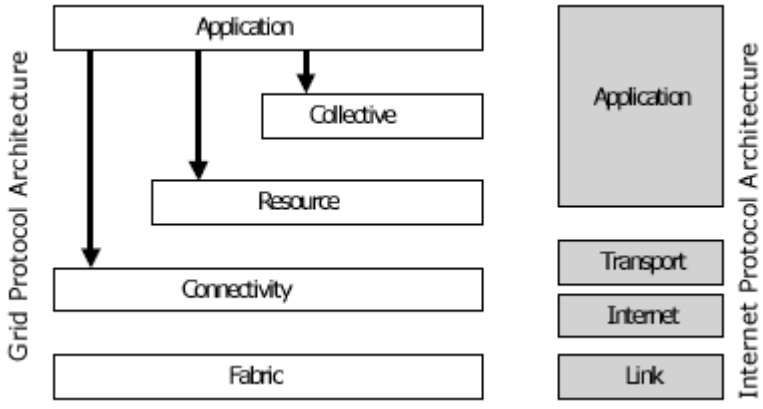


图 2-4：网格层次体系结构与Internet协议间的关系[12]

构造层：本地控制系统的接口，提供了共享的资源。比如计算资源、存储资源、数

据、网络资源和传感器。构造层部件执行的是本地的、与资源相关的操作。

连接层：安全方便的通信，定义了网格环境下的网络交易所需的核心通讯和认证协议。通讯协议实现了构造层资源间的数据交换。认证协议建立在通讯服务基础上提供用于用户和资源识别的加密的安全机制。

资源层：共享单一资源，为个体资源的共享操作定义了安全对话、初始化、监测、控制、记帐、付帐等协议。资源层协议针对的是单个的资源，不考虑分布式资源集合广域上的状态和管理等事务。

汇集层：协调多样资源。资源层关注的是单个的资源，汇集层则关注的是资源的总体。建立在资源层和连接层构成的协议瓶颈上，汇集层实现了广泛的共享操作。汇集层的功能是屏蔽网格资源层中各种资源的分布、异构特性，向网格应用层提供透明、一致的使用接口。汇集层也称为网格操作系统，它同时需要提供用户编程接口和相应的环境，以支持网格应用的开发。

应用层是用户需求的具体体现。在网格操作系统的支持下，网格用户可以使用其提供的工具或环境开发各种应用系统。

2.2.2 开放网格服务体系架构(OGSA)

2.3.2.1 面向服务的思想(SOA)

网格体系结构给出了网格的基本组成和功能，描述了网格各组成部分的关系以及它们集成的方式或方法，刻画了支持网格有效运转的方式。OGSA 是在综合了 WEB 服务和网格协议两方面优势的基础上提出的，对 WEB 服务进行了与网格技术兼容性的扩展。

OGSA 是一个面向服务的体系结构，在网格中：

- 一个服务是一个基于网络的能提供某种功能的实体。
- 一个网格服务是一个遵循一套与其接口定义和行为相关的规范的由WSDL进行描述的服务。每个网格服务都是一个WEB服务，但是对其 进行了网格化扩展。

OGSA与WEB服务和网格协议的关系如图2-5所示[13]。基于Grid和WEB服务的思想和技术，OGSA体系定义了一个统一的对外服务语义，即Grid服务；定义了标准的瞬时

Grid服务实例的创建、命名和发现机制；为服务实例提供了地域透明性和多协议绑定；并提供了与本地平台系统的集成机制；以WSDL接口和相关约束的格式定义了创建和组织高级分布式系统所需的机制，其中包括生命期管理、变动管理、通告等。

在OGSA中，一切都以Grid服务的形式体现。面向服务的模型有许多优点：环境中所有部件都进行了虚拟化，通过层层抽象以统一的方式对待这些服务。所有服务都要提供一系列核心永久接口，在此基础上，可以构造层次式的高级服务。核心网格服务与其他网格元素的关系如图2.5所示。虚拟化还能实现多个逻辑资源实例向同一物理资源的映射，在无需考虑底层资源的实现和管理情况下构造上层服务。Grid服务的虚拟化还提供了共同服务语义行为向本地平台相应机制的无缝映射，从而实现平台无关性。

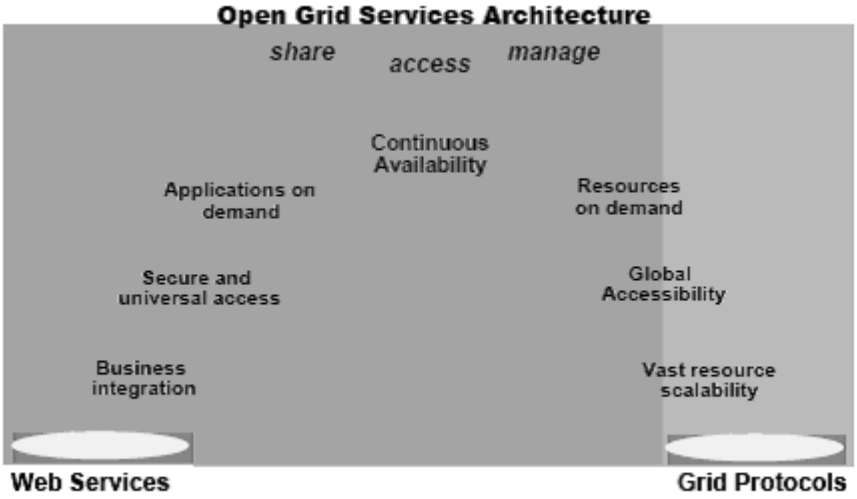


图2-5　OGSA与WEB服务和网格协议的关系

2.3.2.2　OGSA 平台

OGSA平台[13]旨在为各种网格系统所共同面临的基本问题定义标准的解决方案和机制。这些基本的问题包括网格服务间的通信、身份确立、授权对话、服务发现、错误通告、服务集管理等。

如图2-6所示，OGSA平台包括三个基本元素：开放网格服务基础设施（OGSI）、OGSA平台接口和OGSA平台模型。

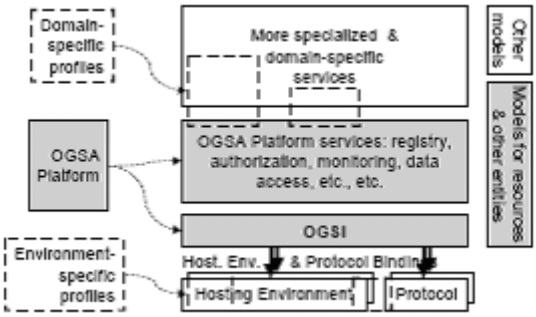


图2-6 OGSA平台及相关环境

• 建立在网格和WEB服务基础上，OGSI定义了一套机制来实现网格服务的产生、管理以及网格服务间的信息交换。一个网格服务是一个实现了一系列规范的WEB服务。这些规范，包括接口和行为，定义了一个客户如何与一个网格服务进行交互。这些规范与其他一些与网格服务的产生和发现相关的OGSI机制一起实现了对分布式、长生命期的状态进行可控的、带错误反馈的、安全的管理。这对分布式的应用是一个基本的需求。

• OGSA平台接口，建立在OGSI之上，定义了一系列OGSI不直接支持的各种功能的接口和相关行为，比如服务发现、数据访问、数据融合、消息、监控等。

• OGSA平台模型，通过为公用资源和服务类型定义模型来实现上述接口规范。

在OGSA平台之上OGSA还定义了一些高层系统服务，主要包括：

• 分布式数据管理服务，实现了对分布式异构数据资源的统一透明访问；

• 工作流服务，实现多个分布式网格资源上多个应用作业的协同操作；

• 审计服务，记录资源的使用并对这些记录信息安全的保存和分析，用于发现异常和入侵检测等；

• 指导和监测服务，提供分布式环境中“传感器”的发现，检测信息的收集和分析，探测到非常状况时产生警告；

• 分布式计算的问题判定服务，实现备份、跟踪、记录等功能；

• 安全协议映射服务，实现分布式安全协议透明的映射为本地平台系统的安全服务，以满足本地安全认证和访问控制的需要。

2.3 OGSI与WSRF

2.3.1 Globus Tools

目前，在国际上影响最大的网格开发项目是Globus[14]。这个项目的成员来自美国 Argonne 国家实验室数学与计算机分部、南加州大学信息科学学院和芝加哥大学分布式系统实验室等单位，并与美国国家计算科学联盟、NASA 信息能源网格（IPG） 项目、美国国家先进计算基础设施同盟（NPACI）等建立了伙伴关系。Globus项目的主要工作是开发、解决建立网格所需要的基本技术。

目前，他们推出的工具集Globus Toolkit （GT）[15]版本有2.x,3.x及将推出4.0。其中2.x是一套旨在实现网格资源管理的中间件，侧重于实现计算网格的功能。3.x以后是OGSA的一种实现，侧重于实现服务网格。

迄今为止，Globus项目开发的Globus Toolkit已经成为事实上的网格标准。一些重要的公司，包括IBM、Microsoft、Compaq、Cray、SGI、Sun、Fujitsu、Hitachi、NEC 等已经公开宣布支持Globus Toolkit。大多数网格项目也都是基于Globus Tookit 所提供的协议及服务建设的，例如美国的物理网格GriPhyN、欧洲的数据网格DataGrid、荷兰的集群计算机网格DAS-2、美国能源部的科学网格和DISCOM 网格、美国学术界的TeraGrid等等。

2.3.2 OGSI

OGSI(开放网格服务基础设施)[16]，定义了网格服务实体的创建，管理，及服务之间交换信息的机制，它主要由以下规范组成：

1）. 网格服务的组成：

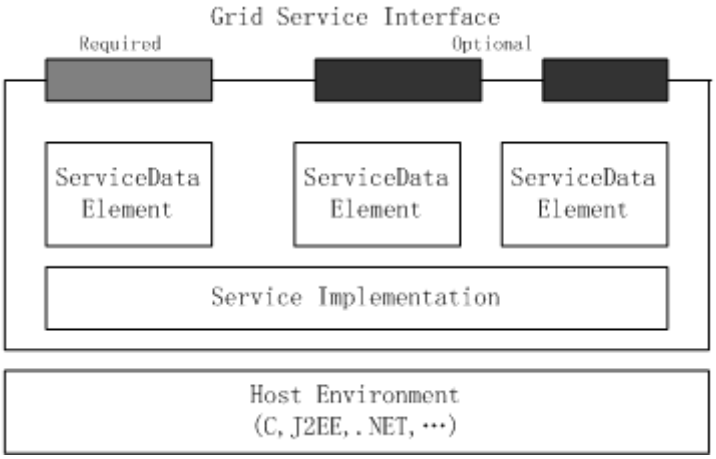


图2-7：网格服务组成层次

2） 网格服务的命名与绑定:

•每一个服务实例有一个全局唯一的、不变的名字 GSH（Grid Service Handle）,在表现形式上，它通常是一个全局唯一的 URL，没有携带网络协议和地址等信息，它只是表示一个网格服务实例的调用地址。

•网格服务引用 GSR（Grid Service Reference）描述了和实例交互所需的实例特殊信息，其中包括协议绑定信息，网络地址等，客户程序通过它来了解服务实例的详细情况，并与服务实例进行交互，在实例生命期内，GSR 会发生变化，一个 GSH 可能对应多个 GSR，但一个时间是一一对应，名字和实现的分离方便了服务的升级和演变。

定义与描述 grid Service 的 GWSDL 与 WSDL 文档

•The Mapper interface 接口将 GSH 转换为 GSR.

3） Factory 服务与服务实例

Factory 服务被称为是创建服务的服务，而服务实例则是服务的一次使用，类似于面向对象里的类与实例的关系，任何实现了 Factory 接口的服务都可以通过创建服务实例来执行服务的操作。而服务也只有通过创建服务实例来为用户使用。

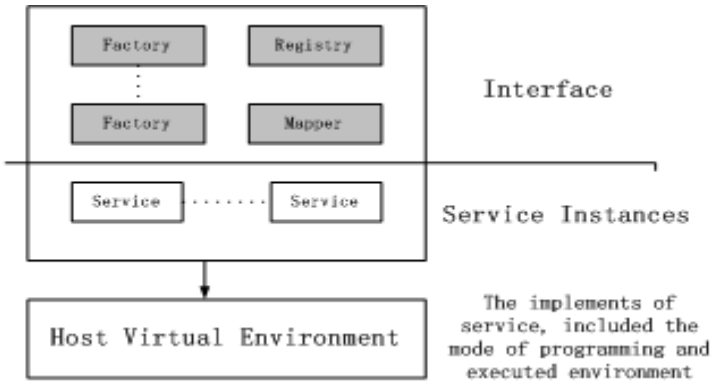


图 2-8：服务与服务实例

4）网格服务实例的创建过程

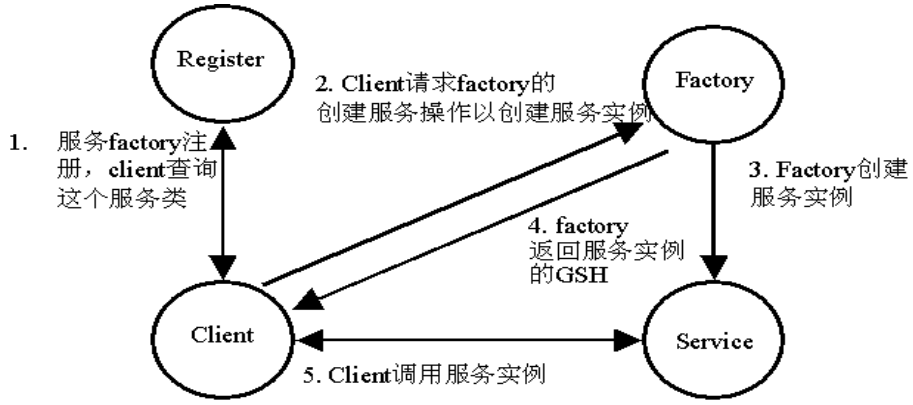


图 2-9：服务实例的创建过程

5）服务数据，服务数据提供者

用来反映服务或其它资源状态的元数据。有些服务数据是由服务动态产生的，也有些服务数据是由专门的服务数据提供者产生的。所谓服务数据提供者就是指为产生某些服务数据而专门编写的程序。通常都是为了获取一些硬件信息的服务数据。为了便于服务的管理，GT3 中的任何网格服务都已经有了预定义的服务数据，如 entry,serviceDataName,interface 等，以表达服务的一些最基本信息。同时，为了方便查询，GT3 中又采取了统一的管理方式，如下图所示：

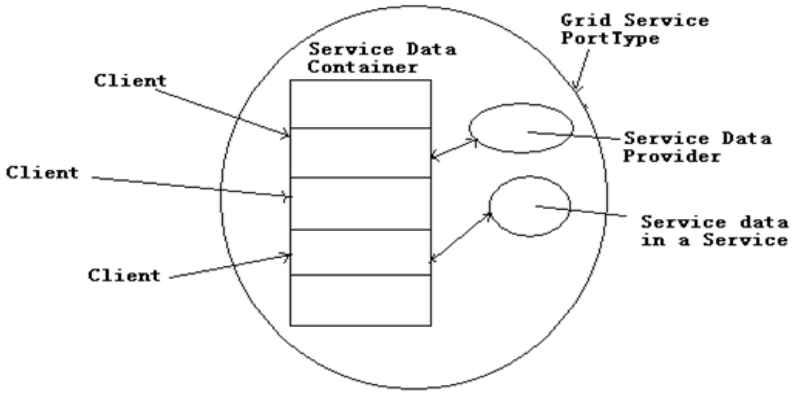


图 2-10：服务数据的管理机制

2.3.3 WSRF

WS(Web Service)现在应用已非常广泛，但WS不能提供给它们的用户维护与管理状态的能力，例如，WS无法保存与用户交互的结果，而这却是一个很有必要的需求，如果WS能够保存这些状态，它就能以一种标准的方式实现有状态的资源的交互，及实现互操作性，同时，WS并不是绝对需要状态，因此也需要实现服务与状态资源的分离。而OGSI规范里的网格服务，虽然提供了记忆状态的能力，但与WS具有很多不一致的地方，使用不容易等很多不足，为此，为了满足这种需求，OASIS推出了WSRF规范标准。

WSRF是一套规范的集合，它主要由以下几个规范组成[20]:

- WS-ResourceLifetime 定义了 WS-Resource 的资源破坏机制，其中包括允许请求方立即或使用基于时间的安排好的资源终止机制来破坏资源。
- WS-ResourceProperties 定义了 WS-Resource 的类型定义如何与 Web 服务的接口描述相关联，以及检索、更改或删除 WS-Resource 属性的消息交换过程。
- WS-Notification 通过基于主题的发布/订阅模式定义事件订阅和通知机制。
- WS-RenewableReferences 定义了端点变为无效时，对需要检索端点引用更新版本的策略信息的常规 WS-Addressing 端点引用。
- WS-ServiceGroup 定义了通过异质引用集合访问 Web 服务的接口。

- WS-BaseFaults 为在 Web 服务消息交换过程中返回的错误信息定义了基本的 XML 错误类型。

通过以上规范，WSRF 又重新定义了网格服务的组成， 这种新的网格服务与 WS 完全兼容，所不同的是，这种网格服务能够保存服务的状态，状态通过一种叫资源的形式来保持，当不需要保持状态时，网格服务也就成了 WS。

2.4 中国虚拟天文台与网格的完美结合

虚拟天文台的最终发展目标就是实现全球天文数据的高级共享，同时提供一整套的智能化工具。TB量级甚至PB量级大型天文数据产出项目的不断涌现，对数据存储、数据管理、数据传输、数据检索等技术提出了更高的要求。在如此海量分布式数据的基础上进行科学研究，就必须有全新的数据共享、数据互操作、作业调度、数据可视化、数据统计分析、数据挖掘、数据安全管理等工具的支持。

虚拟天文台的这些需求正是网格技术要实现的目标。网格技术将实现把整个互联网整合成一台巨大的超级计算机，实现计算资源、存储资源、数据资源、信息资源、知识资源、专家资源的全面共享，为用户提供一步到位的服务。因此，虚拟天文台的建立和实现需要网格技术的支持，虚拟天文台把网格技术作为自己的技术基础将是可行而明智的选择。

另一方面，网格还处在初步研究与发展的阶段，还远没有得到广泛的认可与实际的应用，因此，它也需要找到一个适合的领域来证明及发展，虚拟天文台正好为网格技术的发展提供了最好的实验场所。天文数据有着其他学科数据无法比拟的特点：开放性，海量数据 ，良好归档 ，格式多样 ，全波段数据。

虚拟天文台要实现对这样数据的融合。这样的发展目标为网格技术提供了独一无二的试验场。从网格基础设施的构建，到网格操作系统的开发，最后到网格天文应用工具的实现，虚拟天文台为网格技术提供了一整套的应用需求。

第三章 中国虚拟天文台的体系结构设计

体系结构也称为体系架构，为整个系统提供了一个结构、行为和属性的高级抽象，由对构成系统的元素的描述、元素间的相互作用、元素集成的模式以及这些模式的约束组成。体系结构不仅指定了系统的组织结构和拓扑结构，并且显示了系统需求和构成系统的元素之间的相互关系，提供了一些设计决策的基本原理。它是实现系统其它部分功能设计的前提与基础。

3.1 中国虚拟天文台的系统结构

根据中国虚拟天文台的科学目标，以及采取的网格技术，特别是采用了 SOA 的体系架构，因此，从基本的功能模块角度来分析，China-VO 系统结构组成将如图所示。

整个体系结构分为四层，从下到上依次是构造层、资源层、汇集层和用户层[21]。

构造层是整个虚拟天文台系统的资源基础，其中包括各种数据资源，计算资源，网络资源，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。

资源层将以开放网格服务架构（简称OGSA）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统一的计算访问以及网格系统管理等功能。前面提到的数据访问层的功能将在这部分实现。这里，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。

汇集层包括最能体现天文特色的各种VO服务，比如数据处理、数据挖掘、统计分析、可视化等应用服务。当OGSA体系架构及其实现工具成熟以后，这些服务的开发和发布将是VO建设的重点。

用户层，包括VO客户端服务和VO门户，是整个体系的最高层，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、偏好设置等。

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。

图 3-1：中国虚拟天文台的体系结构

3.2 中国虚拟天文台的服务模型

体系结构可以从多个方面来设计。现在，我们换一个角度来审视整个VO系统。其实在上节中我们已经提到了许多VO需要实现的功能，比如数据访问、数据处理、数据互操作、资源发现等。为了进一步明确VO的核心服务以及对网格平台的要求，下面我们按照OGSA面向服务的设计理念来重新分析VO系统。

OGSA是一个面向服务的体系。在整个网格环境中所有组件都是以服务的形式来体现的。服务可以是不依赖于其他服务而独立存在的原子服务，也可以是建立在其他服务之上的复合服务。不管是原子服务还是复合服务，它们最基本的共同点就是在网格环境中可以提供某种功能。概括起来，VO系统中需要用到的服务或功能主要有以下一些：数据访问 ， 文件访问 ， 数据整合，数据迁移， 数据挖掘 ， 分类， 聚类，计算服务 ，可视化服务 ，数据转换 ，注册与发现 ，元数据服务等，结构如图3-2所示：

3.3 中国虚拟天文台中的资源管理系统简述

从体系结构图，不管是层次图，还是服务模型图，资源管理是整个系统的核心环节，是整个系统的心脏。它包括资源的注册，资源的发现，资源的调度，资源的服务化，资

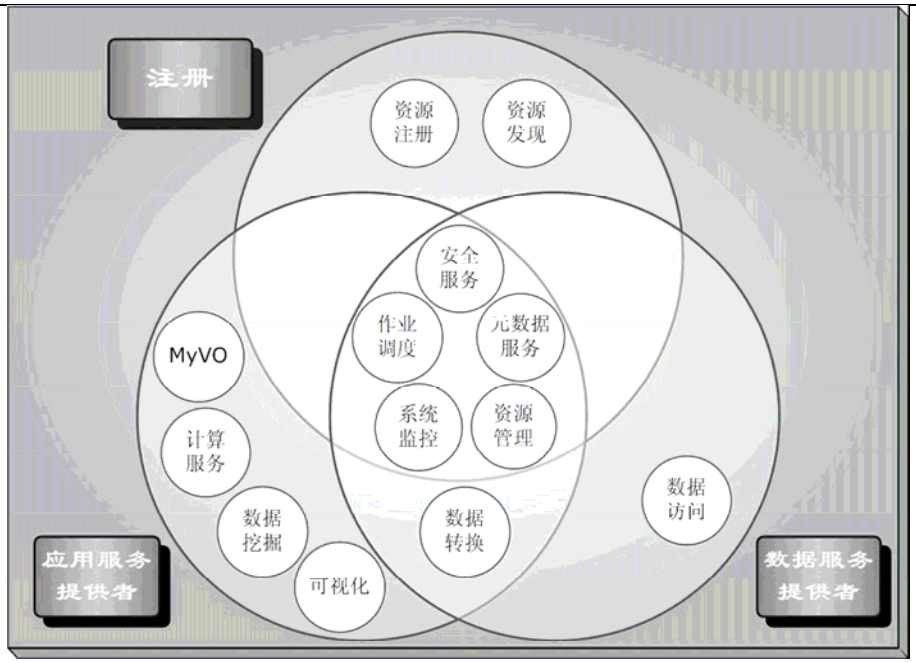


图 3-2：中国虚拟天文台的服务模型

源的监控等多种复杂功能。因此，通过资源管理，实现以最有效的方式为资源使用者分配所需的有效资源，具体来说，需达到以下目标[22]：

1）有效的管理资源，提高资源的利用率

2）屏蔽底层资源的异构性和复杂性

3）管理多个机器协同工作，实现负载平衡

4）支持多种应用使用方式，有效管理多个用户的各种任务

5）按照管理员的意愿可以控制资源的使用方式

6）提供容错能力

7）管理对资源的访问

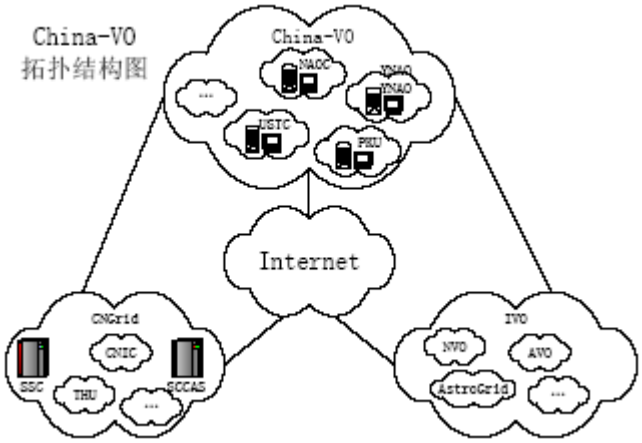
在整个的资源管理系统中，资源的注册与发现是资源管理系统的主要内容。也是其它工作内容的基础。

要实现资源管理的目标，使得系统能够有效地将现有资源融合起来，那么首先需解决的问题就是资源的描述，如何有效的描述资源，以实现互操作性及资源的语义化，其次就是资源如何能被发现及使用，在 SOA 的架构中，资源如何成为服务等。在以下的章节中，对上述问题的解决分别进行了详细的阐述。

第四章　中国虚拟天文台资源的组织与描述

4.1　中国虚拟天文台资源的组织结构设计

虚拟天文台所要管理的资源非常复杂，资源类型异构，分布广泛，如果不能很好的加以组织，对资源的使用效率就很难达到应有的要求，同时，虚拟天文台是一个国际性的合作项目，相互之间必需考虑兼容性，互操作性等要求，当然，这并不代表着所有的 VO 都必需采用相同的结构，但是在相互的接口方面，必须一致。为此，我们先描述一下 China-VO 的拓扑结构。



图：4-1：中国虚拟天文台的拓扑结构图(1)

中国虚拟天文台在建设过程中，不仅要考虑与国际上其它 VO 的连接，还要考虑与国内其它网格项目的融合，尤其是中国科学数据网格，以实现 VO 资源的对外共享。

中国虚拟天文台目前主要管理国内的天文资源，这些资源分布在全国的各大天文台及观测中心，因此，我们首先要把这些资源通过 Intenet 将其连接起来。由于中国虚拟天文台在底层实现上采取 OGSA 的架构，以目前主流的 GT3 来搭建虚拟天文台的网格结点，为此，中国虚拟天文台的资源组织架构如下图所示：

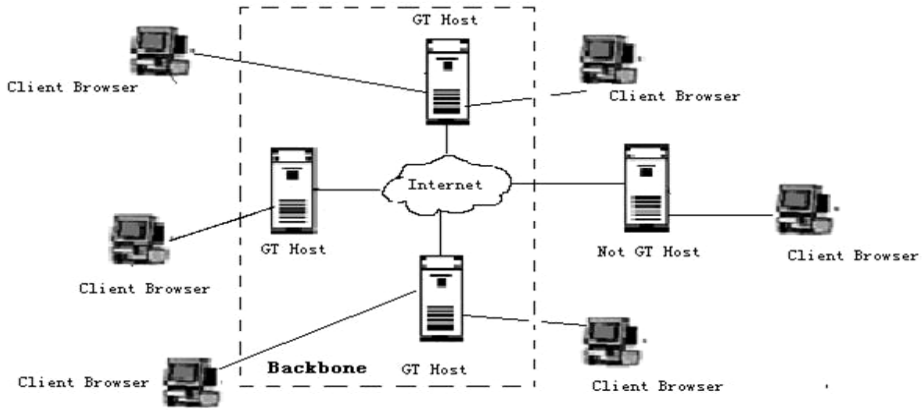


图 4-2：中国虚拟天文台的资源拓扑结构图(2)

GT Host 代表 China-VO 的主要节点，它分布在国内主要天文台里，如上海天文台，南京紫禁山天文台等，以管理局部资源，在用户查询资源的时候，各节点能够主动地搜索到其它节点中的资源，以提供给用户一个完整的资源视图。

在中国虚拟天文台中，对资源的管理必需达到三个基本的要求，一为资源的服务化，二为资源的虚拟化，三为资源的层次化。

4.1.1 资源的服务化

由于 China-VO 采用了 OGSA 的体系结构，所有的资源以服务的方式发布出去，所以提供给用户可用的已经是经过封装的网格服务，通过此网格服务，用户可以对资源进行相关的操作。

虚拟天文台中的资源虽然复杂多样，但我们也不难可以看出，它的资源也具有很多的相似性，我们大致可以简单的将其划分为三大类：

一为数据资源，天文观测的大量数据，是一种丰富且极为珍贵的资源，是天文学家从事天文研究的基础。如果世界各地的天文学家能够将自己观测的天文数据资料提供共享，那将可大大节省其它天文学家的工作。在数据资源的发布上，我们采取对数据的访问封装成网格服务的方式，同时，也给用户提供简单的调用接口，以方便用户获取所需数据。其次为软件资源，就是指天文学家常用的一些数据处理软件，目前，基本上是每个天文学家自己都必须拥有一套相应的软件，因此，在 VO 中，我们需要将这些常用的软件工具封装成服务。三为硬件资源，望远镜等各种观测设备，也需封装成服务，以实

现各种天文仪器的远程共享与控制。这样三类资源服务化的过程基本上代表了中国虚拟天文台建设的三个阶段。

在对资源进行服务化封装的时候，我们不难就会想到，资源是动态变化的，在封装的时候，我们不可能对每个具体的资源进行封装，因为资源只有在资源提供者告诉你的时候，VO 才知道它的存在，为此，在进行服务化的时候，我们必需考虑两个问题，一为资源提供者以什么样的方式告诉 VO，以表明它的存在，其次就是，如何将这个资源转化为服务，基本过程如下图所示：

对于第一个问题，也是 VO 中的资源如何注册与发现的问题，将在下章进行详细的描述。资源转化为服务的具体设计与实现，我们对不同类型的资源采取了有同的方法，详见本章第三节。

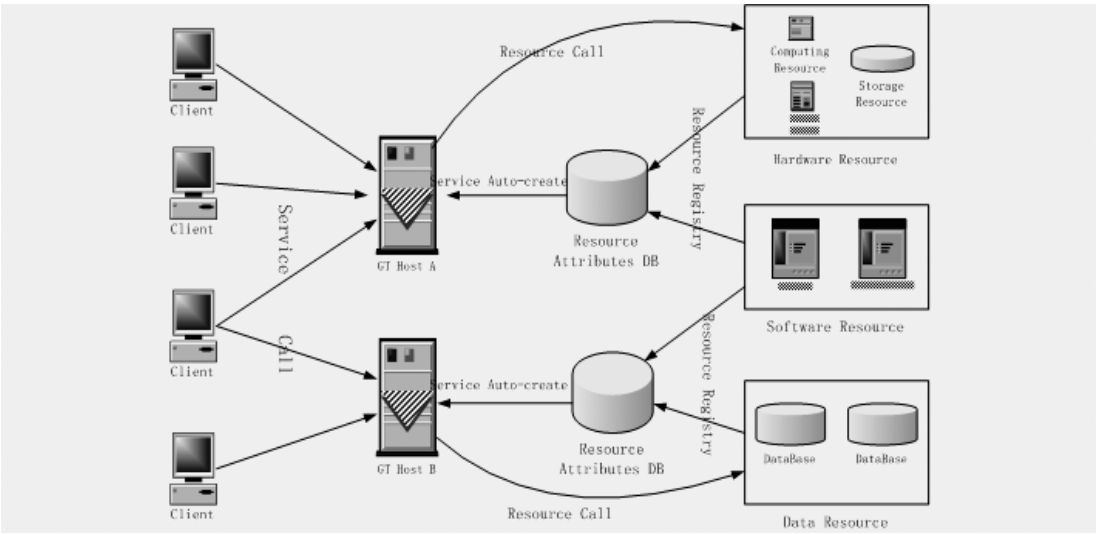


图 4-3：资源与服务的映射

4.1.2 资源的虚拟化

资源虚拟化的基本思想是使分散的和分布式的多个资源看起来就像是一个资源,这样,在访问信息的时候,资源使用者就不需要考虑其所在的位置。资源的使用者并不直接与资源本身发生任何的联系，用户也不知道具体资源的位置及结构特征，他只需提交问题请求，然后得到的就是问题解决后的结果，这一过程都是自动完成的，对用户来说都是透明的.在中国虚拟天文台中，资源变成服务后，资源本身对用户来说，已经实现了资源的透明性，用户只需使用相关的服务，与具体的资源操作无关。通过资源的服务化，

我们实现了基于服务的资源虚拟化。

4.1.3 资源的层次化

中国虚拟天文台的体系结构划分为几个不同的层次，同时，在底层资源的组织上，我们采用了资源树的组织方式，通过树形结构来体现资源本身的层次性，一个资源可能由不同的资源组成，因此，在资源服务化的时候，我们首先要实现资源的元子化，将每个元子化的资源再封装成服务------元服务。这样，不同的服务组合就会有不同的资源需求，从而也实现了资源的有层次的访问。

4.2 资源的描述

4.2.1 资源的语义化

资源描述本身不仅要让人能够无歧义地理解所描述的资源，更重要的是让机器也能相互理解所描述的资源信息。

互联网的普及,信息量的巨大增加,人们很快认识到了它的两个不足：一是计算机不理解网页内容的语义，所有的信息都只有通过人来处理后，才能被接受，二是网上有用信息难找，即使借助搜索引擎，查准率也较低，它在帮助网民得到成批相关网页的同时，也夹杂了许多你所不需要的信息垃圾。问题的关键在于互联网是按“地址”，而非“内容的语义”来定位信息资源的，基于这么一种需求，人们提出了语义网的概念，所谓“语义网”[23]，通俗地说，是按照能表达网页内容的“词语”链接起来的全球信息网；或者说，是用机器很容易理解和处理的方式链接起来的全球数据库,这样，有助于信息与智能的共享，并使网络有能力提供动态与主动的服务，从而更利于人机之间的对话和协同工作。为了实现网页内容的语义化，即实现让计算机能够自动识别和处理网上信息，计算机领域提出了语义网的概念。

语义网的实现大致由以下三部分组成：

一是元数据（Metadata），元数据是：

- 数据的数据 (data about data)
- 结构化数据 (Structured data about data)
- 用于描述数据的内容（what）、覆盖范围（where, when）、质量、管理方式、数

据的所有者（who）、数据的提供方式（how）等信息，是数据与数据用户之间的桥梁；

• 管理、控制信息（Administrative information）

• 是一组独立的关于资源的说明（metadata is a set of independent assertions about a resource ）

• data that defines and describes other data （ISO/IEC 11179-3:2003(E)）

通过元数据的使用，能够更有效的解决以下问题:

• 资源描述（description）

• 资源发现 (resources discovery)

• 认证（authentication）

• 互操作（interoperability）

• 数据管理（data management）

• 访问控制（rights management）

• 数字化保藏（digital preservation）

• 内容分级（content rating services）

二是“资源描述框架”RDF（Resource Description Framework），Resource Description Framework，即资源描述框架。采用“资源-属性-属性值”的“主谓宾”结构（或称三元组），提供一种框架容器，并通过XML 定义了一套形式化的方法，为机器语义理解的结构基础。

RDF 数据模型是个三元组：①资源（或题目），指被描述的人、事物或网页等；②性质（或谓语），它确定用来描述资源的特征、属性或关系；③值（或对象），指命名的性质与资源相关联的一个值，它们就像句子中的主语、动词和宾语。我们举一个例子，可以把一个语义网文件中的每一个 RDF 三元组看作是一个庞大数据库中的一排，主语是关键字，谓语是列，宾语是数值，构成了相关事物的网上信息。

三是本体论（ontology）[24]，牛津英语辞典里对“Ontology”的解释是“对于存在的研究或科学（the science or study of being）”，人工智能领域经常引用Gruber 在1993年的定义“概念体系的规范”（specification of conceptualization）2，1998 年Studer 等人在这个定义的基础上对于本体的特点给出了一个较为明确的解释：“知识本体是对概

念体系的明确的、形式化、可共享的规范说明”。直观地，我们可以把知识本体看成是“领域知识规范的抽象和描述，表达、共享、重用知识的方法。知识本体作为领域概念及概念之间关系的规范化描述，这种描述是规范的、明确的、形式化的，共享的。“明确”意味着所采用概念的类型和它们应用的约束实行明确的定义。“形式化” 指知识本体是计算机可读的(即能被计算机处理)；“共享”反映知识本体应捕捉该领域中一致公认的知识，反映的是相关领域中公认的概念集，即知识本体针对的是团体而非个体的共识。知识本体的目标是捕获相关领域的知识,提供对该领域知识的共同理解,确定该领域内共同认可的词汇,并从不同层次的形式化模式上给出这些词汇和词汇间相互关系的明确定义。如果把每一个知识领域抽象成一套概念体系，再具体化为一个词表来表示，包括每一个词的明确定义、词与词之间的关系（例如“用”“代”“属”“分”“参”关系等）以及该领域的一些公理性知识的陈述（例如“所有的期刊论文都是出版物”）等，并且能够在这个知识领域的专家之间达成某种共识，即能够共享这套词表，所有这些就构成了该知识领域的一个“知识本体”。最后，为了便于计算机理解和处理，需要用一定的编码语言（例如RDF/OWL）明确表达上述体系（词表、词表关系、关系约束、公理、推理规则等）。在这个意义上，知识本体已经成为一种提取、理解和处理领域知识的工具，可以被应用于任何具体的学科和专业领域。对某个知识领域每个人的认识从内容到形式都可能是不一样的，通用的高层知识本体（Common Ontologies）常常从哲学的认识论出发，其概念的根节点往往是时间、空间、事件、状态、对象等抽象术语，而且不一定需要形式化；领域本体（Domain Ontologies）专注于解决领域知识的抽象，较为具体，容易进行形式化和共享；术语本体（TerminologyOntologies）常常表现为一个词表，概念关系的抽取较为随意和简单，不严格要求，甚至可以没有概念定义，例如著名的WordNet 本体；形式本体（Formal Ontology）对于概念术语的分类组织要求较为严格，需要按照一定的分析原则和标准，明确定义概念间的显性、隐性关系，并明确各种约束、逻辑联系等，这类本体常常由术语本体发展而来，但却与术语本体没有截然的界限；另外还有表现本体、任务本体、方法本体、混合型本体等等。

如上所述，某个具体领域的知识本体不可能是唯一的，形式化方式手段也可以不同，但是不同的知识本体必须通过某种机制保证交换和映射的顺利进行，形式化的方式也需

要标准化，这就是知识本体语言的作用。

通过这样三个层次的描述，网页内容就初步实现了语义化，不仅人能理解，机器也能理解，使得接下来的一系列自动化的分析，查找成为可能。

4.2.2 元数据

为了实现语义化的资源描述，它的第一层次就是要定义一套元数据，以描述资源。为此，IVOA注册工作组对天文学上的资源特性进行了研究，结合现有的UCD[25]规范，在OAI-PMH[26][27]规范的基础上制定了一个适合于VO资源的元数据标准，在这个标准中，以一个统一的方式来描述VO中的所有资源。资源元数据[28]主要由标识元数据，履历元数据，内容元数据，数据集元数据，及服务元数据,详述如下：

- ◆ 标识元数据
  - ▼标题（Title）：
    - 数据类型：string
    - 义：资源的名称
  - ▼标签（Ticker）
    - 数据类型：string
    - 定义： 资源名称的一个短小缩写
  - ▼标识（Identifier）
    - 数据类型：URI
    - 定义： 在给定环境中对资源的一个明确指向
- ◆ 履历元数据
  - ▼发布者（Publisher）
    - 数据类型：string
    - 定义：发布资源的实体，它要对资源的可用性负责
    - 发布者标识（PublisherID）
    - 数据类型：URI
    - 定义：资源发布者的标识，可以是一个网络链接
  - ▼创建者（Creator）

数据类型：string

定义： 资源主要内容的创建实体，可以是个人或者组织

▼主题（Subject）

数据类型：string, list

定义：对资源的主题、天体类型或其他属性进行描述的一系列关键词

▼描述（Description）

数据类型：string, free text

定义：对资源内容的一个概括

▼贡献者（Contributor）

数据类型：string

定义：对资源内容有贡献的实体

▼日期（Date）

数据类型：string

定义：在资源的生命期中一个重要的日期，比如创建或者发布日期

▼版本（Version）

数据类型：string

定义：与资源创建或者发布相关的一个标号

▼资源参考（ReferenceURL）

数据类型：URL

定义： 一个指向资源额外信息的URL地址

▼联系方式（Contact）

数据类型：string, e-mail address

定义：资源负责人，包括联系人的姓名和电子邮件地址

◆ 内容元数据

▼类型（Type）

数据类型：string, list

定义：资源内容的种类或类型，可以是Archive、Bibliography、Catalog、

Journal、Library、Simulation、Survey、Education、Outreach、EPOResource、Animation、Artwork、Background、BasicData、Historical、Photographic、Press等其中的一个或几个。

▼范畴（Coverage）

数据类型：string

定义：资源内容的涵盖范畴

▼内容等级（ContentLevel）

数据类型：string, list

定义：内容等级描述，指出资源的潜在用户

▼机构（Facility）

数据类型：string

定义：获取数据的天文台和研究机构

▼设备（Instrument）

数据类型：string

定义：收集数据所用设备

▼格式（Format）

数据类型：string, list

定义：资源所提供信息的物理或者数字显示方式，比如文件格式（FITS，ASCII，VOTable，GIF，…）、存储方式（CDROM、DVD、在线、录像带、出版物，…）等

▼权限（Rights）

数据类型：string

定义：资源的所属及访问权限 95中国虚拟天文台系统设计

◆ 服务元数据

▼接口元数据

服务接口URL（ServiceInterfaceURL）

数据类型： URL

定义：一个指向服务接口说明文档的URL

服务基准URL（ServiceBaseURL）

数据类型： URL

定义：用户激活服务的URL的基本部分

HTTP服务接口返回结果（ServiceHTTPResults）

数据类型： (MIME type)

定义：服务返回结果的MIME类型

▼功能元数据

服务标准URI（ServiceStandardURI）

数据类型：URI

定义：标识一个服务标准的URI

服务标准URL（ServiceStandardURL）

数据类型： URL

定义：一个指向服务标准描述的URL

服务最大搜索范围（ServiceMSR）

数据类型：float, decimal degrees

定义：服务提供者为服务附加的最大搜索范围，以度的形式规定最大的搜索半径

4.3 资源服务化的实现

资源服务化是中国虚拟天文台必需实现的目标之一，也是采用 SOA 架构的必然要求，为此，我们分三类资源来详细描述资源服务化的实现。

4.3.1 数据资源(Data collection)

数据资源是 VO 中最普遍的，也是最重要的资源，同时也是 VO 中首先要实现的服务之一。

VO 中的数据资源主要是星表及天区数据，是从事天文研究的基础，将这些资源封装成服务，本质上就是提供一个用户访问数据的接口。为了更好的实施服务管理，以及

工作流的实现，我们提出元资源(metaresource)及元服务(metaservice)的概念。

元资源：在资源的使用中不能再分的资源，在数据资源中，它可以是一张表，也就是数据集(data collection).

元服务：实现单一功能的服务或封装元资源的服务。

将资源划分为元资源以后，可以简化很多的操作，任务复杂的资源可以看成是由元资源构成的，所以，我们只需预定义一些对元资源的基本操作，对于千变万化的数据资源，最基本的就是数据集，其主要的操作都是一致的。封装的基本流程如图 4-4 所示。

资源的元数据信息被保存在资源元数据库中，服务生成时必需从元数据库中获取相关的数据，生成服务的相关配置文件,其中主要是 wsdl，另一方面，对于数据集的主要操作，特别是针对于天文数据这种特定的领域，对数据的操作就更为规范，基本上主要是一些数据查询的操作。获取指定天区的数据是数据查询的主要目的。服务的 wsdl 的主要定义如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="DBservice"
    targetNamespace="http://www.lamost.org/namespaces/lamost/Database"
     xmlns:tns="http://www.lamost.org/namespaces/lamost/Database"
     xmlns:data="http://www.lamost.org/namespaces/lamost/servicedata"
    xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
    xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions
"
    xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns="http://schemas.xmlsoap.org/wsdl/">
<import location="../../ogsi/ogsi.gwsdl"
    namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
<import location="MathDataType.xsd"
    namespace="http://www.lamost.org/namespaces/lamost/servicedata/"/>
```

```
<import location="DBdataType.xsd"
    namespace="http://www.lamost.org/namespaces/lamost/servicedata/"/>
<types>
<xsd:schema targetNamespace="http://www.lamost.org/namespaces/lamost/Database"
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <!-- Search -->
    <xsd:element name="search" type="tns:search">
        <xsd:complexType name="search">
            <xsd:sequence>
                <xsd:element name="b" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
    <xsd:element name="searchResponse" type="tns:searchResponse">
        <xsd:complexType name="searchResponse">
            <xsd:sequence>
                <xsd:element name="return" type="xsd:string"/>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
</types>
<message name="SearchInputMessage">
   <part name="parameters" element="tns:search"/>
</message>
```

```
<message name="SearchOutputMessage">
  <part name="parameters" element="tns:searchResponse"/>
</message>
<gwsdl:portType name="DatabasePortType" extends="ogsi:GridService
ogsi:NotificationSource">

  <operation name="search">
   <input message="tns:SearchInputMessage"/>
   <output message="tns:SearchOutputMessage"/>
   <fault name="Fault" message="ogsi:FaultMessage"/>
  </operation>
  <sd:serviceData name="DBdata"
                  type="data:DBdataType"
                  minOccurs="1"
                  maxOccurs="1"
                  mutability="mutable"
                  modifiable="false"
                  nillable="false">
  </sd:serviceData>
</gwsdl:portType>
</definitions>
```

数据集资源元数据的定义如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions name="DBdata"
    targetNamespace="http://www.lamost.org/namespaces/lamost/servicedata"
    xmlns:tns="http://www.lamost.org/namespaces/lamost/servicedata"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
```

```
<wsdl:types>
<schema targetNamespace="http://www.lamost.org/namespaces/lamost/servicedata"
    attributeFormDefault="qualified"
    elementFormDefault="qualified"
    xmlns="http://www.w3.org/2001/XMLSchema">
    <complexType name="DBdataType">
        <sequence>
            <element name="location" type="string"/>
            <element name="dbname"    type="string"/>
            <element name="tablename" type="string"/>
            <element name="protocol" type="string"/>
            <element name="port" type="string"/>
            <element name="type" type="string"/>
            <element name="user" type="string"/>
            <element name="password" type="string"/>
        </sequence>
    </complexType>
</schema>
</wsdl:types>
</wsdl:definitions>
```

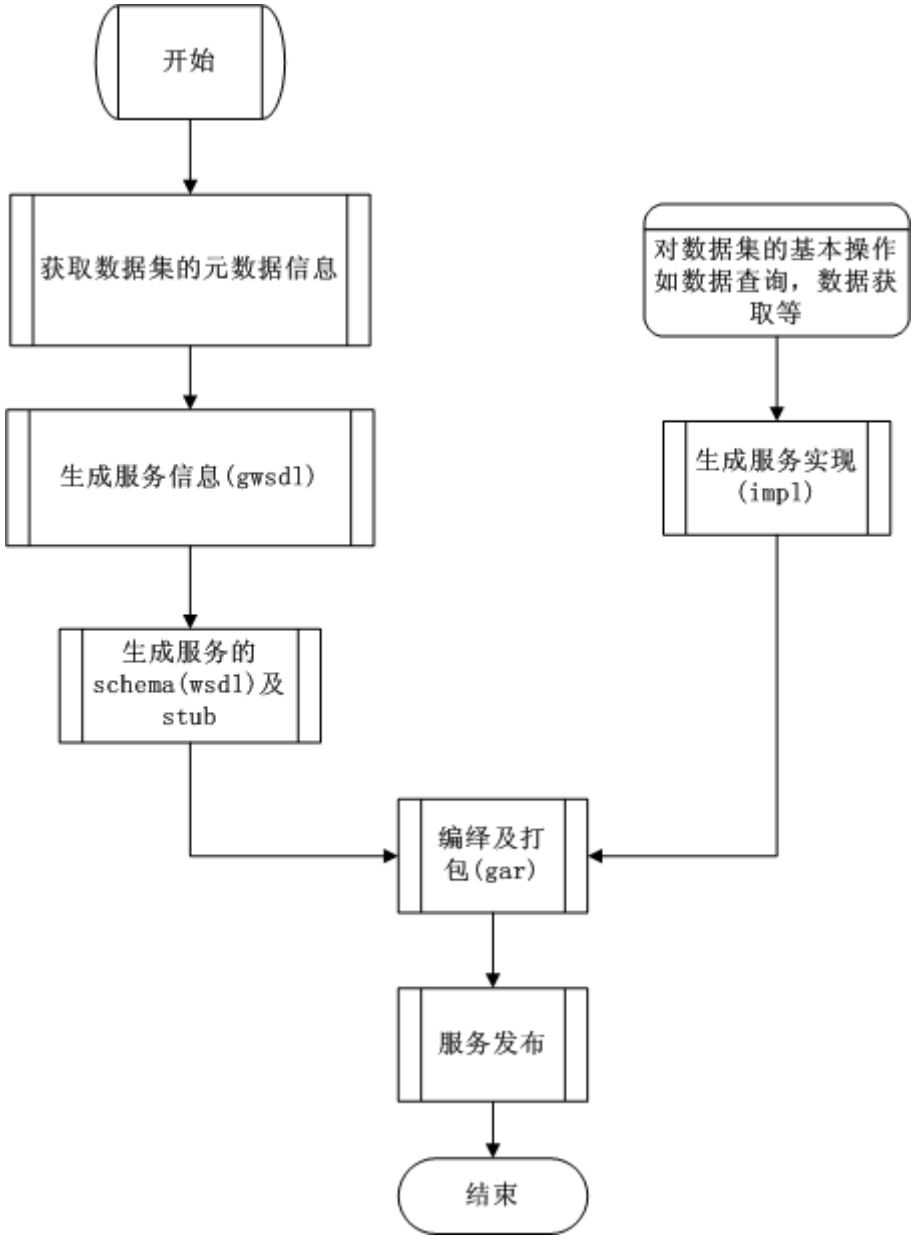


图 4-4 数据资源的服务封装

数据资源封装服务从元数据库中获取相关数据集的元数据信息，这些信息表示了数据集的位置，及访问的基本要求，再与基本的服务操作结合起来，形成一个对于特定的数据资源的服务。

4.3.2 软件资源

软件资源的服务化是服务网格研究的一个热点领域。随着网格研究向应用的过渡，面临的主要问题就是现有的软件如何在网格环境下使用，也即软件网格化的问题，我们面临的选择只有两个，一是一切从头开始，其次就是对现有软件作些改进。显然，第一

种方法是不现实的，也是不可取的，为此，我们只能在第二种方法上进行努力

在中国虚拟天文台的建设中，我们同样面临着这样的问题，目前天文学家应用的各种天文数据处理软件，作图软件等工具不可能随着 VO 的到来而丢弃，否则，VO 就没有了群众基础。所以我们必需将这些软件集成到我们的 VO 环境中。这也就是软件的网格化，而对于我们采用了 OGSA 的特定平台来说，又可以说是一种软件的服务化过程，需要使用某一种软件时，只需调用相关的服务即可。

在 VO 中，我们采用了两种方法来实现软件的服务化，一是对软件本身进行改造，使得它本身成为一个网格服务。这种方法一般来说，是对一些开源软件，我们可以在它的源代码的基础上加一些接口，以封装成一个服务，如我们现在已做的作图服务(ImageMagic),及一些简单的天文计算方面的软件，另一种方法，就是软件本身不动，我们在其上加一个代理，负责软件与网格环境，用户之间的交互。第一种方法具有针对性，对每个不同的软件都得重做一次，而对于第二种方法，具有一定的通用性，很多软件可以共用一个代理，但不足就是不太适用于与用户交互太多的及图形界面的软件。

4.3.2.1 从源代码到网格服务

在对软件进行服务封装时，首先要确认提供给外部调用的接口有哪些，这样才能写出完整的服务描述及实现接口，其次要根据源代码的类型确定服务的实现。其完整的实现流程图如下：

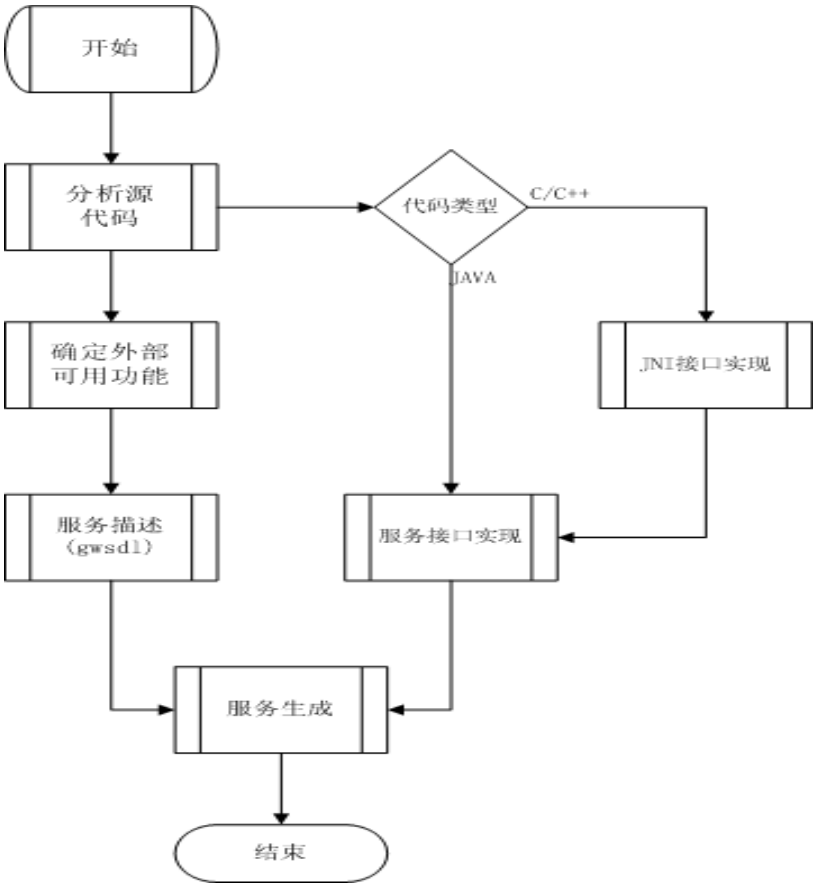


图 4-5：软件资源的服务化（1）

通过对网格服务编写的过程进行分析以后，我们发现在实现上有很多共同的东西，如生成 gwsdl,编写接口实现程序等方面有些是固定的步骤。这为服务封装的自动化提供了基础。同时，在 VO 项目内，有很多的现成的小程序，都能实现一定的功能，同时，我们也拥有很多的程序员，但并不是每个人都知道网格，知道如何写网格服务，为此，我们开发了一个服务自动封装系统，世界各地的程序员，只要对 VO 有兴趣，愿意提供自己的程序，我们就能将其发布为服务，供所有的人使用，这一可大大节省程序员的工作量，同时，也丰富了 VO 的资源。自动封装的流程如下：

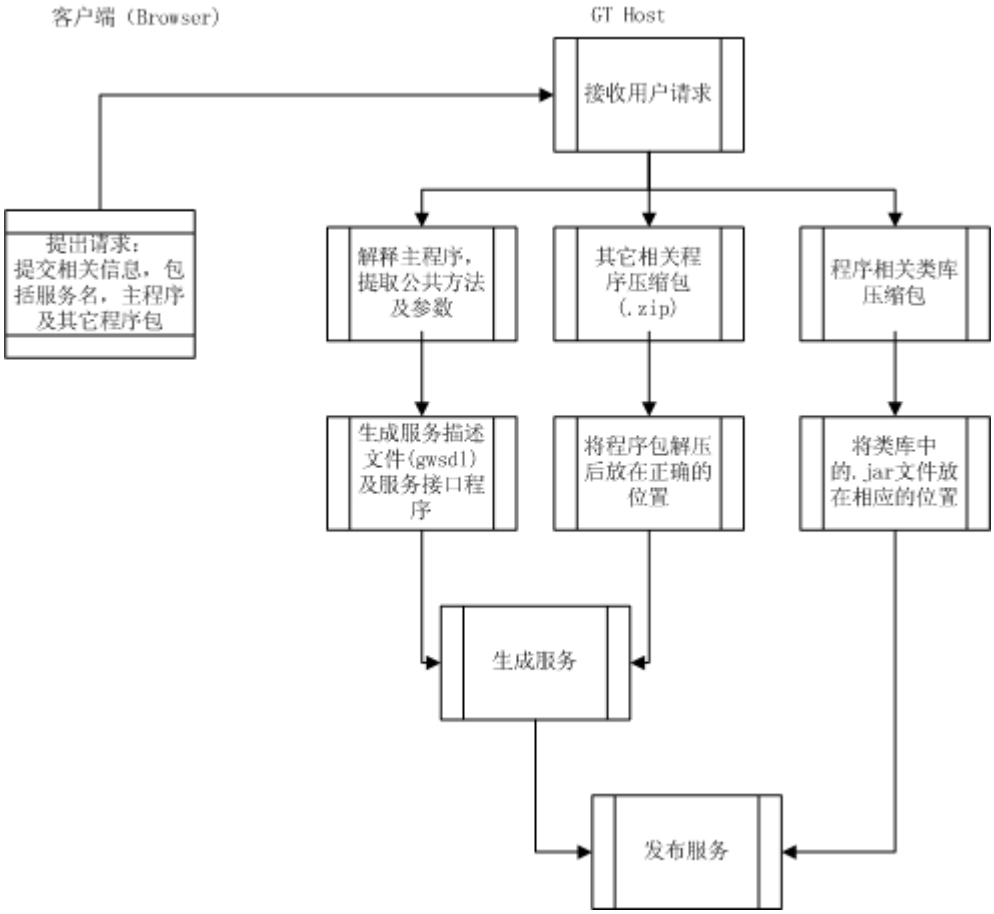


图 4-6：服务自动封装的流程

在整个过程中，关键的步骤就是如何根据用户的主程序来生成相应的服务描述文件及服务接口程序。在获取服务方法时，我们采用逐词扫描的方式，这种方式比较简单有效，因为可供外部调用的方法肯定是公共方法，因此，我们只需从源程序中提取出公共方法的说明就可以确定此方法的名字，参数个数及参数类型等信息。然后在服务实现文件中定义同样的方法，而此方法的内容就是调用用户提供的源程序中同名的方法，具体实现如图 4-7 所示，这样，就实现了 JAVA 源程序到网格服务的自动封装。

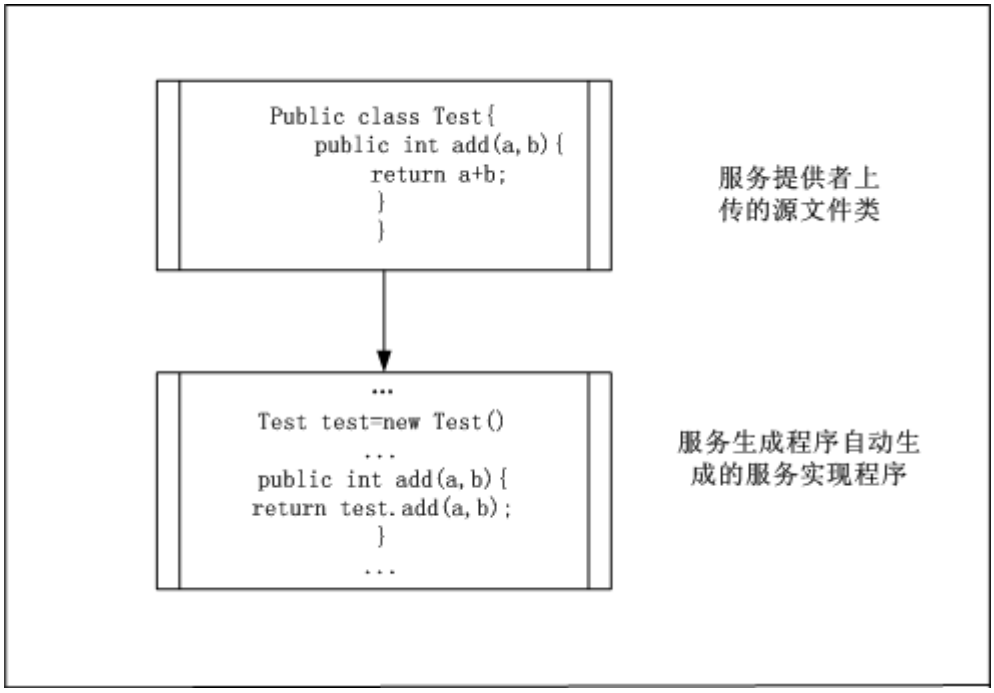


图 4-7：源程序服务化的实现

4.3.2.2　软件代理

实现软件服务化的另一种方式就是为现有软件加一个代理，负责接收用户请求，启动软件，调用相关功能，返回执行结果等。结构如下所示：

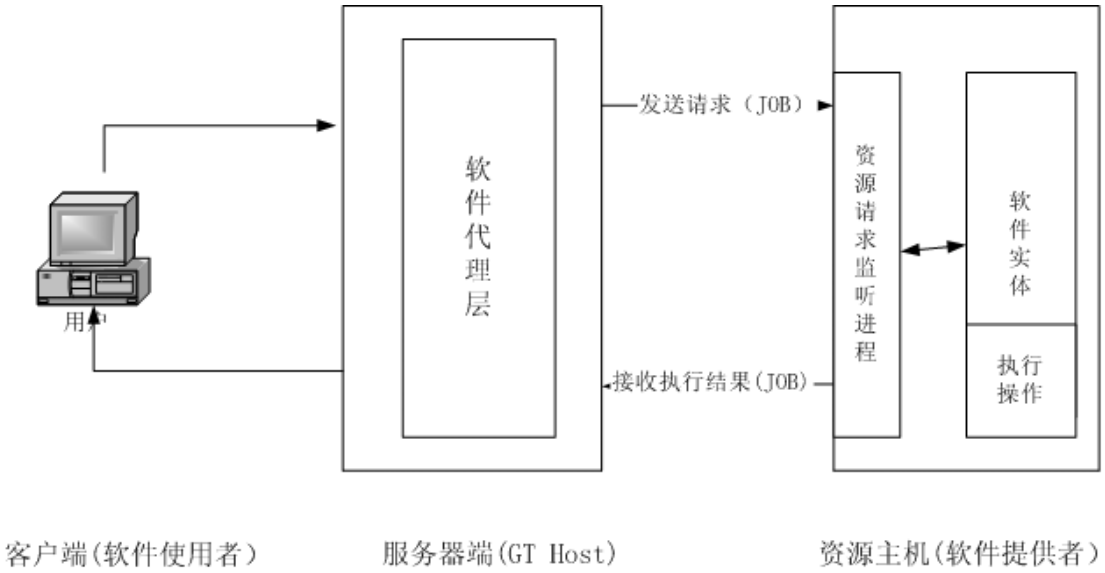


图4-8：软件的服务化封装(2)

软件的代理实际上也是一个网格服务，它的一个服务实例就对应到一个具体的资源

主机，其上的所有软件都可共用此代理。那么它是如何工作的呢，用户需要使用某种软件时，先寻找一个代理的实例(见服务查询)，在这个代理中主要记录了软件所在的主机位置。然后，用户提交一个功能请求，代理首先要将这个功能请求转化为标准的请求头，在这里我们称它为一个工作（JOB）。请求头发送到资源主机上后，经过资源端监听进程解析后起动相应的进程，再将结果填入请求头的result字段中，然后，将此请求头重新返回。代理重新接受此请求头（此时已变成应答），检索出结果字段，返回给服务调用者。

一个工作的完整描述如下：

name: 需执行的软件名字；

path: 软件所在主机的位置;

flag: 参数类型,说明参数本身就是具体的值，还是一个指向参数的地址(目前不支持地址);

argu: 参数;

result: 存储返回结果;

上述信息除flag及argu外，其它从资源提供者提供的资源元数据中得到。用户只需提供相应的参数，就可调用所需的软件。

代理服务的描述如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<definitions name="SRservice"
targetNamespace="http://www.lamost.org/namespaces/SoftResource/Software"
xmlns:tns="http://www.lamost.org/namespaces/SoftResource/Software"
xmlns:data="http://www.lamost.org/namespaces/SoftResource/servicedata"
xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
xmlns:gwsdl="http://www.gridforum.org/namespaces/2003/03/gridWSDLExtensions"
```

```
        xmlns:sd="http://www.gridforum.org/namespaces/2003/03/serviceData"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema"
        xmlns="http://schemas.xmlsoap.org/wsdl/">
    <import location="../../ogsi/ogsi.gwsdl"
        namespace="http://www.gridforum.org/namespaces/2003/03/OGSI"/>
    <import location="SoftResDataType.xsd"
    namespace="http://www.lamost.org/namespaces/SoftResource/servicedata/"/>
    <types>
    <xsd:schema
targetNamespace="http://www.lamost.org/namespaces/SoftResource/Software"
        attributeFormDefault="qualified"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema">
        <xsd:element name="launch" type="tns:launch">
            <xsd:complexType name="launch">
                <xsd:sequence>
                    <xsd:element name="name" type="xsd:string"/>
                    <xsd:element name="argu" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
        <xsd:element name="launchResponse" type="tns:launchResponse">
```

```
            <xsd:complexType name="launchResponse">
                <xsd:sequence>
                    <xsd:element name="return" type="xsd:string"/>
                </xsd:sequence>
            </xsd:complexType>
        </xsd:element>
    </xsd:schema>
    </types>
    <message name="LaunchInputMessage">
      <part name="parameters" element="tns:launch"/>
    </message>
    <message name="LaunchOutputMessage">
      <part name="parameters" element="tns:launchResponse"/>
    </message>
    <gwsdl:portType name="SoftwarePortType" extends="ogsi:GridService
ogsi:NotificationSource">
      <operation name="launch">
        <input message="tns:LaunchInputMessage"/>
        <output message="tns:LaunchOutputMessage"/>
        <fault name="Fault" message="ogsi:FaultMessage"/>
      </operation>
      <sd:serviceData name="SAData"
```

```
                type="data:SoftResDataType"
                minOccurs="1"
                maxOccurs="1"
                mutability="mutable"
                modifiable="false"
                nillable="false">
        </sd:serviceData>
    </gwsdl:portType>
    </definitions>
```

软件资源元数据的核心描述如下：

完整的XML格式如下:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <wsdl:definitions name="SoftResData"
        targetNamespace="http://www.lamost.org/namespaces/SoftResource/servicedata"
        xmlns:tns="http://www.lamost.org/namespaces/SoftResource/servicedata"
        xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
    <wsdl:types>
    <schema
targetNamespace="http://www.lamost.org/namespaces/SoftResource/servicedata"
        attributeFormDefault="qualified"
        elementFormDefault="qualified"
        xmlns="http://www.w3.org/2001/XMLSchema">
```

```
<complexType name="SoftResDataType">
    <sequence>
        <element name="location" type="string"/>
        <element name="name"    type="string"/>
        <element name="path"    type="string"/>
        <element name="function" type="string"/>
        <element name="protocol" type="string"/>
        <element name="port" type="int"/>
        <element name="paranum" type="int"/>
        <element name="flag" type="int"/>
    </sequence>
</complexType>
</schema>
</wsdl:types>
</wsdl:definitions>
```

4.3.3 硬件资源---天文仪器的 VO 集成

硬件资源是一个比较宽泛的概念，包括计算方面的硬件资源，如 CPU，内存，磁盘空间，其它外设，也包括天文仪器，如望远镜等。对于这些硬件资源的共享，很难有一种共同的方法，只能具有针对性的来单独实现。共享 CPU，内存，我们开发了一个分布式计算服务，以充分利用 VO 中的计算资源，共享磁盘空间，我们开发了一个虚拟存储服务，而对于其它方面的设备资源的共享，是 VO 的高级目标，在实现数据，软件的服务化以后再加以考虑。

第五章 中国虚拟天文台资源注册与查询

在前一章，我们详述了资源的统一描述方式，定义了资源描述的元数据标准，但是如何维护资源的信息，如何动态更新资源的信息是虚拟天文台资源管理的一个关键问题，在本章，我们主要来阐述资源信息的管理与维护。

5.1 注册的概念与需求

5.1.1 注册及分类

简单地说，注册(Registry)就是一个被结构化数据描述的资源的仓库（这种结构化的数据就是资源元数据标准所定义的）。它提供了给用户发现资源与服务的手段，也提供给资源所有者发布资源与服务的场所。他们通过资源元数据进行相互的理解与交流。也使得不同的虚拟天文台的注册之间能相互理解与交流。

在虚拟天文台里，根据功能的不同，可把注册分为可查询的注册及发布的注册，可查询的注册就是可直接提供给用户查询各种资源信息的注册中心，而可发布的注册则是只面向资源发布者发布资源。而据注册中心所含资源的多少，又可分为完全注册（full-registry）和本地注册(local-registry).完全注册就是包含了 VO 中所有资源的注册，通过这个注册中心，就可以查询到 VO 中包含的所有资源。而本地注册，一般只包含了局部的资源，通常情况下，本地注册是发布性的注册,而完全注册是可查询的注册。

5.1.2 注册的需求

根据注册所要实现的功能，因此，它必须满足以下方面的要求[29]：

在注册的内容方面，必须以元数据的方式实现对资源的描述，对服务的描述，并且在服务方面，要明确如何去调用服务，其中包括服务的输入输出，服务的特点，或者说，注册要能维护资源元数据所定义的所有的信息。

其次，在对注册的查询方面，要能够实现基于特征的资源与服务的查询，基于特定类型的资源与服务查询，以及实现对资源与服务的描述的查询，并且，所有查询的结果都必须以统一的格式返回----即资源元数据标准。

最后，在对注册的过程及升级方面，注册必须对资源提供者提供简单方便的操作，以实现资源的注册，更新等，同时，也要求注册能自动侦测到服务是否有效等。以便及时的将失效服务注销。

5.1.3 注册的接口标准

为了实现各虚拟天文台注册之间能够共享数据资源，因此，必须实现相互之间的互操作性。为了实现这个目标，IVOA 注册工作组定义了注册必须实现的一些标准接口，并且这些接口必须以 Web Service 的方式加以实现。

注册的查询接口[30]：

1. Search(ADQL 串):支持 ADQL 的查询,ADQL 全名是天文数据查询语言。
2. KeywordQuery(KeyWord): 基于关键字的查询，关键字刻划了资源的主要特征。
3. GetRegistries(): 查找 VO 中的其它注册。

这三个接口是可查询的注册必需实现的。

注册获取(Harvesting)接口：

获取（Harvesting）是指一个注册从其它注册获取资源信息记录的机制。它通常被完全可查询的注册来获取其它许多可发布的注册中的资源信息。在IVOA的注册规范中，定义了 6 种获取接口。

Identify ：获取关于注册的标识符

ListIdentifiers ：列出注册中指定条件的资源记录的标识符

ListRecords ： 列出注册中指定条件的所有记录

GetRecords ： 列出注册中的所有记录

ListMetadataFormats ：列出注册支持的元数据的格式

ListSets ：列出注册中资源信息的分类

5.1.4 注册的体系模型[31]

注册的模型分为资源模型及注册模型，分别图 5-1,5-2 所示：

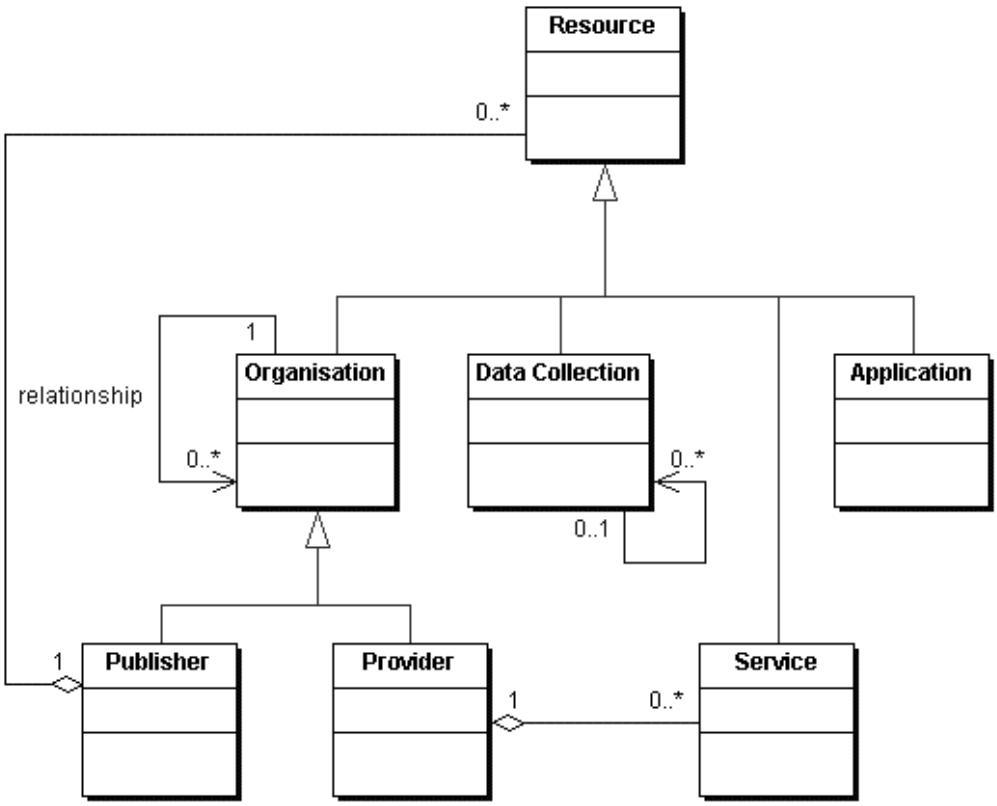


图 5-1：VO 资源模型

5.2 中国虚拟天文台注册的设计

中国虚拟天文台对注册的定位是可查询的完全注册中心，因此，它必需实现注册的所有标准接口，同时，由于 china-vo 采用了 SOA 的体系结构，并且以 OGSA 为基础，VO 中的所有服务都是网格服务，而不是一般的 Web Service，所以，china-VO 的注册必须要能与网格紧密结合起来.目前，NVO，Astrogrid 的注册实际上只是一个资源元数据的仓库，通过查询，它只能告诉你服务的一些基本信息，而并不能直接加以调用，也就是说注册与服务调用是相互分离的东西，而采用了网格以后，我们能够紧密地将两者结合起来。

5.2.1 VO 注册的总体设计

为了使资源注册中的资源元数据与服务本身紧密的结合起来，以便于用户对资源的

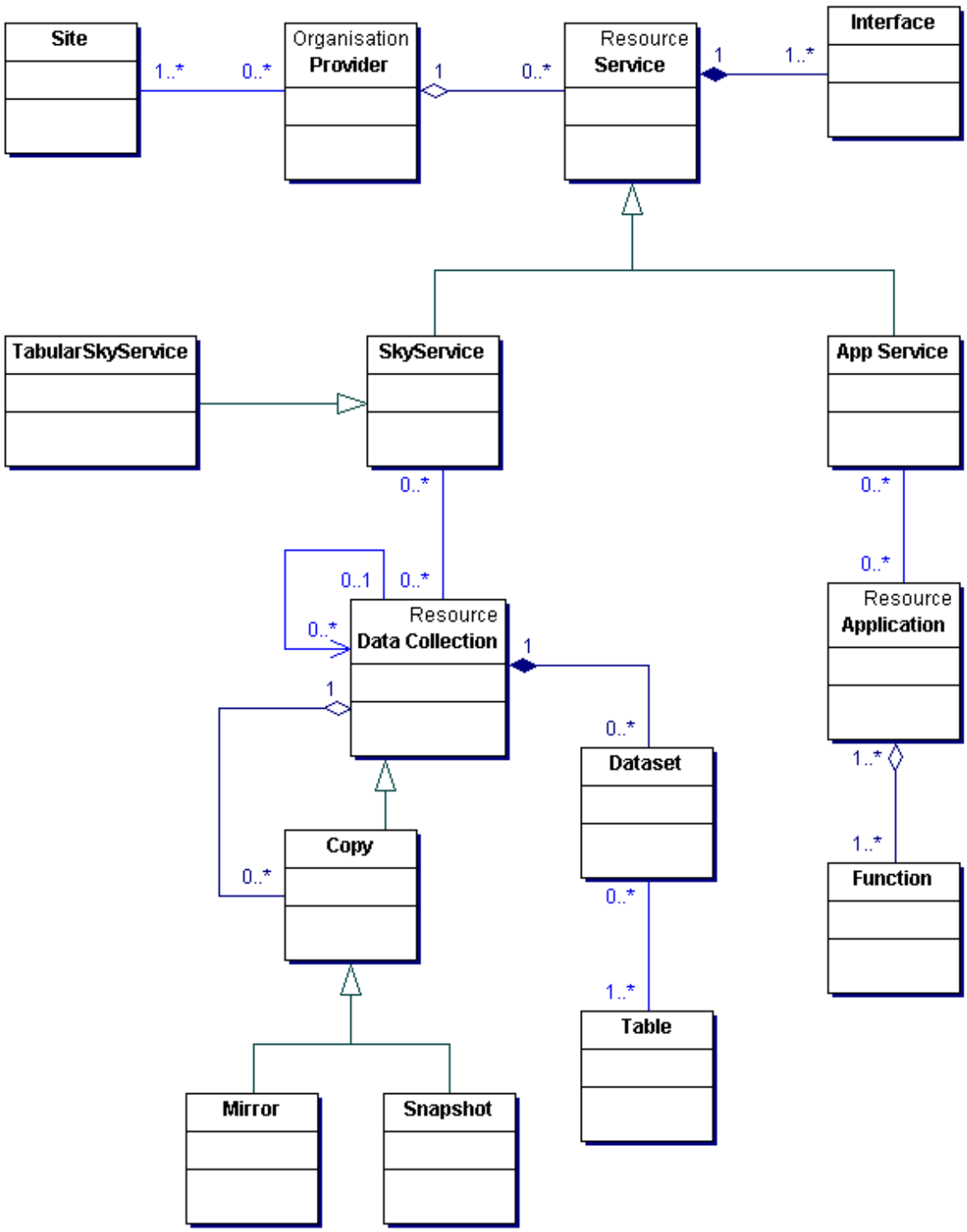


图 5-2：VO 注册模型

定位与使用，我们在注册的总体设计上采用了两套机制，一为资源元数据的注册，另一为服务的注册。两者之间通过资源服务化的手段联系起来。其模型如图 5-3 所示：

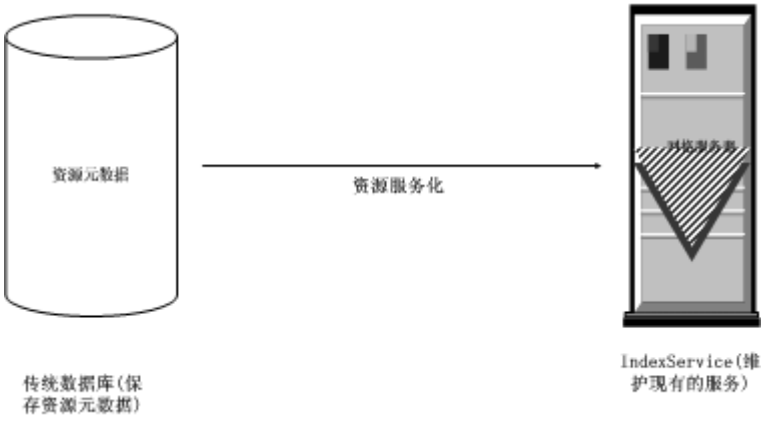


图 5-3：资源中心与服务中心

在这样的一种结构中，资源元数据与其相对应的服务必需维持动态的一致变化，在这样的结构下，资源使用的过程大致如下：

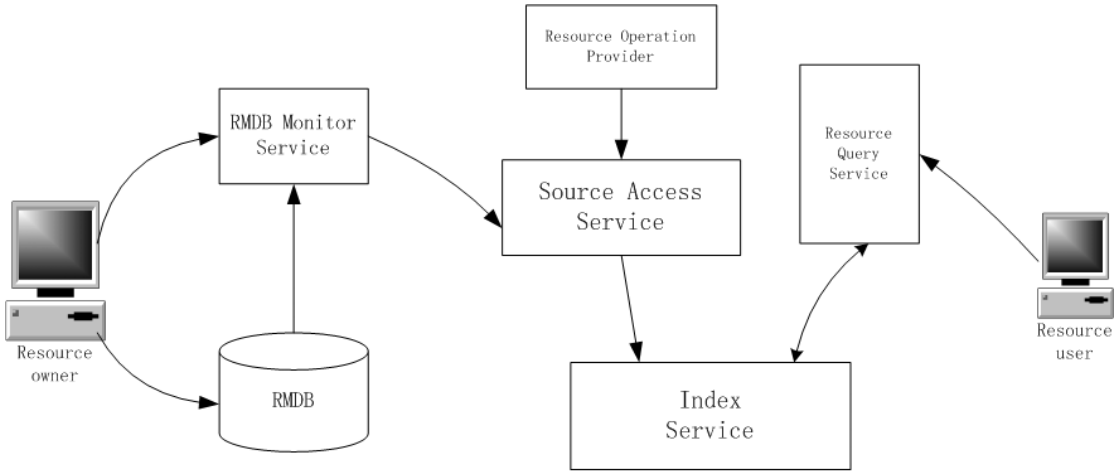


图 5-4：资源的服务化流程

对于每一种资源类型的服务化的方法，我们已在第二章进行了详细的描述，在此不再细说。

5.2.2 资源元数据中心

在资源元数据的维护方面，我们采用了传统数据库 mysql，通过注册表单，资源提供者向注册中心注册资源数据，同时，我们对数据库进行了一次封装，以网格服务的形

式实现了注册所需的接口。在资源元数据的发布方面，我们采用了传统的三层架构的方式，为资源提供者发布自己的数据，对于不同类型的元数据，采用不同的服务化手段，在 IndexService 服务器中维护着对应的服务。

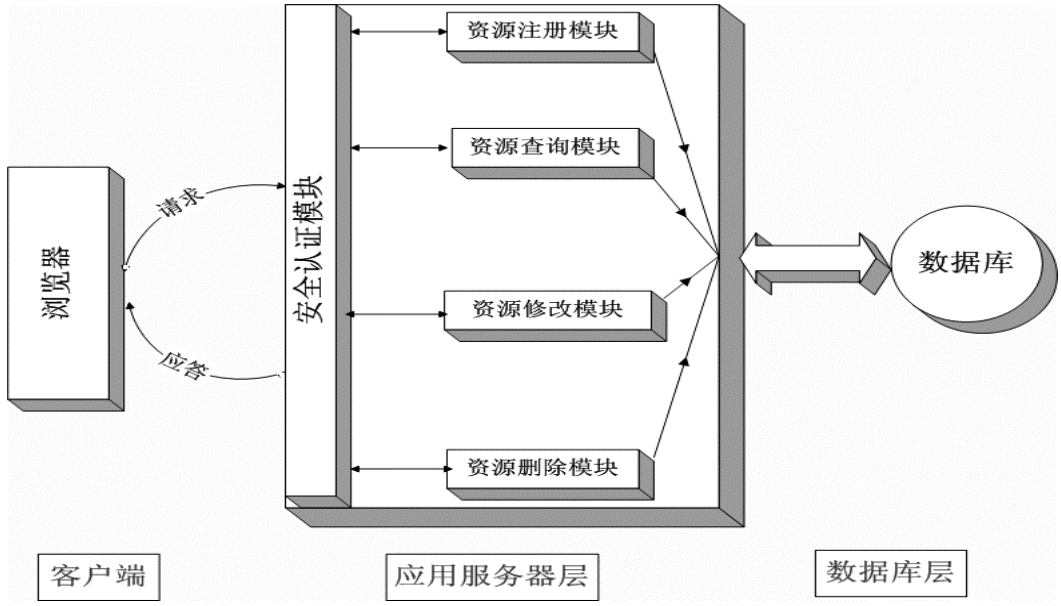


图 5-5：资源元数据的注册

5.2.3 Index Service

Index Service 是 GT3 中信息管理的主要服务，负责网格服务的注册及发现，查询，以提供网格资源的信息。它主要由服务组的管理机制，服务的注册机制，服务数据的预定通知机制，服务数据的查询机制组成，能实时反映出网格当前的服务信息。它的主要功能有：

a)　为服务数据提供者到服务实例的连接提供了一个接口。

b)　能将来自多个不同的服务数据提供者或其它的网格服务的服务数据有效的聚集起来，并能提供一种有效的对服务数据的查询过程。

c)　实现网格服务的注册。

d)　动态的服务数据产生及网格节点索引。

除了实现服务的注册功能之外，Index Service最主要的功能就在于对服务数据的管理，包括服务数据的创建，查询，预定，通知，删除等以及各种相关的服务数据的聚集

机制。通过对相关服务数据的操作，就可以获取网格资源的实时信息，如通过查询Index Service的entry服务数据，就可获取所有在Index Service上已注册的服务信息[17]。

5.2.4 网格服务的自动注册

对于一个服务提供者来说，他所关注的只是自己的服务是否有效，而不会注意是否在注册中心有它的链接，这就要求注册中心能够提供这样一种机制，当服务有效时，能自动发布到注册中心，而当服务失效时，又能自动从中清除，我们在 GT3 的基础上，经过二次开发，满足了这种需求。它的基本思路是，在网格服务起动时，执行注册操作，注册操作从注册配置文件中获取注册中心的地址，从而与注册中心建立连接，把服务的 URL 记录到注册中心的 IndexService 中，此后，IndexService 会起动查询机制，每隔一定的时间就会查询服务的有效性，当服务失效时，就会将此服务删除。注册动作本身与服务无关，我们以操作提供者的方式附加到服务中，服务提供者只需在服务发布描述符文件中加上，再写一个注册配置文件。这样简化了服务编写人员的负担,也极大地提高了系统的效率。基本的配置文件如下：

.wsdd 文件，主要有以下二行：

```
<parameter name="operationProviders" value=" org.globus.ogsa.impl.core.
registry. RegistryPublishProvider "/>
<parameter name="registrationConfig" value="etc/registration_config.xml"/>
```

registrationConfig.xml 注册配置文件的详细内容如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceConfiguration
xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
xmlns:aggr="http://www.globus.org/namespaces/2003/09/data_aggregator"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<registrations>
<registration
                registry="http://www.china-vo.org:8080/ogsa/services/base/index/Inde
        xService"
```

```
                keepalive="true"
                lifetime="200"
                remove="true">
            </registration>
        </registrations>
</serviceConfiguration>
```

主要参数：

registry: 指定服务注册中心的地址

lifetime: 查询周期

keepalive:是否在整个周期内进行通知

remove: 服务失效时，是否从注册中心去除

其注册操作的基本流程如下：

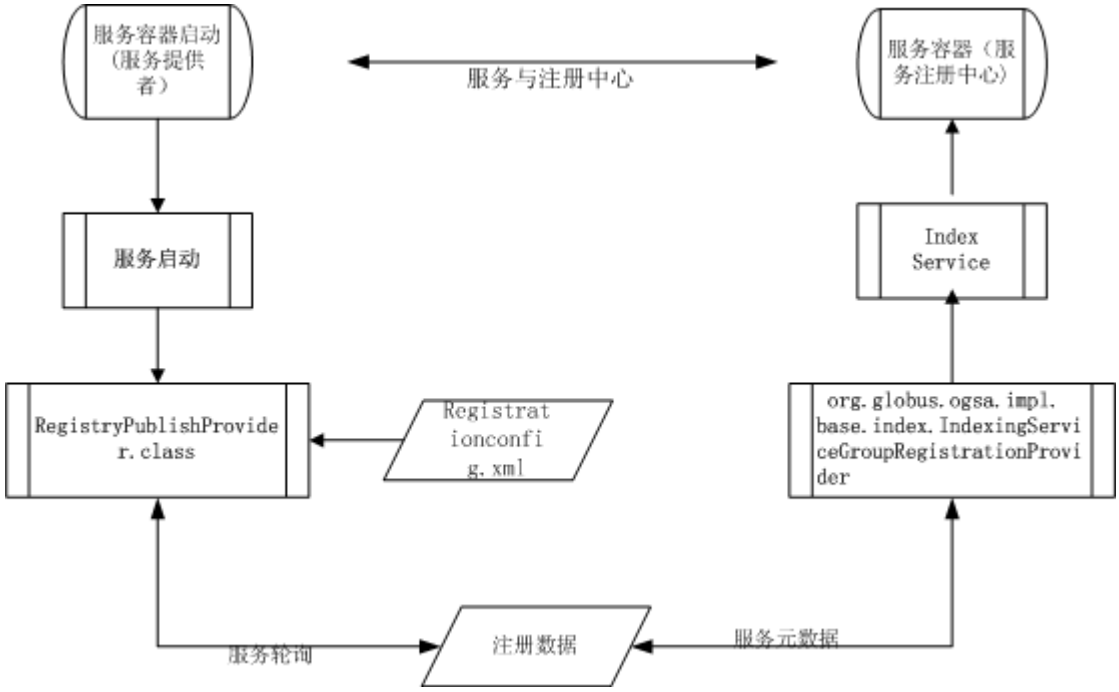


图 5-6：服务注册流程

5.2.5 基于 IndexService 的服务获取

在注册的两个部分中，对于资源元数据的获取，我们实现了 IVOA 的标准注册接口，通过这些接口，可以查询到自己想要的资源描述，同时，我们也提供了通过 IndexService 的查询接口，来实现服务的查询与调用。

在 china-vo 的整个环境中，是由多个注册中心组成的，多个注册中心之间，对于资源元数据的获取，我们通过标准接口中的 harvest 接口来实现，而对于服务自身状态信息的获取与查询，我们以网格服务的形式又提供了新的接口。

5.2.5.1　服务状态信息的查询

在注册中心，我们首先维护着 VO 中所有注册中心的元数据，采用并行的技术，同时对所有的注册中心的 IndexService 进行查询，然后将结果进行融合，封装成标准的格式返回给用户.其主要的实现流程如图所示.

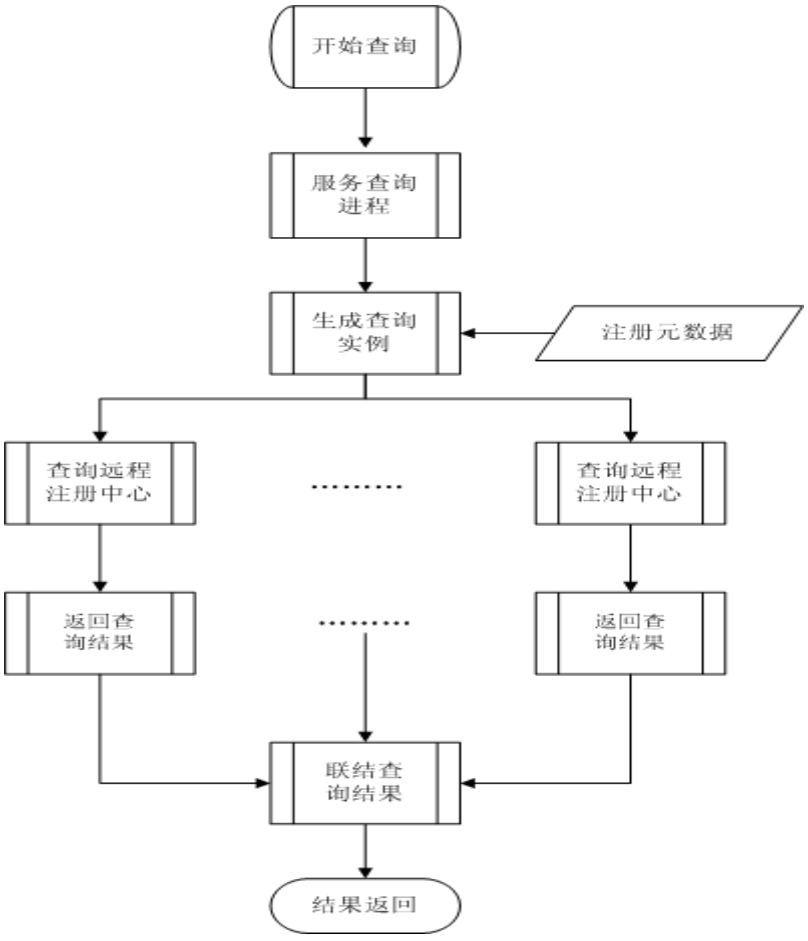


图 5-7：服务状态信息查询

5.2.5.2　基于服务名的服务查找与实例创建

这种方式的查找是定位到一个可用的服务实例返回给用户，以便于用户能够直接调

用返回的服务，所以，这种查找的返回结果为服务实例的句柄。用户通过这个 GSH，就可调用这个服务提供的功能。同样，在不同的注册中心，都可能会有这种类型的服务，如何找一个最好的（负载最轻的）服务给用户，这是我们的目的，具体实现见第 6 章。

5.2.6 基于 IndexService 的信息管理系统的实现

前面，我们详细的介绍了资源信息注册及管理的各个方面的设计，在具体的整体实现上，我们可以与 China-VO 的各个层次对应起来，其整体的结构图如下：

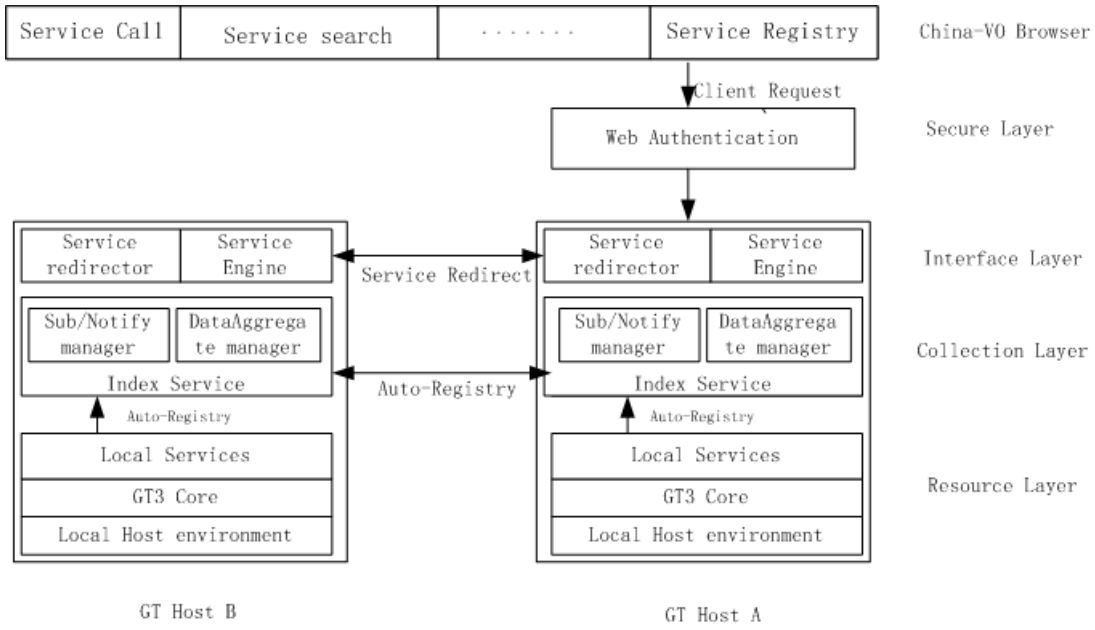


图 5-8：服务信息管理系统的层次结构

最顶层为 China-VO browser，为一般用户查询 VO 资源信息及对 VO 服务功能的调用提供了极为方便的界面。用户只要通过浏览器，就可获取丰富的天文资源服务信息。当然，为保证系统安全，系统必须提供安全认证的功能。

接口层(Interface Layer)，主要是服务引擎以及远程服务重定向模块，以及注册中心实现的各种接口，负责为 VO 服务提供一个统一的调用接口，同时提供高级用户的开发环境。客户程序也通过它来调用所有 VO 服务。当调用的服务是远程服务时，通过远程服务重定向器实现对远程服务的调用。而对于本地的服务，则可直接调用。服务的调用过程中，客户不需关注服务的位置，所有操作的执行都是系统自动完成的。服务调用的核心实现描述如下：

```
DynamicInvoker invoker=new DynamicInvoker(serviceURL); //唤醒远程服务
Vector method=MethodResolve.get(methodstr);     //获取指定的服务操作信息
String operation=(String)method.get(0);          //获取服务名
checkArgu argu=new checkArgu(gsh);
Vector   parameter=argu.getParameter(method);   //获取服务参数
Object para=new Object[i];
Setpara(para);                                   //设置服务参数
result=invoker.call(operation,para);        //执行服务操作，返回服务结果
```

汇集层（Collection Layer）是网格服务的信息中心，所有的服务及服务数据都在此进行汇集，客户通过调用 Index service 服务来获取所有的资源信息。服务的自动注册由 GT3 中的 RegistryPublishProvider 实现，当然我们也提供了主动动态注册与消毁的接口，而对于服务数据的聚集，其核心实现描述如下：

```
DataAggregationType DataAggregation = new    DataAggregationType ();
                                          //创建服务数据聚集类型
DataAggregation.setSource (sourcehandle);          //设置服务数据的源服务
DataAggregation.setParams (sub);                   //设置服务数据
subscriptionID = aggregator.addDataAggregation(dataAggregation); //聚集实现
```

最低层是资源层，是本地主机环境及本地网格服务的具体实现。

通过这样层次化的设计，实现了基于 GT3 的 IndexService 的资源与服务信息的管理及查询。

第六章 VO 系统节点资源的监控与负载平衡

前一章我们详述了 VO 中的各种资源信息的管理是通过注册中心来进行汇集与查询的，而资源的实体仍然是在资源提供者处，只要它愿意，加入到 VO 中，那么它就成为 VO 系统的一个结点，在这个结点中，可以提供各种符合 VO 规范的服务，因此，整个 VO 可以看成是由许多的结点组成的一个网络系统，同样我们也可以想象，在许多不同的结点上，很可能会有许多相同的服务，系统分配给用户哪个结点上的服务最合适，是资源管理与分配的一个关键问题，要完美的解决这个问题，我们就要知道服务所在结点的资源状态及负载情况，这就是资源监控的目的。

获取了服务结点的状态信息以后，那么怎样的服务分配才是合理的呢，显而易见，就是要实现负载均衡，使得系统中的每个结点的负载与此结点的承受能力相适应。

负载均衡并非传统意义上的“均衡”，一般来说，它只是把有可能拥塞于一个地方的负载交给多个地方分担。如果将其改称为“负载分担”，也许更好懂一些。说得通俗一点，负载均衡在网络中的作用就像轮流值日制度，把任务分给大家来完成。在 VO 中，负载均衡的意思就是对同一功能服务的请求合理地分配到具有这一服务的主机上，以最快的完成用户的请求。通过负载均衡，我们可解决网络拥塞，服务就近提供，实现地理位置无关性，对用户提供更好的访问质量，提高服务器响应速度，提高服务器及其他资源的利用效率，避免了服务结点的单点失效等问题。

要使系统能够稳定的运行，负载平衡是必不可少的一项工作，本章就详细描述了在 VO 环境下资源监控与负载平衡的实现。

6.1 节点状态的监控原理

在 China-VO，由于采用了 OGSA 的架构，每个节点也就是一个服务提供的主机环境，由于所有的资源都是以服务的方式发布，所以，资源监控这种功能我们也以服务的方式来实现----resourceMonitorService.

在 OGSI 的规范里，服务通过服务数据来维护服务的状态信息。并且这些服务数据可以定期的发送到 IndexService 中，进行统一的维护，用户通过查询这个服务数据，就可以了解所有跟此数据相关的服务的状态。

根据这个原理，我们在节点监控的实现上也采用了这种最基本的方式，resourceMonitorService 发布在每个服务结点上，同时，为此服务定义了一个名为 ResourceData 的服务数据，每个 resourceMonitorService 都与 ResourceData 数据相关联，在 ResourceData 中，定义了各种属性，来表达结点的资源状态，同时，我们也定义了一些基本操作，来更新 ResouceData 中的属性。其基本原理如下图所示

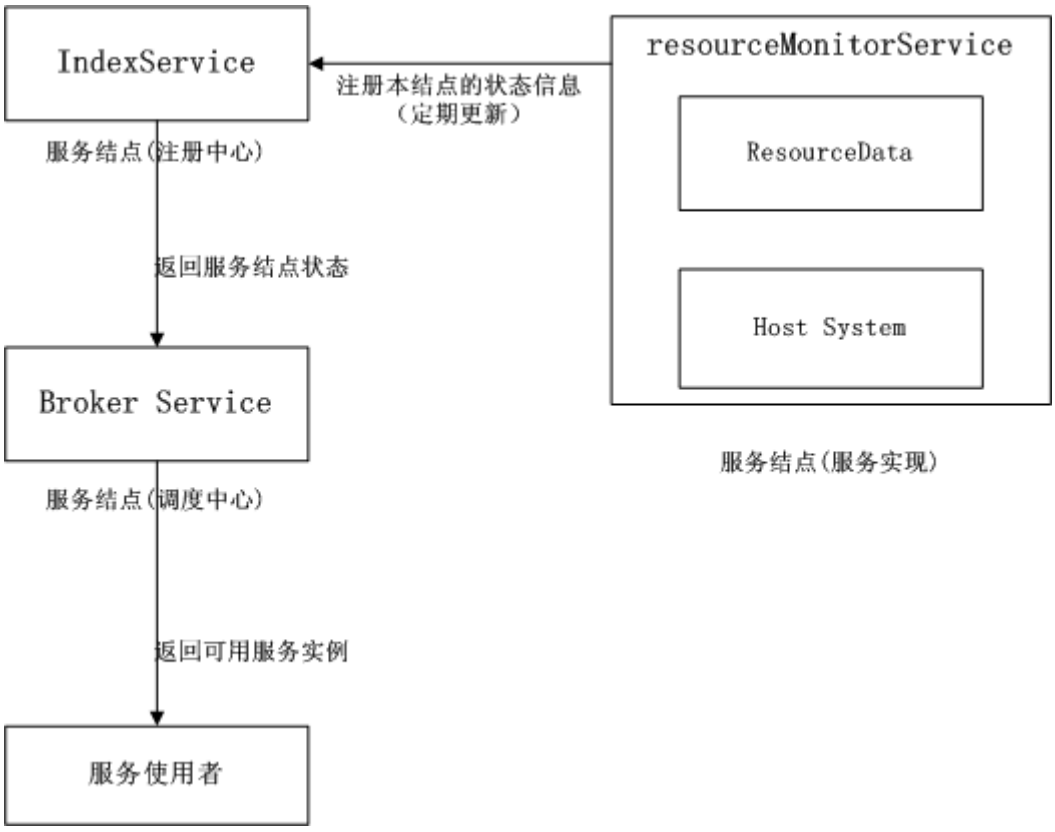


图 6-1：节点监控的实现模型

6.2　节点监控的实现

6.2.1 资源属性定义

目前，在 VO 系统中，我们主要关注的是服务的可用性，因此，我们主要从系统的最关键资源方面来衡量系统的负载情况，主要有三个方面：

一为 CPU 的使用率，主要包括 CPU 的类型，频率，负载等状况。

二为内存的使用状态，主要包括内存总容量，已消耗及可用空间，

三为磁盘空间的使用状况，主要包括磁盘总空间，已用与可用空间。

此外还包括主机地址，主机名等基本信息。

其 XML 示例显示如下：

<ResourceData
gt:originator="http://159.226.169.47:8080/ogsa/services/cwdsgrid/resourceMonitorService"
xsi:type="ns2:ResourceDataType" xmlns:gt="http://ogsa.globus.org/"
xmlns:ns2="http://www.cwds.org/namespaces/resourceMonitor/servicedata"
xmlns="" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
<ns2:location>159.226.169.47</ns2:location>
<ns2:hostname>www.china-vo.org</ns2:hostname>
<ns2:url>URL:</ns2:url>
<ns2:cpustate xsi:type="ns2:CpuStateType">
<ns2:type>Intel(R) Pentium(R) 4 CPU
1.60GHz</ns2:type>
<ns2:speed>1614.401</ns2:speed>
<ns2:load>0.010000000000000009</ns2:load>
<ns2:idle>0.99</ns2:idle>
</ns2:cpustate>
<ns2:memorystate xsi:type="ns2:MemoryStateType">
<ns2:capacity>1.055338496E9</ns2:capacity>
<ns2:used>1.04630272E9</ns2:used>
<ns2:free>0.0</ns2:free>
<ns2:ratio>0.0</ns2:ratio>
</ns2:memorystate>
<ns2:diskstate xsi:type="ns2:DiskStateType">
<ns2:capacity>6048320</ns2:capacity>
<ns2:used>3923212</ns2:used>

<ns2:available>1817868</ns2:available>

<ns2:ratio>0.69</ns2:ratio>

</ns2:diskstate>

</ResourceData>

6.2.2 监控服务的实现

目前，我们的服务结点可以是 linux 机器，也可以是 windows 机器。在不同类型的机器上，我们采用了不同的策略。其主要差别体现在获取具体的资源状态上。其实现静态类图如下：

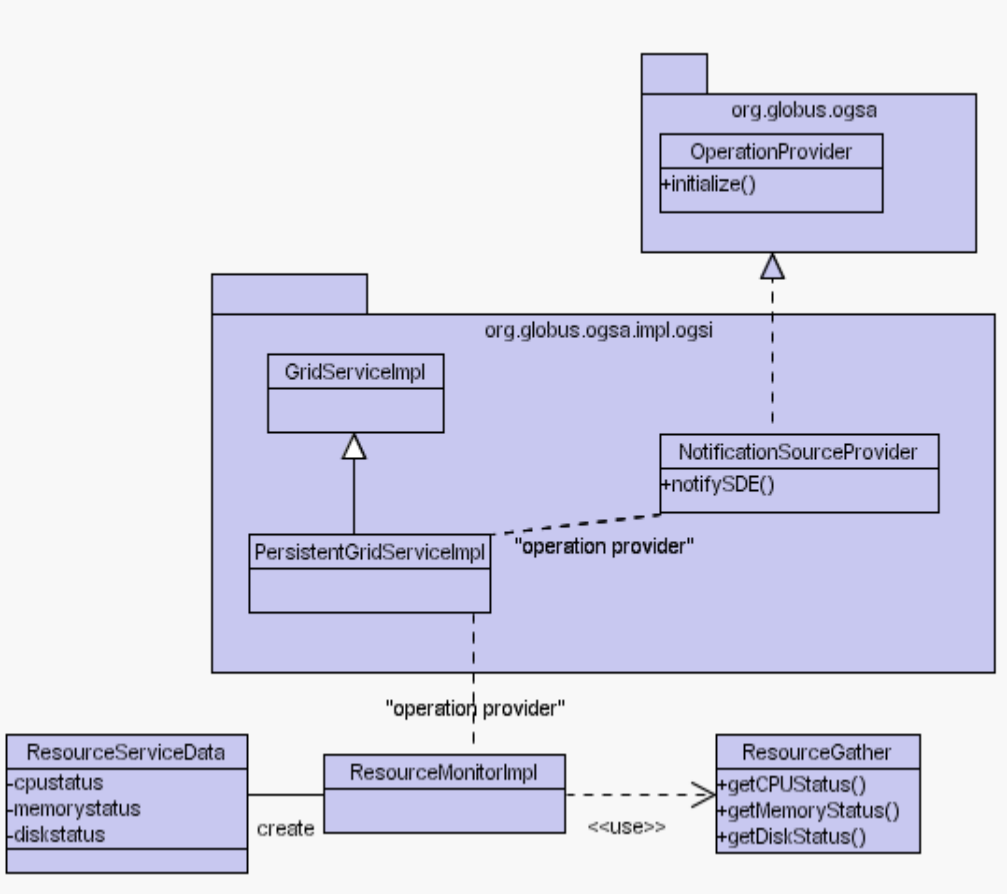


图 6-2：节点监控服务的设计类图

在类 ResourceServiceData 中，记录了此 monitor 服务所在主机的状态，而在 ResourceGather 类中定义了获取主机状态的几个基本操作.

6.2.3 资源状态的更新

服务主机的资源状态是动态变化的，因此，必须实时的对状态信息进行刷新，才能保证此信息的有效性，在状态的更新方面，主要有两个过程。

1 监控服务如何获取到最新的状态信息

2 监控服务如何将变化的状态信息发布到注册中心(IndexService)

对于第一个过程，我们采用定期执行的方法，利用线程技术，定期执行相关的操作，而对于第二个问题，我们采用了预定/通知机制。这样，当服务数据 ResourceData 发生变化时，它就能自动发布到 IndexService。预定通知机制的基本过程如下：

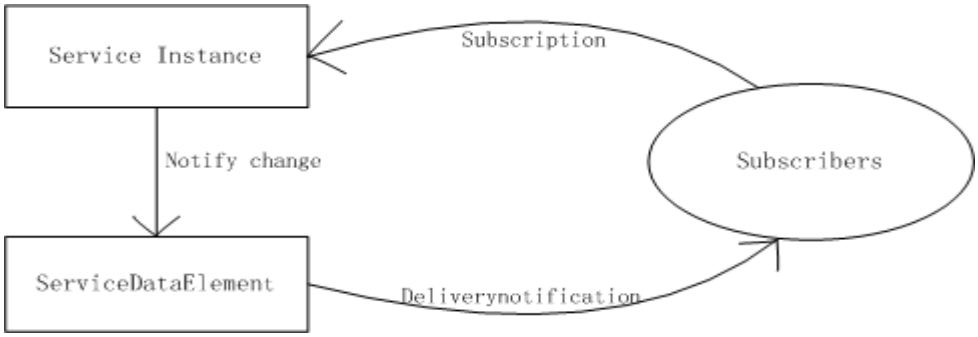


图 6-3：服务数据的预定/通知原理

6.3 负载平衡的实现

在资源监控实现以后，接下来就要考虑负载平衡的问题，也就是如何分配一个最优的服务给用户，因此，关键就是要确定判定最优的算法。

6.3.1 负载均衡的算法设计

负载均衡的主要要素是节点的资源使用状态。所以及时、准确的把握节点负载状况，并根据各个节点当前的资源使用状态动态调整下一次的服务分配，是负载均衡算法考虑的一个关键问题。

前面我们详细的讨论了资源状态信息的收集，在我们的负载均衡算法里，主要考虑以下因素：

- CPU 利用率 CPU_i %
- 内存利用率 MEM_i %

◆ 可用磁盘空间 Di

◆ 服务的实例个数C_i

如何综合评价节点的负载情况，我们采用了加权平均的方法。我们定义一个整数来表示系统的负载情况，表示为 LOAD(Ni).它的值主要由上述因素来决定。不同的因素根据它的权重来决定它的值。基本计算公式如下：

LOAD-机器性能=(CPU.Speed * CPU_i % + MEM.Capacity* MEM_i %)*(Di>100M)

LOAD-网络性能=C_i

LOAD(Ni)= LOAD-机器性能/ LOAD-网络性能

选择 LOAD(Ni)最大值的服务节点分配出去。

6.3.2 基于负载均衡的服务查询与分配策略

负载均衡的主要应用是在服务的分配上，当 VO 系统中存在有多个同一功能，但分布在不同的结点上的服务时，我们就要运用均衡算法来选择把哪个结点上的服务分配给用户使用，这主要由系统的资源调度服务(resourceBrokerService)来实现.

资源调度服务的主要功能之一就是为用户分配合适的服务实例。它的基本过程可描述如下：

1. 用户请求，通过提供服务名告诉系统需要什么服务。

2. 资源调度服务收到请求后，查询注册中心,找到所有满足此要求的服务。以及现有的服务实例数。

3. 查询注册中心的 Resource Data 数据，获取当前所有服务结点的资源状态。

4. 由服务找对应的资源，计算出服务主机的负载值，找出负载值最大的一个结点。

5. 创建服务实例，返回服务句柄给用户。

具体的实现，如以下类图所示：

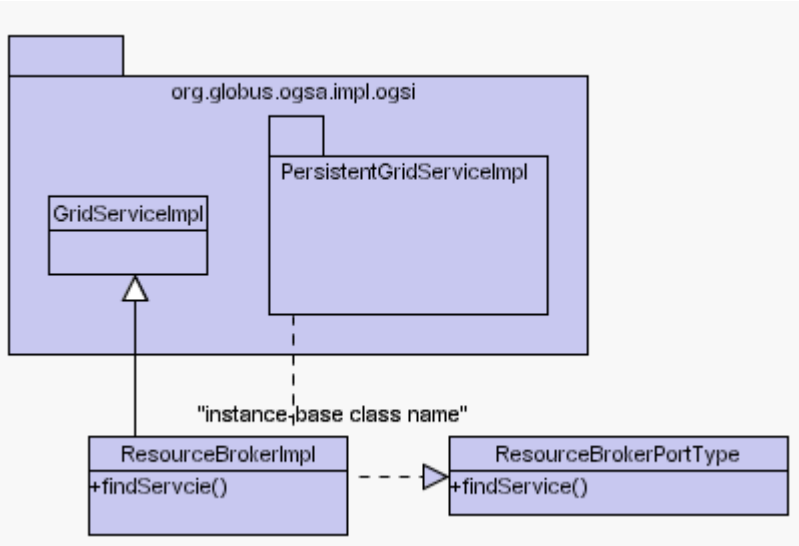


图 6-4：服务分配的设计类图

6.3.3 讨论

采用轮询策略来实现资源的动态监控，同时，设计了一种简易算法来实现资源的负载平衡，从而能够为用户分配一个最优的服务实例。但目前，在负载平衡上考虑的因素还不是很全面，算法也有许多待改进的地方。影响负载平衡的因素倒底占多大的比例，确实很难衡量，它没有固定的指标，而是与具体的需求有关。上述算法在某种程序上保证了服务实例的可用性，但是不是完全最优的，还有待于进一步的分析与检验。

结束语

前面我们详细地阐述了中国虚拟天文台在资源管理方面的建设情况，在 GT3 平台的基础上，我们初步实现了资源的服务化，资源的查找与监控，资源与服务的分配，以及服务的调用等基本功能。一个比较完整的资源管理平台已初现雏形。但我们也不难看出，它离实际的应用还有很大一段距离，主要存在以下一些问题：

1）建立在 OGSA 架构上的虚拟天文台，由于是以网格技术为基础，就必然会受到网格本身技术发展的限制。由于目前网格自身还处在研究与发展阶段，没有完全可用的技术平台，导致其上的应用开发也处在一个不稳定的基础之上，比如，目前我们的开发是建立的 GT3 的基础之上的，但现在，GT4 已经出现，而且架构完全不同，如果系统又要移植到 GT4 上，势必导致原有工作的浪费及影响 VO 开发的进展.

2）目前的服务管理都是针对单个服务的，例如，服务的调用，我们仍然只能一次次的调用服务实例中的功能接口，来实现我们的完整功能。而通常，一个完整的功能是由一系列服务，按照一定的顺序执行来完成的。为此，我们要考虑的就是如何能以一种自动的手段，让用户以一种简易的方法来自由组合自己的服务序列，系统在完成这一系列的服务之后，返回最终的结果给用户。这个过程，我们称它为工作流管理。

工作流的概念起源于生产制造业与办公自动化领域。工作流是一类能够完全或部分自动执行的经营过程，根据一系列过程规则，文档、信息或任务在不同的执行者之间传递、执行。工作流的目的是通过将工作分解成定义良好的任务、角色，按照一定的规则和过程来执行这些任务并对它们进行监控，达到提高办事效率、降低生产成本，提高企业生产经营管理水平和企业竞争力，实现现代企业经营过程重组（BRP）、经营过程自动化。

中国虚拟天文台中的工作流，实际上是一种服务流，它以用户要求的方式，顺序的执行相应的服务，因此，实现工作流的前提就是用户如何来描述自己的要求。也就是服务流过程描述语言。目前，我们只是初步地定义了服务流描述语言，示例如下：

<serviceflow>

<service name="">

<operation name="">

<argu>3</argu>

</operation>

</service>

</serviceflow>

<serviceflow>标记说明一个服务流过程的开始，<service>标记标识一个服务的开始，<operation>说明执行一个服务操作，<argu>表示一个参数，当此操作有多个参数时，就得要有多个<argu>标记.

在此描述语言的基础上，接下来要做的事情就是如何分解服务操作，以实现并行化，最后，如何融汇结果等是我们下一步要实现的目标.

3）由于虚拟天文台的设计与科学需求描述处在不断完善的阶段，相应的，将引起资源管理系统的不断变化，同时，资源管理系统本身仍然存在许多不完善的地方，比如，服务化的方式，负载平衡算法等技术本身都处在研究与优化阶段，很难有一种完美的实现，只能在不断的实践过程中找到最优的方式。

参考文献

[1] 赵永恒. 互联网时代的天文学革命: 虚拟天文台.科学, 2002, 54(2):13

[2] [LAMOST] Large Sky Area Multi-Object Fiber Spectroscopic Telescope. http://www.lamost.org

[3] [LSST] Large-aperture Synoptic Survey Telescope. http://www.lssto.org/lsst_home.html

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

[5] [Garching2002] Toward an International Virtual Observatory: Scientific Motivation, Roadmap for Development and Current Status.http://www.eso.org/gen-fac/meetings/vo2002/

[6] [IVOA] International Virtual Observatory Alliance. http://www.ivoa.net

[7] 刘鹏. 网格概念的界定.http://grid.cs.tsinghua.edu.cn/ grid/paperppt/ GridConcept.pdf

[8] 李国杰. 信息服务网格—第三代Internet. 计算机世界, 2001 (40): B8-B10

[9] Ian Foster, Carl Kesselman, Jeffrey M. Nick, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf

[10] Ian Foster, Carl Kesselman. The Grid: Blueprint for a New Computing Infrastructure. http://www.amazon.com/exec/obidos/ASIN/1558604758/o/qid=958665349/sr=2-1/103-6896860-5655839

[11] Ian Foster. WHAT IS THE GRID? A THREE POINT CHECKLIST. http://www.gridtoday.com/02/0722/100136.html

[12] Ian Foster, Carl Kesselman, Steven Tuecke <<The Anatomy of the Grid,Enabling Scalable Virtual Organizations>>

[13] Ian Foster. The Globus Toolkit: The Open Source Solution for Grid Computing http://www.globusworld.org/globusworld_web/plenary/IanFosterKeynote.ppt

[14] I. Foster, D. Gannon. The Open Grid Services Architecture Platform.

http://www.gridforum.org/Meetings/ggf7/drafts/draft-ggf-ogsa-platform-2.pdf

[15] [Globus] Globus Project. http://www.globus.org

[16] [GT] Globus Toolkit. http://www.globus.org/toolkit/

[17] The Globus Project. The Globus Toolkit v3 Developers and Administrators Tutorial. Session 1: Overview.

[18].<<Open Grid Services Infrastructure(OGSI)>>　http://www.ggf.org/ogsi-wg

[19].<<The Globus Toolkit 3 Programmer's Tutorial>> http://www.globus.org

[20]. WSRF_overview-1-0.pdf,http://www.globus.org

[21].崔辰州　博士论文：中国虚拟天文台系统设计 2002,6

[22]. RM-20040426.pdf,http://www.ivoa.org

[23] [RDF] Resource Description Framework. http://www.w3.org/RDF

[24] Natalya F. Noy, Deborah L. McGuinness. Ontology Development 101: A Guide to Creating Your First Ontology. http://www.ksl.stanford.edu/people

/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html

[25]. UCD-20041026.pdf,http://www.ivoa.org

[26][OAI] The Open Archives Initiative Protocol for Metadata Harvesting.

http://www.openarchives.org/OAI/openarchivesprotocol.html

[27] Ray Plante. An Evaluation of the Open Archives Initiative for VO Registries.

[28]VO Resource Identifiers. http://www.ivoa.net/forum/registry/

[29] [Registry] IVOA Resource Registry.

http://www.ivoa.net/twiki/bin/view/IVOA/IvoaResReg

[30] Ray Plante. Registry Requirements.

http://www.ivoa.net/forum/registry/att-0009/01-registryreq.txt

[31]AstroGrid Registry

http://wiki.astrogrid.org/bin/view/Astrogrid/AgCd06Registry

[32]《一种动态网络负载平衡集群的实践方法》 绿盟月刊 2003-03

致 谢

这本论文的完成凝聚着很多人的关怀和帮助。内心的感激之情是难以用语言表达的。

首先，我要感谢我的导师李廉教授，李老师知识渊博，宽厚待人，治学严谨。在硕士三年的时间里，李老师一直给予我谆谆教诲，引导着我走近科研，培养我开拓性的思维，走向事业和人生的成熟。在这里我要用最真诚的情感对李老师说句最朴实的话：“老师，谢谢！”

我能参加虚拟天文台项目工作要感谢国家天文台的赵永恒研究员和崔辰州博士所给予的机会。同时他们也教了我很多天文方面的知识和技术。指导我从事科研的方法，在此深深的道声谢谢！

感谢兰大高性能计算中心的周庆国老师，不管在兰大，还是在北京，周老师一直关心着我的学习与生活，帮助我解决各种困难与问题。

感谢兰大信息学院关心与教育过我的所有的老师，你们的治学态度与奉献精神都是我学习的榜样。

感谢同我一起从事虚拟天文台研究的同学们。虽然我们来自不同的学校，有着不同的年龄，但彼此默契的合作中结下了深厚的友谊。忘不了我们一起讨论的日子，感谢你们提出的各种想法与建议。

感谢兰大伴我一起度过快乐学习生活的兄弟姐妹们，你们给了我很多的鼓励和建议，尤其是我们实验室的师兄、师姐、师弟、师妹们。这份份友情是我今生宝贵的财富。

感谢天文台实验楼 324 室一起学习，生活的兄弟姐妹们给我的帮助与启发。

感谢清华大学的刘鹏博士及与我一起学习，研究网格的网格理论学习小组的同学们，感谢你们对我提出的建议与指导。

最后仅将此论文献给我深爱的家人和女友。他们给予我无私的亲情和爱，给我加倍努力的动力。想起父母为我求学之路付出的汗水，不禁一阵阵的酸楚。他们的爱已经凝聚成了我生命中的一部分。没有他们时时的牵挂，也就没有我今天的成就。