摘 要

网格入口软件是指对网格资源和服务的内容进行整合显示的一种 Web 应用。网格入口软件屏蔽了网格资源复杂的内部细节，使用户能通过一个熟悉的用户界面、一致的操作方式和高效方便的访问机制来使用网格系统和获取网格服务，从而解决网格系统资源使用复杂的问题。网格监控为网格系统中其他网格中间件提供与资源有关的重要数据，是网格系统性能调整和错误发现的依据，是保证任务顺利完成的重要支撑，而任务监控是网格监控中不可缺少的一部分。任务调度是关系网格是否能高效使用资源、快速完成任务的关键构件。

本文通过分析网格的使用模式，设计实现了网格入口软件－WebGrid。WebGrid 的研究与实现是围绕网格监控和任务调度展开的，通过研究已有的网格监控机制和任务调度机制，针对任务监控研究比较薄弱的现状，提出了任务运行期监控的概念，结合已有的分布式技术，开发了自主的网格监控系统；采用了基于遗传算法的任务分配算法，该算法采用资源-任务的间接编码方式，通过 DAG 图获取子任务的层次关系，并将子任务按照层次深度排序，解决了种群的非法问题。在单一资源上采用 Globus 的任务调度策略。在 WebGrid 用户安全管理中采用了 MyProxy 机制，解决了 GSI 安全机制和 web 安全协议的不一致问题，促使了两者之间的平滑结合。WebGrid 采用基于 Web 的浏览方式，融合了 Web 的功能和并行计算技术，提供给用户图形化的界面及方便、易用的操作环境。

关键词：网格；网格监控；任务分配；遗传算法；网格入口软件；开发网格服务架构;有向无环图

Abstract

Grid Portal is a web based application that providing content aggregation from different grid resources and grid services. Grid portal offers a better user interface, consistent accessing pattern and easy usage of the grid services, and solves the complexity in using grid computing resources. Essential and source-related performance data for middleware's of grid is provided by grid monitor, it lays the base for grid system to regulate performance and find errors. Grid monitor is an important foundation for the complete of grid job and job monitoring is an indivisibility part of grid monitoring. Task scheduling is an important component of grid to utilize grid resources fully and complete computing tasks rapidly.

The use mode of grid is defined in the dissertation, a grid portal-WebGrid, is designed and implemented. The research and implemented of WebGrid are based on grid monitoring and task assignment, by research existing grid monitoring and task assignment mechanism, job running monitoring mechanism is adopted in WebGrid, by combine existing distribute technique, a self-determination grid monitoring system is developed, and a task assignment strategy based on GA (genetic algorithm) is addressed. The chromosome-coding method and the operator of genetic algorithm are discussed in detail. The relationship between sub-tasks can be obtained through the DAG, and then the subtasks are ranked according to their depth-value, which can avoid the emergence of invalidate chromosomes. In the single resource, Globus's principles are adopted to determine the sequence of the subtasks. MyProxy mechanism is introduced in security manage module of WebGrid, the disagreement of Web security protocols and Globus's grid security infrastructure is resolved. WebGrid adopt browse mode based on Web, integrated the function of Web and parallel computing.

Key Words: grid; grid monitoring; GA; task assignment; grid portal; OGSA, DAG

西北大学学位论文知识产权声明书

本人完全了解学校有关保护知识产权的规定，即：研究生在校攻读学位期间论文工作的知识产权单位属于西北大学。学校有权保留并向国家有关部门或机构送交论文的复印件和电子版。本人允许论文被查阅和借阅。学校可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或扫描等复制手段保存和汇编本学位论文。同时，本人保证，毕业后结合学位论文研究课题再撰写的文章一律注明作者单位为西北大学。

保密论文待解密后适用本声明。

学位论文作者签名：张云锋 指导教师签名：

2005 年 6月10日　　2005 年 6月11日

西北大学学位论文独创性声明

本人声明：所呈交的学位论文是本人在导师指导下进行的研究工作及取得的研究成果。据我所知，除了文中特别加以标注和致谢的地方外，本论文不包含其他人已经发表或撰写过的研究成果，也不包含为获得西北大学或其它教育机构的学位或证书而使用过的材料。与我一同工作的同志对本研究所做的任何贡献均已在论文中作了明确的说明并表示谢意。

学位论文作者签名：张云锋

2005年 6月10日

未经作者、导师同意
勿全文公布

第一章 绪论

1.1 网格概述

1.1.1 网格概念

网格（Grid）概念产生于90 年代中期，是从电力网（Power Grid）概念借鉴过来的。网格的最终目的，是希望大家能够象使用电力一样方便地使用分布在网络上强大而丰富的计算能力。在《网格：一种新的计算基础设施蓝图》一书中，Ian Foster和Carl Kesselman 就尝试着给网格下定义：一个计算网格是一个硬件和软件基础设施，此基础设施提供对高端计算能力可靠的、一致的、普遍的和不昂贵访问[1]。

然而在2000年后，Ian Foster又指出网格计算关心的是：在动态的，多机构的虚拟组织中协调资源共享和协同解决问题。其核心概念是：在一组参与节点（资源提供者和消费者）中协商资源共享管理的能力，利用协商得到的资源池共同解决一些问题。

1.1.2 网格体系结构

网格体系结构是关于如何建造网格的艺术和科学方法。它给出了网格的基本组成与功能，描绘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机制。

到目前为止，最为重要的网格体系结构有三个：第一个是 Foster 等很早提出的五层沙漏结构；第二个以 IBM 为代表的工业界的影响下，在考虑到 Web 技术的发展和影响后，Forster 等结合 Web Service 提出的开放网格体系结构（Open Grid Services Architecture，OGSA）；第三个是在 OGSA 基础上进一步发展的 Web 服务资源框架（Web Service Resource Framework，WSRF）。下面我们主要介绍前两种体系结构。

1. 五层沙漏模型

五层沙漏体系结构[2]是一种影响十分广泛的结构，它的主要特点就是简单，主要侧重于定性的描述而不是具体的协议定义，因此很容易从整体上进行理解。其基本思想是以协议为中心，强调服务、API 和 SDK 的重要性。

五层结构的一个重要特点就是沙漏形状，其含义就是各部分协议的数量是不同的，对于其核心的部分，要能够实现上层各种协议向核心协议的映射，同时核心协议向下层协议的映射，因此核心协议是整个结构中的瓶颈，数量不应该太多，资源与连接层共同组成这一核心的瓶颈部分，如图 1-1 所示：

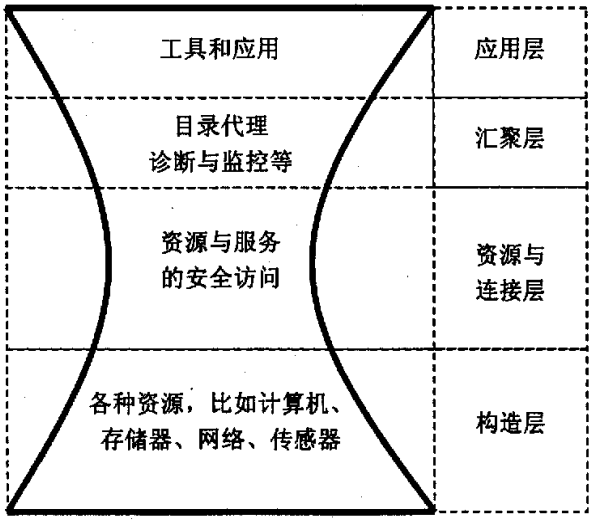

图 1-1 沙漏形状的五层结构

- **构造层**：基本功能是控制局部的资源，向上提供访问资源的接口；构造层的资源可以是计算资源、存储系统、数据库、网络资源和传感器等。构造层应该实现的功能是：查询机制（发现资源的结构和状态等信息）、控制服务质量等。
- **连接层**：基本功能是实现安全的相互通信，定义了核心的通信和认证协议，用于网络之间的事务处理。
- **资源层**：主要功能是实现对单个资源的共享，它建立在连接层的通信和认证协议之上，定义了安全初始化、监视、控制单个资源的共享操作、审计以及付费等。
- **汇聚层**：主要功能是协调多种资源的共享；汇聚层组件是为了满足虚拟组织或者应用领域的需求，所以必须提供通用的、更高级协议、API

和 SDK。

- 应用层：存在于虚拟组织环境中，根据任一层上定义的服务来构造应用。

2. 开放网格服务架构 OGSA

开放网格服务架构 OGSA 是网格标准委员会（Global Grid Form，GGF）的重要标准，是一种新的网格体系结构。OGSA 允许在异地的或不同的虚拟组织之间——无论是在单独的企业还是在共享资源和提供服务的几个大的公司之间，传送综合服务和资源[3]。

在 OGSA 里，我们主要关注于服务：计算资源、存储资源、网络、程序、数据库以及其他类似资源都表示为服务。一个面向服务的观点使得我们面临如下一些需求：标准接口定义机制、本地/远程透明性、对本地操作系统服务的适应以及统一的服务语义。一个面向服务的观点还简化了虚拟化——即将不同实现封装成一个通用接口。虚拟化允许跨多个异种平台对资源进行一致地访问，并提供本地或远程定位透明性，还可以将多个逻辑资源实例映射成同一物理资源，以及在一个 VO 内基于更低级的资源组合进行资源管理。虚拟化允许服务组合，形成更复杂的服务——无论正在被组合的服务是如何实现的。网格服务的虚拟化也支持将通用服务语义行为无缝地映射到本地平台设施的能力[4]。

1.1.3 网格平台软件-Globus

Globus 项目是由美国 Argonne 国家实验室主持的国际上最具影响的网格计算项目[4]，它发起于 20 世纪 90 年代中期，其最初的目的是希望把美国境内的各个高性能计算机中心通过高性能网络连接起来，方便美国的大学和研究机构能够使用，提高高性能计算机的使用效率。随着对 Globus 项目的深入研究，针对它的目标也进一步扩展，希望通过 Globus 项目可方便对地理上分布的研究人员建立虚拟组织，进行跨学科的虚拟合作。目前，Globus 项目把在商业领域中 Web Service 技术融入其内，希望不仅仅局限于科学计算领域，还能够对各种商业应用进行广泛的、基础性的网格环境支持，实现更方便的信息共享和互操作，从而对商业模式、人员的工作方式和生活方式产生深远的影响[2]。

根据 Globus 的观点，在网格计算环境中，所有可共享的主体都是资源，Globus

关心的不是资源本身，而是如何把资源安全、有效、方便地提供给用户使用。Globus 通过对资源管理、安全、信息服务及数据管理等网格计算所涉及的基础理论和关键技术进行研究，开发出了能在各种硬件平台上运行的网格计算软件工具包（Globus Toolkit）。该软件适用于规划和组建各类网格试验平台，已在多个网格项目中得到应用，例如 NASA 网格(NASA IPG)、欧洲数据网格（Data Grid）、美国国家技术网格(NTG)等。Globus 已经成为国际上通用的网格平台软件[5]。

1.2 研究背景

随着高性能应用需求的迅猛发展，单台高性能计算机已经不能胜任一些超大规模应用问题的解决。这就需要网格技术将地理上分布，系统异构的多种计算资源连接起来，共同解决大型应用问题。在网格环境中存在各种动态资源，他们在地理上分散，又可以动态的加入或离开不同的虚拟组织。如何使网格应用程序方便地使用各种资源是必须解决的问题。

网格系统从结构上来说是基于广域网的分布式异构系统，从应用模式上来说也是分布式异构的并行应用模式，这就导致网格系统内部结构复杂，使用也很复杂。而现在使用网格的用户大多数不再是计算机专业人员，他们对于分布式技术、并行计算技术等网格的细节不熟悉，这就导致用户在使用和获取网格系统资源时面临巨大的困难，这也是制约网格系统获得广泛应用的主要原因之一[6][7]。

网格入口软件克服了用户直接使用网格资源的困难，对用户屏蔽网格资源复杂的内部细节，使得用户能通过一个熟悉的用户界面，一致的操作方式和高效方便的访问机制来使用和获取网格服务，从而解决网格系统使用复杂的问题。但是网格入口软件需要用户授权服务器以用户身份去执行任务，用来发起用户对资源的操作，而在网格环境下，资源被 GSI 所保护，所以 Globus 的构件成为网格事实上的标准。GSI 支持这种授权，而标准的 Web 安全协议并不支持。这样就在网格安全和 web 安全协议之间产生了不一致[8]。

但是现有的网格入口软件大多不是基于 Web 的，远程用户使用不方便，在网格作业管理，安全管理等方面做得不好，尤其是在网格监控方面暴露出诸多不足，如：监控内容有限，监控内容主要是反映主机和网络性能状态的信息如负载等，缺乏对用户任务的监控[9]。

1.3 研究目的及现状

1.3.1 研究目的

本文的研究目的是研究开发一个基于 Web 的网格入口软件，在入口软件用户安全管理部分采用了 MyProxy 机制，解决了 GSI 安全机制和 web 安全协议之间的不一致，使两者之间平滑结合；并在网格入口软件中增加了作业管理和资源管理，针对网格监控信息不足的情况下提出了任务运行期监控的概念，提出了 MDS 结合的网格监控方法和自主开发的网格监控方法，增加了对网格任务执行过程的实时监控，在任务调度中采用了基于遗传算法的任务调度策略代替了 Globus 的先来先服务任务调度策略。该网格入口软件监控内容丰富，对用户屏蔽了网格资源复杂的内部细节，使远程用户能通过一个基于 Web 的界面，一致的操作方式和高效方便的访问机制来使用网格系统和获取网格服务，克服了用户直接使用网格资源的困难。

1.3.2 研究现状

当前网格入口软件工具主要有：国外的有美国先进计算基础设施全国联盟（NPACI）的 HotPage；Argonne 的 COG，东京工业大学开发的的 JiPANG，美国 Mississippi 州立大学开发的 MCWP，加州大学伯克利分校开发的 Grid Portal Development Kit，美国得克萨斯州大学（TACC）开发的 GridPort，欧盟 GridLab 项目下的 Gridsphere，NMI 开发的 OGCE 等。

当前对网格任务监控的研究还处于起步阶段，还没有形成一个公认的行之有效的解决办法。一些网格系统对任务监控进行了尝试，在一定程度上能实现部分功能：Globus 对任务监控的支持不够，缺乏对任务状态及运行情况的报告，缺乏友好的客户端。在 Legion 中，提交任务的 Legion 工具可监测任务对象，但是它们返回给用户的信息很有限。只有当用户指定特定的任务运行在特定的主机上，并且用户正好拥有那台机器的帐号，则用户可以登录到那台机器，使用传统的系统可提供的工具来监控任务的执行过程。如果用户提交任务，但不指定该任务在哪台机器上执行，则用户不能获得任何与该任务有关的信息，这不符合网格的思想。Nimrod 系统提供了图形工具能帮助用户查看任务的执行过程，但该系

统没有提供方法以访问中间文件或者提供传统意义上的输入文件。欧洲Data Grid的R-GMA完全可替代MDS的GIIS，使用灵活，它开发的L&B(Logging and Bookkeeping)为用户提供了对任务的有效管理服务。L&B提供的服务包括三方面：（1）Logging，是为任务调度系统存储长期信息的服务；（2）Bookkeeping，是为当前活动任务存储短期信息的网格服务；（3）Accounting，记帐，提供对资源消耗统计的网格服务[10]。

以上网格系统除Data Grid的L&B外都没有提供一种机制获取任务的资源消耗信息。除了Nimrod外，都没有综合的任务监控工具。L&B为任务提交与监控服务之间提供了桥梁，但它还不是针对用户的监控系统而是资源管理系统的子系统，用户可直接获得的信息还不能完全满足用户的要求，如对任务进行实时的监控。总之，目前的网格系统对任务监控的支持非常薄弱，在这方面的研究有着广阔的空间。

当前对网格任务调度的研究有：Buyya提出了一种基于应用经济模型的优化调度模型，其目的是在资源的拥有者和使用者之间建立一种“交易”，以尽可能低的费用满足资源使用者进行计算任务的最低要求；Vincenzo介绍了一种基于遗传算法的资源调度算法，其目的是为了尽可能地提高资源的使用率和吞吐量；另外Abraham等人介绍了模拟退火等进化算法在网格资源调度中的应用；Xu等人介绍了蚂蚁算法的应用。但是它们没有考虑到任务之间的依赖关系[12]。

1.4 论文的主要工作

论文的主要工作包括：

- 在分析网格使用模式的基础上，研究实现了一个基于Web的网格入口软件WebGrid。
- 在网格用户管理部分采用MyProxy机制，解决了GSI安全机制和web安全协议之间的不一致。
- 提出了任务运行期监控的概念，自主开发了的网格监控系统。
- 在系统中实现了基于遗传算法的任务分配策略。

1.5 论文的主要贡献

- 在网格安全管理部分中采用MyProxy机制，解决了GSI安全机制和web

安全协议之间的不一致。

- 提出现有网格环境下实现任务监控的方法，开发了网格监控系统。
- 在系统中实现了基于遗传算法的任务分配策略。

1.6 论文的组织

全文共分六章。

第一章 为网格研究绪论，重点介绍网格入口软件研究的背景、现状及我所研究的目的和意义。

第二章 介绍网格使用模式和网格入口软件研究现状。

第三章 介绍网格监控中的任务监控方法。

第四章 介绍基于遗传算法的网格任务分配方法研究。

第五章 介绍基于 Web 的网格入口软件的体系结构及关键技术。

第六章 介绍基于 Web 的网格入口软件的设计与实现。

第七章 总结与展望。

第二章 网格入口软件研究

网格系统从结构上来说是基于广域网的分布式异构系统，从应用模式上来说也是分布式异构的并行应用模式，这就导致网格系统内部结构复杂，使用也很复杂。未了克服用户直接使用网格资源的困难，需要在用户和网格系统之间开发新的中间件，以对用户屏蔽网格资源使用复杂的内部细节，使用户能通过一个熟悉的用户界面，一致的操作方式和高效方便的访问机制来使用网格系统和获取网格服务，从而解决网格系统使用复杂的问题，这样的中间件称为网格入口软件[6][7]。

2.1 网格的使用模式剖析

2.1.1 网格的使用模式的要求

作为构造虚拟组织系统和应用的平台，普遍存在的Web技术很有竞争力。这些技术能够出色的支持浏览器--服务器交互模式，是今天的WEB的基础.但是它们缺乏满足在虚拟组织中更复杂的互动所需要的能力。例如，现在的WEB浏览器并不支持单一签证或授权技术。如果能采用清晰的步骤来结合网格和web技术，那么能大大促进网格技术的发展。例如，单一签证技术扩展了GSI的能力，如果集成到浏览器的话，将实现对许多Web服务器的单一签证，GSI将赋予Web客户代理的能力。这些能力，使得用web技术来建立“虚拟入口”(对复杂的虚拟组织应用提供一个瘦客户接口)更加容易。

另外，作为网格发展中一个致命的问题是它的使用环境不是很友好，现在的许多网格应用大多集中在科研或者军事上，还没有运用到商业上来，这在很大程度上是由于其使用环境的不友好所致的，所以现在急需一种使网格使用环境变得友好和方便的技术，从而促进网格的商业化发展。而网格入口软件是实现这种愿望的最好技术，它是连接网格应用、网格用户和网格系统的有效桥梁。

2.1.2 网格使用环境的特点

相对传统的C/S和B/S结构来说，基于网格的使用环境有其自己的特点：

- 首先它是一种特殊的两层的 C/S 结构

和传统两层C/S结构不同的是，网格客户端中嵌入了网格中间件层—网格操作系统，其主要的作用是实现网格资源的统一分配和管理，从而实现网格的单一系统映象，所以网格客户端不再是传统客户端那样完成数据的逻辑处理和表示功能，而是一个网格的入口软件；网格服务器端也不是传统意义上的物理服务器，而是单一系统映象下的虚拟服务器，通常由若干个提供网格服务的设备在网格操作系统的协调管理下共同完成任务，克服了传统服务器单独提供服务的局限。

- 具有客户端可编程能力。

传统浏览器的主要功能是“浏览”，将服务器传送过来的页面进行解释和显示。由于网格操作系统能向应用层提供完善和灵活的系统调用接口，使得网格浏览器已经突破了“读”的限制，而且可以通过编程接口向网格操作系统发送控制、调度等命令，使用户具有了“主动”支配和监控应用服务的能力。

- 服务器端服务组合的能力。

传统的服务器端的服务是定制的，用户只能被动地使用服务器上编制好的服务，而不能根据自己的需要主动安排服务，而网格的服务器可以让用户使用服务定义的接口将所需的服务自由地进行组合调用，达到了更高层次的服务共享。

- 用户使用高度透明化。

使用传统web服务时，用户必须知道提供服务的服务器的域名，并且客户端与服务器端只能建立一对一的连接，而网格用户在使用网格服务时，只需通过指定的访问接口(系统调用或是系统命令)来指定服务的名称和类型和提交用户数据，由网格操作系统来透明的完成服务之间的分配和协调，并且可以在网格客户端和网格服务设备之间透明的建立一对多的连接关系。

- 支持“会话(session)”功能。

网格操作系统具有“会话”功能，可以将用户提交的任务状态保留在指定的文件中，从而突破了传统web服务器基于一次连接提供一次会话服务的“无状态记录”局限，从而使得网格用户具有更强的远程控制能力，可以基于一次会话使用多次连接，可以远程的部署、启动、查询、监控网格上任务的执行情况，也可以根据自己的需要等待或是不等待任务执行完毕，更好的提高了客户端的工作效

率。

- 减少了系统中的性能瓶颈。

嵌入了网格操作系统的网格使用模式突破了web应用的“五大瓶颈”，实现了网格上的单一系统映象。网格操作系统就象是网格的处理器一样，由于放在客户端，不但可以快速的响应客户请求，减少了通信时间、通信量和通信频度，也消除了传统web中由服务器来响应客户请求而造成的服务瓶颈。

- 提高了系统的整体性能。

由于网络上通信次数的减少，网络负载不会受到用户数增长的直接影响，而主要取决于网络数据传输量的大小，同时因为网格操作系统具有负载均衡的能力，可以按照服务负载的大小来分配网格服务设备，从而可以较好的保证网络与服务器端的性能。

- 提高了系统的应用扩展能力。

网格浏览器、网格操作系统和网格服务器之间的接口采用统一定义的访问格式编写，有利于用户或是应用程序开发员对应用范围进行扩展，也有利于网格管理员对软硬件设备进行扩充[8]。

2.2 网格入口软件功能

网格入口软件通过超链接、菜单选项、表单、按钮等手段，提供网格计算环境中的单点登录、作业提交、资源搜索和选择、数据传输和拷贝等，提供各种商业网格服务访问页面，与 web 工具相结合提供协同工作的问题解决环境。总之，网格入口软件是网格计算环境的重要组成部分，它使用户摆脱学习服务命令和编程接口等琐事，而集中精力解决与具体领域相关的问题。

网格入口软件在 Globus 的基础之上提供给用户更高层和更方便的服务：

- **安全服务**：网格入口软件的安全服务是基于 Globus 的 GSI 的。用户通过 Web 浏览器使用用户名/密码登录网格入口软件，此后，网格入口软件就扮演用户代理的角色和网格资源进行安全交互，因此，入口软件服务器必须获取用户的代理证书。
- **远程作业管理**：提交、执行和监控网格作业的能力是网格入口软件的基

本需求。Globus 的 GRAM 仅提供了基本的远程作业服务实现，需要开发高层的网格入口软件服务支持它。允许用户在分配的资源上能够看到作业队列情况和调度信息，能够跟踪作业执行和了解作业执行失败时的日志信息。当将大型的网格应用和复杂的工作流程执行系统作为网格服务进行部署时，网格入口软件必须能够提供对他们的访问工具[7]。

- **监控服务**：网格监控工具也是入口软件中的一个必要角色。每一个网格用户在网格上都应该拥有一个私有的、持久不变的信息存储空间，用来存储用户需要的重要信息。这些信息包括用户感兴趣的各种资源索引、资源的详细信息、用户提交作业的运行状态（挂起、正在运行、运行失败、错误信息）等等。基于 Globus 的 MDS，信息服务工具还可提供用户查询网格资源的各种动态信息，例如，某个网格结点的 CPU 空闲率[13]。

2.3 网格入口软件研究

网格入口软件大致可分为四类：

- 面向用户的入口软件。这类软件一般主要集中于简单的作业提交，作业追踪，文件管理和资源选择等。NPACI 的 HotPage 是其代表作，其他的还有西安交通大学开发的 WebCom 等等。
- 入口构件软件。这类软件为用户提供所需的 API 函数，用户通过该 API 函数实现与网格环境的交互。如 Argonne 的 COG, 东京工业大学的 JiPANG 等。
- 科学应用入口软件。科学应用入口软件是一个面向应用的使用环境，通过该环境用户可以编程实现其复杂的应用任务，并使用远程资源之行其任务。如美国 Mississippi 州立大学开发的 MCWP 等。
- 基于 web 的入口软件。如 Grid Portal Development Kit(GPDK), GridPort、Gridsphere, OGCE 等等。国内基于 Web 的网格入口软件主要有华南理工大学开发的通用网格平台，上海交通大学开发的上海网格等。

内容	WebCom	CoG	MCWP	HotPage	JiPANG
用户管理和认证	有	有	有	有	有
信息服务	较强	一般	一般	强	强
图形用户界面	完全	有	有	不完全	完全
文件传输	不完全支持	完全支持	不完全支持	不支持	不完全支持
远程任务提交方式	基于浏览器的图形界面	基于 Java 的 GUI 界面	基于浏览器的图形界面	基于浏览器的图形界面	基于 Java 的 GUI 界面
远程软件安装	不支持	支持	不支持	不支持	支持
作业调度	支持	支持	不支持	不支持	支持
通信安全	SSL,SSH	Java 客户端认证	SSL,SSH	Globus GSI 和 MyProxy	基于 Java2 的 Jini 机制
资源发现	无	无	有	无	有
用户定制	无	无	无	有	有

图 2-1 各种网格入口工具的性能比较[6]

（1）**GPDK**，是加州大学伯克利分校开发的网格入口软件开发工具包现在已经不被支持，GPDK 提供了一个实用网格的多层中间件，使用了 MVC 设计模式，从访问网格服务的逻辑中分离控制层和表示层。GPDK 用三个核心的组件来对应 MVC 视图。入口软件引擎（Portal Engine，PE），是 GPDK 中的总控制中心；使用 Java servlet 实现具体控制各模块页面请求的行为页面对象（AcTion Page Objects，APO）；用来显示结果的视图页（View Pages，VP）。APO 负责控制和封装请求对象来响应各种各样的操作。VP 负责把 APO 响应入口软件操作的结果以 HTML 的方式显示给用户具体的页面视图[14]。

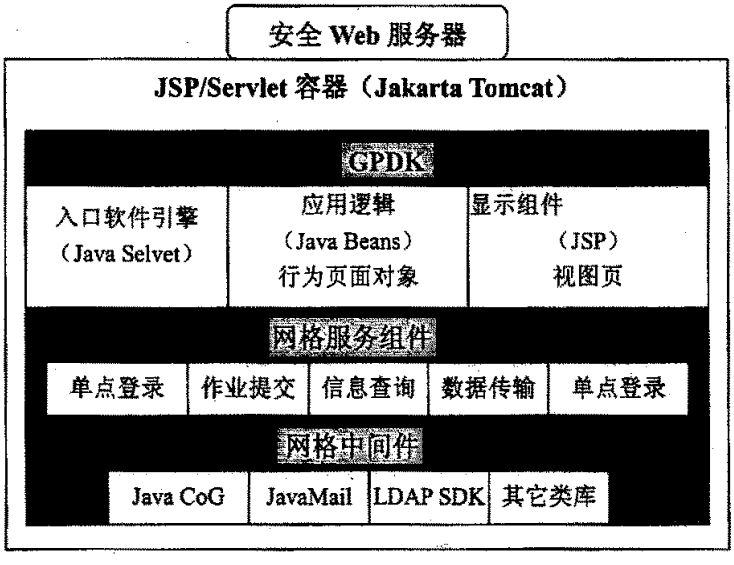

图 2-2 GPDK 体系结构

GPDK 寻求提供应用级入口软件和用户级入口软件能力，具有以下功能：

- GPDK 的核心设计成通用的、可复用的公共组件。通过 Java CoG 访问 Globus 提供的网格服务，并保持与新版 Globus 的兼容性。
- 为每个用户提供一个可定制的网格入口软件登录环境（Profile），用来保存用户访问网格入口软件的相关信息如历史作业提交情况、资源和应用的授权信息、以及其它的专业用户感兴趣的信息。Profile 易于扩展，可以根据应用网格入口软件的定制需求添加更多环境属性。
- 提供一个完全开放的开发环境，利用 GPDK 网格服务的核心组件，可以定制和开发各种网格入口软件服务。GPDK 还提供了可扩展库和一个模板入口软件，帮助开发人员迅速地开发和部署各种具体的网格应用和用户定制的入口软件服务。
- GPDK 尽量采用成熟的软件技术（Servlet、Javabean 等）、通用的接口协议（HTTP、LDAP 等）和开放的类库（Java CoG、JavaMail、Netscape LDAP SDK 等），使其与现有的网络应用和服务之间可以相互操作，并具有良好的可扩展性。

（2）**GridPort 3**，是 NPACI（NaTional Partnership for AdvancedComputaTional Infrastructure）开发的网格入口软件开发工具包。现在版本是 GridPort 3，基于 J2EE、DBMS 和 Java 等技术，改变了以前（GridPort2.x 之前）基于 Perl 语言设计的方法。GridPort 的体系结构框架是基于 J2EE 框架上建立起来的，支持数据库、安全（JAAS）和事务管理[15]。GridPort 是构建在 GT3 之上的，并且集成工作流系统如，GridAnt、Open GCE RunTime Engine（OGCE）、Pegasus 和 Condor DAG-Man。GridPort 3 提供了以下服务：

- 帐号创建和认证；
- 批处理作业提交；
- 命令执行；
- 作业序列（Job Sequsencer）；
- 文件和数据管理；
- 网格入口软件信息库（Grid Portal InformaTion Repository，GRIR）；
- 演示入口软件。

（3）**OGCE**：全称是开放网格计算环境（Open Grid CompuTing

Enviroments)，是 NMI 的一个网格入口软件原型。它是构建在 Jetspeed 入口软件框架上。 一个功能非常强大的入口软件，易于安装和配置。支持各种 Globus 版本，并且可以集成 GridPort 3 入口软件。但是，它的体系结构设计的过于凌乱和复杂，不利于在它上面开发具体的网格具体应用入口软件[16]。

（4）**Gridscape**：一个快速创建交互和动态网格实验床（Testbed）Web 入口软件的工具，是 GridBus 项目的一个产品。它有两个关键的独立组件（一个 web 应用和一个相关的管理工具），基于 MVC（Model-View-Controller）使用 Java 语言设计的[17]。它的设计目标是：

- 允许快速的创建网格实验床入口软件；
- 允许简单的入口软件操作和管理；
- 提供一个清晰的和用户友好的网格测试床资源全景视图；
- 灵活的设计和实现，例如，核心组件可以被重用，提供高层的可移植性。

第三章 网格监控中的任务监控研究

在以前的网格监控系统中，无论是 GMA 还是 Globus 的 MDS 都是针对网格资源的监控，而缺乏对网格任务的实时监控，而网格任务监控是网格系统进行性能调整和错误发现的依据，是保证任务顺利完成的重要支撑，是网格入口软件重要的一部分。任务监控应获取的信息可分为两大类：任务状态和任务运行情况。任务状态是指任务处于哪个阶段，是处于等待状态还是正在运行，是运行失败还是正常退出；任务运行情况是指任务正在运行时的有关情况，如运行进度、对资源的使用情况[18]。本章详细介绍了网格任务监控的方法，讨论了任务状态的定义，任务信息的收集，以及任务运行期监控的要解决的关键问题和解决途径。

3.1 网格监控概述

在网格支持下，用户综合使用互联网上的资源如同使用本地资源一样。用户通过入口软件提交任务，资源代理接受并为任务寻找与之相配的资源，最后任务被分配到相应的资源上完成。在这一过程中，网格监控起着重要的作用，是整个任务分配顺利完成的基础；一方面，要为计算任务找到合适的资源，这必须要能提供描述资源特征的性能数据；另一方面，网格及网络上的资源动态性大，这必须有一种实时反映网络及资源当前状态的机制；另外，错误的检测和发现机制也需要可靠的数据依据。

网格监控包括资源监控、任务监控、数据统计与性能分析、日志、异常报警。资源监控即获取特定资源的有关性能数据或了解当前状态。任务监控又称任务跟踪，是指任务从分配到执行完毕整个过程中的状态监测。当用户在客户端提出监控任务请求时便启动任务监控服务。它为当前的活动任务存储短期的信息。日志功能可针对用户不同要求来记载资源或任务的历史记录，并在此基础上进行数据统计和性能分析。异常报警可判断异常并做出反应，具体的异常情况视不同的网格应用系统而异[19]。

此外，随着网格向着商业化方向的推进，网格用户了解与其任务有关的信息显得越来越重要。用户期望能了解任务的当前状态、预测任务的完成时间。必要

时终止任务的执行、了解任务的资源消耗情况等。所以网格监控中的任务监控成为网格应用中的重要环节。任务监控系统应获取得信息可分为两大类：任务状态和任务运行情况。任务状态是指任务处于哪个阶段，是处于等待状态还是正在运行，是运行失败还是正常退出；任务运行情况是指任务正在运行时的有关情况，如运行进度、对资源的使用情况。

3.2 任务状态的监控

3.2.1 任务状态的定义

用户任务从开始提交后在整个生命周期内经过了如下组件：用户界面层、资源代理、分配服务和计算结点。任务监控服务负责收集整个过程中的重要事件数据并以可靠方式存储、一般情况下“事件”反映了任务状态的改变[20]。任务生命周期如 3-1 所示。

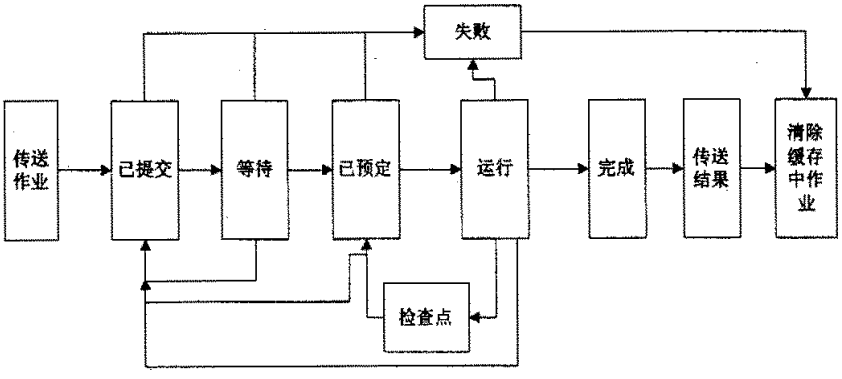

图 3-1 任务生命周期

从用户的视角出发，任务状态被定义为：

- 已提交－用户通过入口软件提交任务，入口软件寻找合适的资源代理，当找到后发布“提交”事件，此时任务处于“已提交”状态。
- 运行-任务正在运行。
- 等待-由于各种原因，任务在资源代理队列里等待。如：没有找到合适的资源。
- 已预定－任务在计算环境本地队列里等待。
- 完成－任务执行完成。

- 失败－任务执行失败。
- 检查点－由于一些原因，任务停在检查点并等待重新启动。
- 任务上传中－任务正在通过 GASS 服务传输作业到远端 GateKeeper 过程中。
- 任务结果返回中－任务正在通过 GASS 服务传输作业执行结果到本地过程中。
- 作业清除－作业执行完毕后，当用户得到所有由任务产生的输出文件后，将 GASS 缓存中的作业清除。

3.2.2 任务状态信息的收集

我们对作业状态的监控采用订阅－通知机制，当我们提交作业前，将为该作业增加作业状态监听器和作业输入输出流监听器，当作业状态改变至执行时，通知作业管理者，由作业管理者通知 GASS 服务器，将作业结果写到 GASS 缓存中，当作业状态改变至完成时，由 GASS 服务器将作业执行结果返回客户端，然后清除缓存中作业。

3.3 任务运行期的监控

3.3.1 要解决的关键问题

任务运行期的监控就是监控任务的各个进程对资源的占用情况，对任务运行情况的监控最终落实到对各主机上执行的进程的监控，在主机操作系统的支持下，可以很方便开发出相应的信息提供者。但是，如何将网格任务和网格资源节点上运行的进程建立起关联？这是解决任务监控问题的关键所在。具体为以下两个步骤：监控系统首先应找到执行任务的资源，然后在资源上找到任务对应的进程[10][18]。

3.3.2 问题的解决途径

1．监控服务端与执行任务的资源连接的建立

目前 Globus 的任务管理和资源管理模块都没有对任务状态报告的接口，所以当任务被提交后，监控客户端并不知道该任务被分配到了哪个资源节点。所以

要实现对任务运行情况的监控，就必须在监控服务端与资源的主监控器之间建立联系。我们可以通过目录服务来实现，本地资源管理者接收任务后向全局目录服务注册，任务监控服务通过查询全局目录服务找到相应的本地目录，从而与本地资源的主监控器建立连接。

2．任务与进程对应关系的建立

监控系统找到执行某一任务的资源后，依然不能顺利地通过监控进程来了解任务的执行情况。因为在资源上运行的进程有可能属于某个网格任务，也有可能属于其他网格任务，还有可能属于本地用户的非网格任务。所以要实现对任务运行情况的监控，就必须将网格任务与其进程建立起对应关系。

目前已有的网格资源管理系统都是按照以下机制来实现任务分配的：当用户提交任务时，网格资源管理系统收到一任务清单，该清单是对任务的描述和所需资源的说明。通过认证与授权后，网格资源管理系统解析该任务清单，转换为能被本地资源管理者理解的形式，从而在一个本地用户下启动任务的运行。这个本地用户与网格用户没有必然的联系，因为在网格里，不需要为网格在每台执行网格任务的主机上建立用户，所有的网格用户都有可能被映射到同一本地用户。这样，资源显然不知道正在为哪个用户服务，监控系统无法知道某一进程是属于哪个网格服务[18]。

我们可以通过以下方法来解决：

当资源管理者接受任务时，为该任务在本地创建一个临时用户。该临时用户名与任务的 ID 号有一一的对应关系。在本地资源应是在相应的临时用户下启动任务进程。当任务结束时撤销其临时用户。这里对用户的建立与撤销是由 JobManager 来完成的。这样通过监测进程的所属用户名便可得知其任务的 ID。在网格中，各实体之间的信赖关系是通过安全证书的相互鉴别建立起来的。资源代理与资源之间相互鉴别之后建立信赖关系，在这之后才能向资源发出创建进程的请求。同时这种信赖关系也为在资源上建立临时用户提供了安全方面的保障。操作系统也提供了 API，可以很方便建立、访问用户和设置进程的所属用户以及监控进程。

第四章 基于遗传算法的网格任务分配方法研究

遗传算法是一类模拟自然过程，特别是模拟生物界自然进化和遗传过程的随机搜索算法，具有在复杂空间求解问题近似最优解的能力.遗传算法是基于一个候选解群的迭代过程，它使用遗传算子在问题空间进行搜索，采用适应值来评价候选解的优劣，正是因为遗传算法的强健性和对复杂问题的求解能力，已经有学者将遗传算法应用于分布式系统的任务分配与调度[23]，遗传算法作为一种最有效的启发式局部随机搜索工具，对于 NP 完全且难问题有满意的效果，因此我们采用基于遗传算法的任务分配策略。

首先我们假设:①某一个大型的计算程序已经被分解为若干个子任务，而且每个子任务之间的数据依赖关系已知②资源的实时状态已知，每个子任务在每一个资源上执行的代价已知;资源与资源之间的通信延迟已知，任意两个存在数据依赖关系的任务之间的数据传输量已知。

4.1 问题定义

① 用一个有向无环图 G 来表示各个任务间的调度约束关系[28]。

② n 个任务的集合 T={T1, T2,....., Tn}, Ti 为任务集合的第 i 个任务;

③ m 个资源的集合 R={R1, R2, ... Rm}, Ri 为资源集合的第 j 个资源;

④ 一个 m×n 的矩阵 Tr [m，n], Tr[i][j]为资源 Ri 与资源 Rj 间的数据传输延时，依据每个通信信道的通信速度和通信进程间的距离而变化，是一个实时动态变化的值;

⑤ 一个 m×n 的矩阵 E[m，n], E[i][j]为子任务 Ti 在资源 Rj 上的执行时间，这个依据每个网格节点的处理速度的变化而变化，是一个实时动态变化的值。

4.2 染色体的编码和解码

由 GA 个体的表现型集合所组成的空间称为问题空间。由 GA 基因型个体所

组成的空间称为 GA 编码空间。由问题空间向 GA 编码空间的映射称作编码（encoding），而由编码空间向问题空间的映射称作译码（decoding）。染色体的编码方式有很多，有直接编码方式和间接编码方式[24]。本文采用间接编码方式，对每个任务占用资源编码，染色体的长度等于任务的数量，染色体中的每一位都是正整数，每一位的位置编号代表任务的编号，而该位置上的正整数值代表该任务所占用资源的编号。其中， Ti 表示任务的编号，Rj 表示任务 Ti 执行时占用的资源的编号。在产生初始群体时，每一个染色体中的资源编号 Rj 都是随机产生的，经过交叉、变异算子后，任务 Ti 可能占用任何一个可用的资源，所以最优解一定对应某一个染色体编码。产生了一个染色体后，还必须对其进行译码，得到不同资源上任务的分布情况。将任务按照占用的资源分类，生成多组按照资源编号分类的任务序列，每个序列的编号就是某一个资源的编号，序列中的元素就是在该资源上执行的任务，这样我们就得到了所有任务在多个资源上运行的分布情况。采用这种编码和译码的方式，可以很方便地得到任务集合在多个资源上的分布情况，而且对于运行在不同资源上的任务，不需要考虑它们之间的逻辑关系，只需考虑在同一个资源上运行的任务之间的逻辑关系。下面介绍如何处理在同一个资源上运行的所有任务之间的逻辑关系和它们的执行顺序[12]。

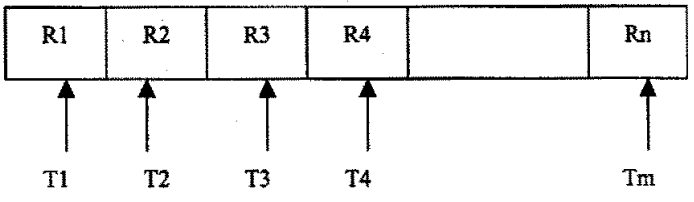

图 4-1 资源-任务编码

4.3 约束关系处理

在同一个资源上运行的任何两个任务都必须满足 D A G 图规定的逻辑关系。假设任务 Ti, Tj 都在某个序列中，且 Ti, Tj 满足 Ti→Tj，即任务 i 必须在任务 j 之前执行。如果在执行序列中任务 j 先于任务 i 执行，则会出现死锁现象。采用基于深度值的排序方法，可以避免出现死锁。

前面提到的所有任务之间的逻辑关系可以用一张 D A G 图表示，可以对 D A G 图进行分层，每层有一个深度值，深度值越小，代表优先级越高，深度值的

计算公式如式(1)：

$$level(Ti)=\begin{cases}0, & Ti\text{ 无父结点}\\ 1+\max(level(parent(Ti))), & \text{其他 Ti}\end{cases}$$

其中，parent(Ti)返回的是任务 Ti 的父结点集合，max(level(parent(Ti)))的返回值是 Ti 父结点集合中具有最大深度值的父结点的深度值。可以采用深度遍历的方法获取 D A G 图结点的深度信息，获取层次深度信息后，对在相同资源上运行的任务序列，就可以按层次深度值由小到大排序，这样优先级越高的任务执行顺序越靠前，对于同一个资源上的所有任务，经过排序后得到的任务执行序列就可以避免出现死锁现象[29]。

4.4 初始群体的生成

初始解群的生成方法是：将 DAG 图中所有结点集合按深度值划分为 h+1 个子集，其中第 i 个子集为 DAG（i），0<i<h+2，h 是 DAG 图中的最大深度值。对于子集 DAG（i），随机地将 DAG（i）的所有任务按深度值作升序排列，就可得到一个合法的调度。只要重复执行这个过程，就能得到一定群体规模的初始群体[29]。

4.5 适应值计算

根据前面所介绍的译码方法，对每个染色体按照资源编号分类，得到每个资源上运行的任务集合，按照上面介绍的约束关系处理原则得到每个资源上任务的运行序列。根据任务之间的逻辑关系和任务的运行序列，可以计算出每个资源完成该资源上所有任务所花费的时间，取最大花费时间的倒数为适应值的大小，因此花费时间越长，适应值越小。计算适应值必须计算每个子任务的完成时间，假设任务 i 在资源 j 上的完成时间为 fin[i][j]，则 fin[i][j]=start[i][j]+ E [i][j]，适应值的大小 bestow[i][j]=1/fin[i][j]。其中 start[i][j]为任务 i 在资源 j 上的开始执行时间。start[i][j]由 3 个因素决定：①资源的空闲时刻；②任务 i 的所有父结点任务的最晚完成时间；③最晚父结点任务所在资源与任务 i 所在资源之间的通信延迟。start[i][j]的计算公式如式(2)，

start[i][j]=max{ free[j]，max(fin(parent(i)))+ T r [m][j]} (2)

其中：free[j]是资源 j 上最近的一次空闲时刻；max(fin(parent(i)))返回值是任务 i 的所有父结点任务的完成时间的最大值；Tr[m][j]是该父结点任务所在资源 m 和任务 i 所在资源 j 之间的通信延迟[30]。

4.6 选择

在交叉操作前，要选择适当的个体进行交叉操作。选择就是从群体中选择优胜个体，淘汰劣质个体的操作。选择的目的是为了从当前群体中选出优良的个体，使它们有机会作为父代繁殖下一代子孙。判断个体优良与否的标准是它们的适应值。个体选择的方法很多,主要有适应值比例选择、Boltzmann 选择、排序选择、联赛选择等形式，为了防止由于选择误差，或者交叉和变异的破坏作用而导致当前群体的最佳个体在下一代的丢失，我们采用精英选择策略，如果下一代群体的最佳个体适应值小于当前群体最佳个体的适应值，则将当前群体最佳个体或者适应值大于下一代最佳个体适应值的多个个体直接复制到下一代，替代最差的下一代群体中的相应数量的个体[27]。

4.7 交叉和变异

我们采用一点交叉方式，交叉点的位置随机确定，经过交叉产生两个新的子个体。当交叉操作产生的后代个体的适应值不再比它们的前辈更好，但又未达到全局最优解时，就会发生早熟收敛。这时引入变异算子往往会产生很好的效果。一方面，变异算子可以使群体进化过程中丢失的等位基因信息得以恢复，以保持群体中的个体差异性，在一定程度上克服了遗传算法的早熟收敛，有利于增加种群的多样性。另一方面，当种群规模较大时，在交叉操作基础上引入适度的变异，也能提高遗传算法的局部搜索效率。这里变异实质上就是将某个子任务迁移到另一个资源上执行。为了防止某个任务在迁移后的执行时间增大而造成种群退化，规定迁移后任务占用的资源不是随机产生，而是在除了该子任务目前占用的资源外的资源集合中选择使该子任务执行时间最短的资源，将其迁移到该资源上执行。由于我们采用的是资源-任务的间接编码方式，交叉和变异算子生成新染色体的过程实际上是对子任务占用的资源的重新分配。在交叉和变异操作后，对所有子任务重新按其占用的资源分类，并对在同一资源上运行的子任务，按照深度值的大小排序，这样就不会出现死锁现象。在交叉和变异操作后生成的新个体依

然符合任务之间的逻辑关系。

证明：变异过程只是将某一深度值的子任务从一个任务列表中移出，再将该任务插入到另一列表中的适当位置，因为插入点 H（Ti）≤n，H（Ti+1）≥n，即迁移过程中没有破坏任务列表中各任务的深度值顺序，因此变异后形成的新解仍然是合法解[30][31]。

4.8 算法流程

（1）初始化矩阵E和矩阵T r，并根据D A G图生成每个子任务之间的逻辑关系，计算每个子任务的深度值；

（2）随机产生大小为M的初始种群；

（3）根据每个资源上的任务的执行序列，计算每条染色体的适应值；

（4）对解的适应性进行评价，如果终止条件不满足，执行（5），否则转步骤（8）；

（5）采用精英策略思想，执行选择机制，形成下一代解群；

（6）选择染色体进行交叉操作和变异操作；

（7）计算解群中每个解的适应值，采用精英策略保存最优解，转步骤（4）；

（8）输出最优解，算法终止。

第五章　基于 Web 的网格入口软件 WebGrid 关键技术研究

我们的 WebGrid 网格平台基于 Globus 和 OGSA 标准，在用户管理和安全方面针对 web 环境下的新的安全需求，借鉴采用了 MyProxy 机制，解决了 GSI 安全机制和 web 安全协议之间的不一致，使两者之间平滑结合，使广大用户能够安全透明的使用网格资源；针对传统网格入口软件监控信息实时性和动性不足等缺点，在网格监控模块增加了对网格任务的监控，用户可以实时的监控网格资源信息和作业的执行情况；在任务调度中采用了基于遗传算法的任务分配策略。

5.1 WebGrid 体系结构

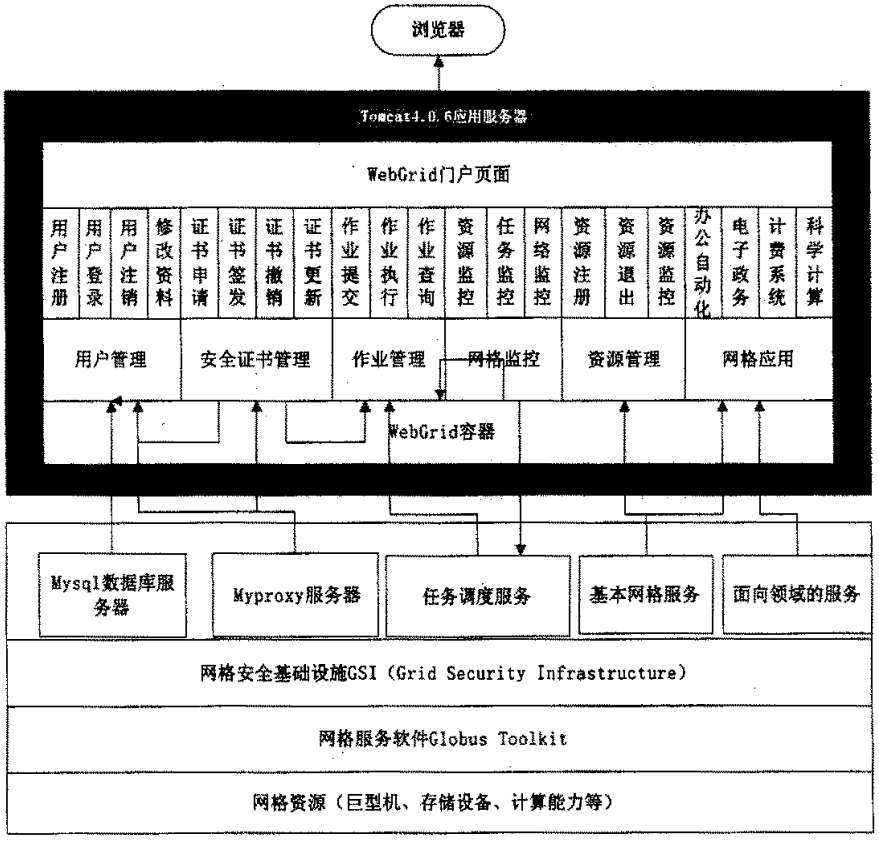

图 5-1 WebGrid 体系结构[50]

5.2 网格安全管理

5.2.1 网格计算环境的安全需求

从本质上讲，Internet 的安全保障一般提供下面两方面的安全服务：(1)访问控制服务，用来保护各种资源不被非授权使用；(2) 通信安全服务，用来提供认证，数据保密性与完整性和各通信端的不可否认性服务。但是这两方面的的安全服务不能完全解决网格计算环境下的安全问题。

从用户的角度出发，最基本的问题就是如何在网格这样的复杂环境中提供给用户简单易用的安全功能。比如单点登陆，一个用户只需要在提交网格计算任务前进行一次认证，以后在任务运行时，安全机制就可以在任务申请其他资源时进行自动的认证，不需要用户再次参与认证过程，还有用户证书和私有密钥必须得到安全而灵活的保护，使用户在任何地方都可以使用自己的证书和密钥，而证书和密钥存放在安全的地方。并且，不应该直接的使用用户自己的证书或密钥，通过代理的方式，使用代理证书来进行认证，这样有利于保护用户的密钥和个人信息的保护。最后，网格的安全机制应该是对用户是透明的。

资源提供者则更关心如何控制和保护自己的资源。比如，虽然资源是提供给网格虚拟组织共享的，但是资源提供者有能力决定资源的本地访问控制，从而改变资源在虚拟组织中的访问控制策略。另外，本地安全系统和虚拟组织安全系统的协调和整合的问题，目前本地经常使用 Kerberos、AFS 等安全系统。取代或是修改这些安全系统是不实际的，我们只能通过映射和代理的机制予以解决[24]。

5.2.2 相关术语与安全策略

我们采用 Globus 的 GSI 的安全基础设施，GSI 的主要安全技术手段包括安全认证、安全身份鉴别、通信加密、私钥保护以及委托与单点登录等。

为了后面的讨论方便，我们先介绍一些概念或术语[2]：

- 主体（Subject）：主体是一个参与安全互操作的参与者。在网格系统中，主体一般代表一个用户或是代表用户进程，还可能是一个资源或是代表资源的进程。主体一般是主动方。
- 客体（Object）：客体是被安全策略保护的资源，是被动方。

- 证书（Credential）：又可称为信任状，是用来证明一个主体身份的一段信息，一般包括口令、私钥和数字证书等。
- 认证（Authorization）：指决定一个主体是否能够访问或使用某个客体的过程。
- 信任（Trust）：如果主体 A 假设相信任何由主体 B 签署的信息这个前提是正确的，则称 A 信任 B。
- 信任域（Trust domain）：信任域是指一个逻辑上的管理组织，在这个组织中实行统一的安全策略。也就是说，一个信任域是由统一的安全策略管理的主体和客体的集合。
- 用户（User）:一个网格任务的请求者，通常是人或代表人的进程。
- 进程（Process）:一个逻辑实体，代表一个用户在特定资源上进行的计算。
- 用户代理（User Proxy）:在一个有限时间内行使一定权限的一个进程。
- 资源（Resource）:在一个计算过程中使用的计算、存储、文件系统，也可能是一组机群、一个分布式的内部网络。
- 资源代理（Resource Proxy）:一个有一定权限的资源管理者，在安全方面主要负责转换本地域和网格之间的安全操作。从而为网格用户屏蔽了本地域的安全实现机制。

现将网格特有的安全需求总结如下：

- 认证需求，包括一站式认证、代理、协同认证、资源认证、基于用户的信任关系。
- 通信保护需求，包括灵活的信息保护策略、支持各种可靠的通信协议、支持独立的数据单元的安全通信。
- 授权需求，包括资源所有者授权、限制代理。灵活的安全策略，包括互操作性、名称映射。用户可选的安全策略、证书安全策略等。

我们定义网格入口软件的安全策略：

- 用户代理和资源代理之间的所有连接需要安全鉴定。
- 所有的安全鉴定是相互的。
- 允许程序或进程以用户的身份运行，并且代理用户的部分权限，对于那些可能运行很长时间，并且运行过程中无需用户干预就可动态申请资源

的程序，也需要这个策略。

- 具有相同资源的资源代理之间相互信任。
- 一个进程管理者是一个负责创建进程的资源代理，而一个进程信任创建它的进程管理者。
- 一个进程管理者和它创建的进程假设在一个单一的信任域中执行。
- 由相同的用户代理/资源代理创建的所有进程之间彼此信任。
- 全局主体和局部主体同时存在。在网格节点上，有一个从全局用户到局部用户的偏序映射。也就是说，一个资源存在两个名字，一个是网格全局名，另一个是本地名。
- 一个经过认证的全局主体在被映射到一个本地主体后，可以等价地认为该全局主体作为本地主体已经在本地经过了认证。
- 所有的访问控制决定都是在本地由本地主体做出的。这也就是说，资源的存取控制权限由其所处环境中的本地安全策略策略确定，并通过本地安全机制来实现。

5.2.3 安全认证

一个网格任务在运行过程中可能动态收缩或增长，即要求获得或释放资源。该任务在获得资源的时候，是以一个具体的用户身份获得的。然而，让用户直接来获得资源的认证过程是不实际的。这样的资源可能非常多，或者该任务的运行时间非常长。这样，用户希望能在任务执行后不再直接干预认证过程。因此，我们采用了用户代理(User Credential)来代替用户而不需要用户的直接干预。User Credential 是一个 session 管理器进程，他被允许在一定时间内代替用户。User Credential 有自己的证书，从而使用户在任务执行过程中不需要在线，并且在每次安全操作中不需要使用用户的证书。由于 User Credential 的生存时间是受用户控制的，可以限制在一个任务的执行期间，所以就算 User Credential 的证书泄密，后果也不如用户的证书泄密严重[25]。

网格入口软件认证的关键是认证证书。入口软件的证书包括四部分的信息：

- 主体名称(subject name)：用来明确认证证书所表示的人或其他对象。
- 属于这个主体的公钥：用于 X.509 认证。

- 签署证书的认证中心标识：其记录了认证中心的名称。
- 签署证书的认证中心的数字签名：可以用来确认认证中心的合法性。

5.2.4 安全身份相互鉴别

如果双方都有证书，而且都信任彼此的认证中心，则双方可明确彼此的身份，这实际上是相互鉴别问题。我们采用 SSL 协议作为它的相互认证协议。

在相互可以认证之前，双方首先要相信彼此的认证中心。在实现上，要求双方有彼此认证和中心自身的证书，认证中心自身的证书中包含认证中心的公钥。这样才能确保双方由认证中心签署的证书具有合法性。

简单的安全身份相互鉴别过程描述如下：为了进行相互鉴别，A 方与 B 方首先建立一个连接，然后 A 方给 B 方自己的证书。A 方的证书告诉 B 方 A 是谁、A 的公钥是什么、签署 A 证书的认证中心是谁。B 方收到证书后，B 方首先要通过检查认证中心的数字签名来确认证书是合法的。一旦 B 检查了 A 的证书合法性，B 需要确定 A 是其认证证书所指的主体。为此，B 生成一个随机信息并把它发给 A，要求 A 对这个信息进行加密。A 用自己的私钥加密信息后，把加密信息发给 B。B 用 A 的认证证书中 A 的公钥对加密信息进行解密。如果解密的结果与初始的信息一致，则 B 就可以信任 A 了，同理，可反向采用上述过程，使得 A 信任 B。这样 A 和 B 就可相互信任，并建立安全连接通道[32]。

5.2.5 通信加密

在缺省情况下，GSI 在双方之间不建立加密通道。一旦相互认证成功，则 GSI 就脱离出来，这样通信双方在进行通信时不会有额外的加解密开销。如果通信双方需要进行加密通信，则网格入口软件可以建立一个共享的密钥用于对信息进行加解密，这里采用公钥技术与对称加密技术结合的加密方式，在保证安全性的同时尽量减少加解密的开销。另外网格入口软件可以保证通信完整性，通信完整性表示窃听者可能会了解到通信双方的通信内容，但不能对通信内容进行修改[33]。

5.2.6 私钥保护

在一般情况下，GSI 要求 Globus 用户的私钥保存在计算机的一个文件中。

为了阻止本地计算机的其他用户窃取私钥，此文件必须经过一个用户知道的口令进行加密保护。这样用户在使用 GSI 中的认证证书时，必须输入口令来解密包含私钥的加密文件。要确保用户能安全的访问这个文件系统以拿到网格证书，只能让它定格在某台机器上，或以某种加密方式对话。一旦离开系统就拿不到网格证书。当然，用硬件存贮也可以解决以上问题，但是，在网格环境下保证硬件的通用性不是一件容易的事情。所以我们采用 MyProxy 机制来解决这个安全问题。

5.2.7 安全委托与单点登录

在通常情况下，如果用户之间与资源管理者之间在建立联系之前，都必须通过相互鉴别的过程。这样，如果一个用户要与多个资源管理者进行联系的话，就必须多次访问保存私钥的文件，即需要多次输入密码，而且在代表用户的程序的运行过程中，也可能进行相互认证，使得相互认证的过程很繁琐。

GSI 对标准的 SSL 协议进行了扩展，使得 GSI 具有了安全委托能力，减少了用户必须输入口令来得到私钥的次数。扩展的功能包括代理证书和证书委托。如果一个网格运算需要多个网格资源，或者说需要一个代表用户的代理来请求资源，GSI 通过创建代理来避免输入口令，这样可以在不同的节点之间成一个安全信任链[34][35]。

5.3 用户管理

在网格系统的研究中，相对于资源管理而言，网格用户管理研究相对薄弱，网格用户的表示、命名以及网格用户管理的功能及体系结构的确定对于实现一个实用的网格用户管理系统非常重要。目前的网格系统，包括 Globus 等都是以服务或资源为中心的，而较少考虑从用户角度出发的特定用户的特定需求，根据用户的不同组织、角色和完成的功能提供不同的网格视图。

用户管理子系统包括的功能是：①与安全服务或其他用户身份认证中心配合，负责网格用户证书的申请和批准，网格用户身份的创建和删除②存放用户的公钥和私钥的信息，并可对用户信息进行修改③负责用户的登录和退出，相应地创建或删除用户的上下文④与资源或服务方配合，解决网格用户与资源方本地用户的对应。

5.3.1 网格用户管理基础模型

在网格中想要进行任何网格计算，一般都需要用户提供它的证书（包含私钥），再通过认证后才能使用网格资源。普通进入网格的方式存在两个麻烦：①网格证书是以文件的形式存放在文件系统中的，而私钥必须具有保密性。要确保用户能安全的访问这个文件系统以拿到网格证书，只能让它定格在某台机器上，或以某种加密方式对话。一旦离开系统就拿不到网格证书。当然，用硬件存贮也可以解决以上问题，但是，在网格环境下保证硬件的通用性不是一件容易的事情。②不是所有的网格应用程序都可以使用 GSI 的，没有 GSI，就缺乏作证书代理的能力。例如客户端的网络浏览器，它有网格证书，但缺乏 GSI 的能力去作证书的委托。这意味着虽然它们凭网格证书到达网格入口，但却不能委托入口帮它办事。正因为这样，网格入口被迫发给用户一个长期证书，赋予用户长久的特权。但是这样容易给他人复制证书，暴露个人隐私。同时网格入口为了管理这些长期证书，也加重了自身负担[32]。

许多网格入口工具包需要用户授权服务器以用户身份去执行任务，用来发起和监控用户对资源的操作，而在网格环境下，资源被 GSI 所保护，所以 Globus 的构件称为网格事实上的标准。GSI 支持这种授权，而标准的 Web 安全协议并不支持。这样就在网格安全和 web 安全协议之间产生的不一致性[32][36]。Web 浏览器可以使用证书进行网格的认证，但是缺乏委托（Delegation）功能将使用户受到很大限制。用户可能需要将自己的长期证书复制到他想要从其上登录的地方（如网格入口）。这样做不但麻烦而且不安全。

为了解决如上问题，我们提出了一个在线的证书仓库系统-MyProxy 的概念。MyProxy 的目的是：（1）允许用户在网格的任何地方访问他的证书，甚至他所在系统没有网格软件支持或是无法安全的访问他的长期证书。（2）允许用户委托证书给其他资源，即使一些应用没有提供委托机制。（3）尽可能的移除那些不再使用的证书，降低被攻击的机会。比如为了认证而将证书复制在 Web 服务器上。（4）可扩充性。（5）允许用户尽可能的控制他的证书。网格入口只有在得到用户的批准后才能得到用户的证书[39]。

MyProxy 是一个数据库服务器，用户可以将他的证书送给 MyProxy 存储，并在需要的时候可由用户或某个服务取回。我们将 MyProxy 的使用分为两个阶段，

委托证书和取回证书。

委托证书阶段：也是用户在 MyProxy 服务器注册的过程。用户使用自己的永久证书与 MyProxy 服务器建立连接并证明自己的身份，然后委托一组代理证书给服务器，委托的方法如前所述，另外还包括一些策略，如取回证书时使用的认证信息和对取回证书的限制等。认证信息包括用户的 ID 和口令，这些信息由用户自己注册。在限制策略中，可以限制代理证书的有效期等，比如通常服务器拥有的代理证书有效期为一周，过期后用户需要重新委托证书给服务器。不过，用户可以对服务器中的代理证书进行修改、删除等操作，这为用户带来了方便。

证书取回阶段：在委托了代理证书之后，用户或代表用户的一个服务就可以连接 MyProxy 服务器并请求一个用户证书的委托。用户必须提供他在委托时所设的 ID 和口令，在验证了这些信息并检查了限制策略之后，服务器就委托一个新的代理证书给用户或服务，这个代理证书在以后可以签发更多的代理证书[37]。

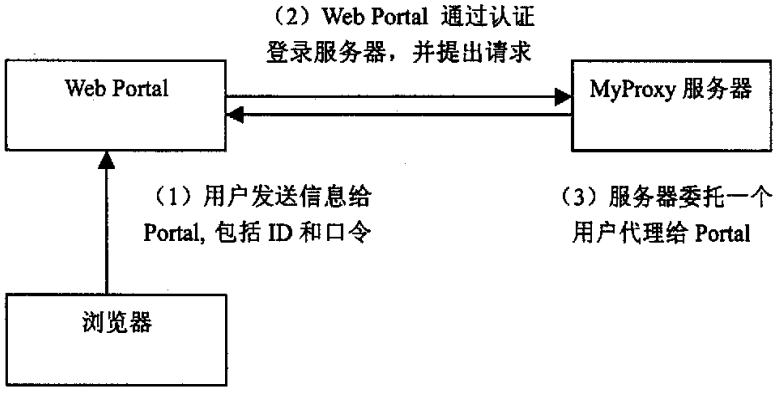

图 5-2 用户与 MyProxy 交互的过程

如上图所示，用户在 MyProxy 上注册后，就可以在以后通过某个网格入口（如 Web 浏览器）登录到 MyProxy，并且用户可指定网格入口使用哪个 MyProxy。然后，网格入口向 MyProxy 请求一个代理证书给用户。MyProxy 在验证了用户的 ID 和口令后，委托一个代理证书给网格入口，网格入口就可以代表用户进入网格并可以使用标准的网格应用。当使用完后，需要删除入口的代理证书。并且如果用户忘记登出，证书的有效期也会限制用户代理的继续使用[38]。

5.4 作业管理

网格作业管理直接关系到网格资源性能的发挥和使用率的提高，是网格的一个重要的组成部分。作业管理子系统的主要作用是强化系统的作业管理功能，提供作业提交、调度、执行及控制等功能，更加有效的利用系统资源、平衡网络负载、提高系统的整体性能。

5.4.1 作业管理基本概念

- 作业（Job）：作业就是指用户提交的已编译好的脚本。
- 任务（Task）：在特定操作系统上的一个可执行程序。作业由多个任务组成，一个运行的任务可以是操作系统中的一个进程或多个进程。
- 批处理（Batch processing）：由操作系统上的一个子系统而不是由用户使用交互式的会话过程运行作业的方式，称为批处理。
- 网格节点（Node）：本身是一个完整的系统，可以是具有一个或几个 IP 地址的一个集群、一个工作站或是单个 PC 机。
- 负载（Load）：对网格中的每个节点，选定一组作业使用的资源类别（如节点处理能力等）作为统一指标，将一个节点上全部作业占用的资源按这一组指标进行测算，得到的值称为节点的负载。
- 负载平衡（Load Balance）：是指在一个网格范围内通过合理的作业分配，保持节点负载基本平衡。
- Gatekeeper：一个拥有 root 权限的进程，主要负责处理任务分配请求。当 gatekeeper 接收到一个由 client 发出的任务分配请求后，它负责的操作包括：与 client 进行相互安全鉴别、把 client 映射为本地的一个用户、在本地启动一个拥有本地用户权限的任务管理者、把根据 RSL 描述解析出的任务分配参数传递给新创建的任务管理者。
- 任务管理者（Job Manager）:由 gatekeeper 根据任务请求创建。主要负责在本地系统中启动任务，并处理与其它用户的通信。
- 动态协同分配代理（DUROC）:用来提供与系统相关的调度器，负责各个资源管理者之间的协同交互[2]。

5.4.2 作业提交及解析

用户作业的执行通常会涉及到分布式数据、软件、网络资源、存储资源和软件资源，另外用户可能对服务质量有一定的要求，如任务必须在什么时间之前完成。这些资源需求由 RSL 提交给了网格调度系统。用户或应用定义需求后提交任务给网格调度服务，网格调度服务根据这一需求找到合适的资源。作业描述语言包含了必要的信息足以判断资源是否能够满足这一需求。作业描述语言包含三方面的信息：一是作业本身的属性，如可以分为几个部分，相互之间的依赖关系，被并行化为 DAG 的形式；第二是作业的约束条件，如作业的完成时限，服务质量等[36]，可能还有网格经济模型中的“代价”的概念；第三是对实际静态资源的需求，如要求程序运行的操作系统及执行环境，最小的内存要求，网络带宽的要求。

5.4.3 映射策略

网格是建立在各个资源节点上的一个抽象层，对网格资源的访问都将最终是对本地节点上资源的访问。那么为了执行本地安全访问控制，网格中的主体必须映射为一个本地资源“知道”的主体。我们称网格中的主体为全局性的，而节点的主体为本地的。

全局主体映射为本地主体就是指全局名转化为本地名（登录名或用户 ID）。完成这种映射可以有两种实现方法：（1）资源代理维护映射表，表的内容就是映射关系。这种方法需要管理员来维护映射表，由于网格的用户可能很多，管理员的维护工作会很大，但是该方法的优点是简单和统一。（2）用户自己来添加映射关系。这需要用户持有全局和本地的证书，并且用户在通过全局认证和本地资源的认证后，就可建立两者之间的映射关系。资源代理在映射关系中起到协调的作用。这种方法克服了方法一的缺点，但是要求用户拥有本地证书[34]。

5.4.4 作业分解

如果要执行需要分布式资源的任务，需要动态协同分配代理 DUROC 负责各个资源管理者之间的协同交互。DUROC 接收作业描述语言后，首先把抽象的资源描述转化为具体的资源描述，处理底层的资源描述，把复杂任务进行分解成具

有先序关系的 DAG 图的形式[28]，然后使用基于遗传算法的任务分配策略来进行任务分配[26][27][29][31]。

5.4.5 作业调度

作业管理系统的核心是作业调度，调度一般分为以下四个步骤：

- 资源发现

  调度器首先要把抽象的资源描述转化为具体的资源描述，然后为用户发现一系列可用的节点。大部分的资源发现算法都跟一些网格信息服务有关，比如 Globus 中的 MDS。为了满足应用的最大需求，这个最初经过授权的资源列表还可以过滤[42]。

- 资源选择

  一旦可用的目标机器列表已经知道了，资源代理的第二阶段的任务就是选择用户所期望的最佳资源值。为了满足用户的时间限制，资源代理还必须动态的收集可用的资源信息、系统负载、网络性能等信息，这种信息是由不同的服务提供的，比如 NWS。然而在一个比较实用的环境中，用户不应该指明时间限制，而应该指明代价限制，因此资源代理还要去收集资源代价方面的其他信息[43]。

- 作业调度

  资源代理的下一步就是作业调度了，由于前面所述任务的分解已由资源协同分配代理完成，而上面已介绍了资源发现和选择，所以如何获取资源信息和任务的分解方法在此不做讨论。我们在系统实现中采用基于遗传算法的调度策略，首先将已分解的具有依赖关系的子任务分配到已选择的网格资源上，对于分配到同一资源上的任务序列采用 Globus 的先来先服务任务调度策略执行作业。动态协同分配代理 DUROC 负责各个资源管理者之间的协同交互，通过 DUROC 提供的通信库以进行任务之间的通信[44]。

- 作业监控和迁移

  由于网格环境经常会有不可预知的变化，因此网格本身就是一个动态的系统。网格环境的变化包括：系统或网络错误，系统性能下降，增加一些新节点，资源成本的改变等。在这种情况下，作业迁移就能唯一有效地保证所

提交的作业的完成以及用户需求的满足。跟作业迁移有关的主要任务有作业监控、资源调度和检查点。作业监控负责检测可能会引起迁移的警报点，并把检测到的信息发送给调度器，由资源调度器来评价是否值得迁移作业，同时为作业进行一次新的分配。检查点则是周期性的捕获正在运行作业的状态，这样作业就能在迁移情况下在稍后的时间从那个状态接着运行[41][45][46]。

5.4.6 作业提交与执行过程描述

由于 Globus 的 GSI 对安全处理做了大量的工作，使得在网格环境中的任务提交与执行的安全性和方便性有了很大的提高。下面就对作业提交与执行进行一个简单的描述：

1. 在进行作业提交与执行之前，用户从 MyProxy 服务器上获得安全认证证书，并创建临时的用户代理。
2. 如果是需要多个分布式资源的作业请求，则把作业请求提交给动态协同分配代理 DUROC，DUROC 解析作业请求，将作业分解后通过基于遗传算法的任务分配算法将子任务分配到各资源，发送到相关的资源管理者 GRAM。
3. GRAM Client 与远端节点的 gatekeeper 进行相互安全鉴别，即对二者的安全证书和身份进行鉴别。
4. 通过相互安全鉴别后，GRAM Client 通过 GASS 服务把任务提交给远端 gatekeeper，由 gatekeeper 把任务交给任务管理者进行具体的处理。
5. 如果任务在执行过程中需要访问远程资源，也必须在任务进程与资源代理之间进行相互安全鉴别，通过安全鉴别之后，还要进行授权、本地 ID 映射后，任务进程才可以使用资源。
6. 如果任务在执行过程中需要访问远端数据或文件，也必须在任务进程与远端文件服务资源代理之间进行相互安全鉴别。通过安全鉴别之后，还要进行授权、本地 ID 映射后，任务进程才可以进行远端数据或文件访问。
7. 当任务执行完后，通过 GASS 服务取回任务执行结果，并撤销用户代理[48]。

5.5 网格监控

5.5.1 总体框架

网格计算环境最重要的特征之一就是可以动态组建虚拟组织，并且虚拟组织的成员可以随时随地加入或离开。然而不论是集中式网络监控，还是分布式网络监控，都需要在地理位置相对固定的节点上部署监控系统。显然，传统的网络监控系统并不合适动态变化的网格计算环境。在充分考虑网格计算环境自身的情况下，本文提出网格监控模型具备本地存储的特征：

网格资源状态信息将尽量存储在本地的目录服务器中。由于网格计算环境中存在各种动态变化的资源，如果采用集中的信息存储，需要随时轮询更新，这样大大增加了网络负载和数据传输。而且如果将网格资源信息存储在固定的服务器上，一旦出现故障，或结构发生变化，网格资源的调度和分配将处于完全的瘫痪状态。在本模型中，网格计算环境被划分为若干虚拟组织，每个虚拟组织由一个全局目录服务器和多个节点构成。全局目录服务器并不储存具体的网格资源信息，它只提网格资源的名称和位置信息。“节点”的概念可以是工作站、服务器，或是计算机机群，甚至是局域网。节点之间可以按照逻辑需要组成有层次的结构，全局目录服务器通过目录信息树来记录这些结构。

5.5.2 用户监控流程

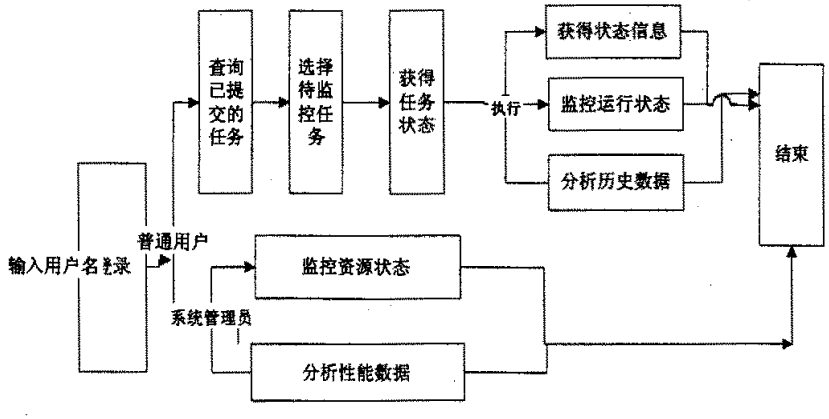

图 5-3 用户监控流程图

5.5.3 与 MDS 结合的监控模型

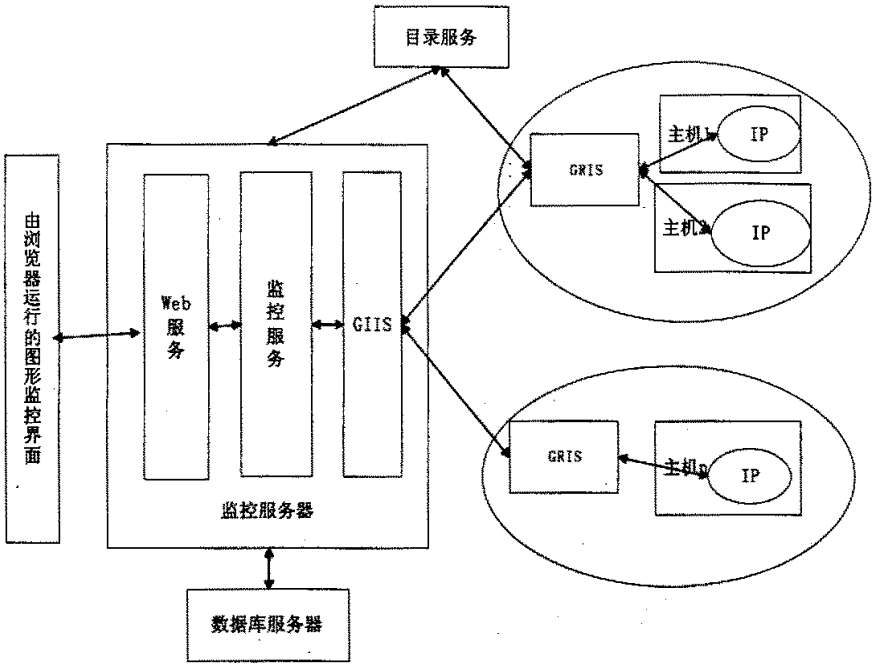

图 5-4 使用 MDS 实现监控服务

我们采用与 MDS 结合的方案，充分利用 MDS 原有功能、扩充新功能、提供良好的信息访问形式。具体为：

（1）MDS 已有许多信息提供者，可利用起来，另外还应扩充新的信息提供者类别、增加新的信息提供者。

（2）利用 GRIS 和 GIIS 提供的信息整合和查询功能。

（3）MDS 提供的用户对信息的访问方式有两种：一是控制台命令方式，二是 LDAP 浏览器方式。对于一个相对独立的面向网格应用客户的监控系统，这两种方式都不合适，所以应当改善。

（4）MDS 未能实现任务监控功能，可以将任务状态信息与任务执行情况的提供者定义为新的信息提供者[52]。

由于 MDS 提供的信息不能准确的表达网格环境下任务的执行状态和任务运行期占用资源情况，所以我们必须开发新的信息提供者。开发新的信息提供者有如下步骤：

（1）开发新的 MDS 的信息种类和该信息带在 DIT（目录信息树）中的描述，

也就是说要定义一个提供者“概要”、OID 和名字空间。

（2）创建一个输入/输出接口程序，该程序必须能被 GRIS 后端的 fork()和 exec()调用，并根据已定义的结构返回 LDIF（LDAP 数据交换格式）数据对象。GRIS 后端是指 slapd server 的扩展模块，它通过 OpenLDAP Genegic API 来扩展其功能。

（3）修改 grid-info-resource-ldif.conf 文件，将其增加新条目。在 grid-info-resource-ldif.conf 文件中定义了所有当前可用的信息提供者，为使得用户编制的信息提供者程序可用，则必须加入相关的条目[52]。

5.5.4 自主开发的监控系统

该方案不使用已有的网格监控中间件，而是综合使用现有的各种分布式技术自主实现整个网格监控服务系统。体系结构为：

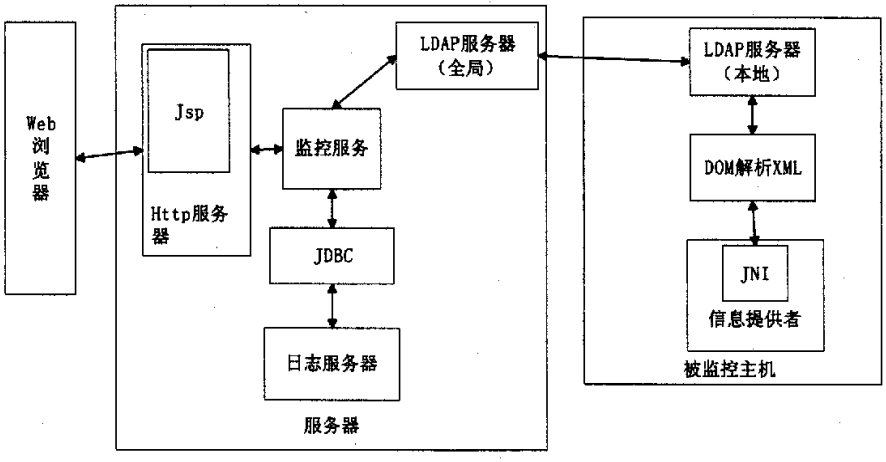

图 5-5 自主开发的网格监控系统结构图

具体实现方案为：

（1）客户端采用网页形式，用 JSP 设计。

（2）服务器端主要采用 Java Servlet 技术实现监控服务。建立了日志数据库，通过 JDBC 接口实现对日志数据库的绑定。

（3）被监控主机上运行的信息提供者开发使用 JAVA 和 C 语言。使用 JNI 实现 JAVA 程序和 C 程序的结合。

（4）对监控数据的描述采用 XML 形式。

（5）被监控主机采用 DOM 接口解析 XML 形式的数据，然后存储在本地 LDAP 目录

服务器中，并加盖有效时间戳。本地节点的资源和任务信息实时更新。为了提高网格监控和查询的响应时间和执行效率，我们还提供一个缓存服务。被监控主机接收到一个新的查询请求时；如果发现缓存中的资源和任务信息仍在有效时间范围内，则直接从本地 LDAP 服务器中读取数据并返回，如果发现缓存中的资源或任务信息已经过期，则需要通过 DOM 接口解析得到最新的资源和任务信息[53]。

（6）监控主机的目录服务器上并不存储具体的网格资源和任务信息，它只提供网格资源和任务的名称、静态特征和位置信息。当网格节点想加入某个虚拟组织，则移动到全局目录服务器上，经过批准后将相应的名称和位置等信息加入目录树中。若想获得网格资源和任务的信息，首先要通过全局目录服务找到该网格资源和任务的位置和静态信息，比如主机地址和端口号等，然后向该网格节点发出请求获得相应数据。

5.5.5 两种监控体系结构的比较

采用与 MDS 结合的方案来开发网格监控服务时，设计者只需考虑客户端的设计、服务端与 MDS 的紧密结合、自定义信息提供者的设计；而不需要考虑自己设计目录服务，也不需要考虑被监控主机与监控服务器之间的建立以及监控数据的解析和传递过程的种种细节问题。该方案的优点是思路简单、清晰；实现方法具体、明确。其缺点是缺乏灵活性，以至于不方便某些功能模块的实现。

自主开发方案需要将整个监控服务系统的所有细节进行全盘、细致的考虑与设计，不仅需要自己设计目录服务，而且还需要考虑分布在不同主机的监控服务系统的各部件之间的通信、数据的表示形式等。

第六章　基于 Web 的网格入口软件 WebGrid 设计与实现

6.1 系统的总体功能

在核心高性能计算平台的基础上，完成网格入口软件的设计和开发，基于 Globus Toolkit 有机集成 Linux 集群系统，实现统一的校园网格所需要的资源信息共享，资源管理和调度，以及安全机制，并通过网格入口软件系统来发布和提供网格应用服务。网格入口软件系统目前支持用户管理、作业管理、资源管理、资源监控与任务监控等。

6.1.1 系统用例图

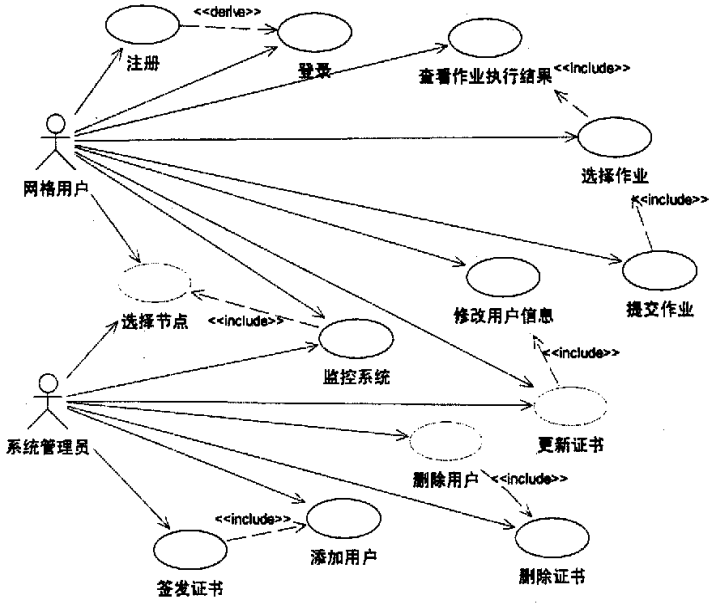

图 6-1 系统总体用例图[50]

6.1.2 系统实现拓扑结构图

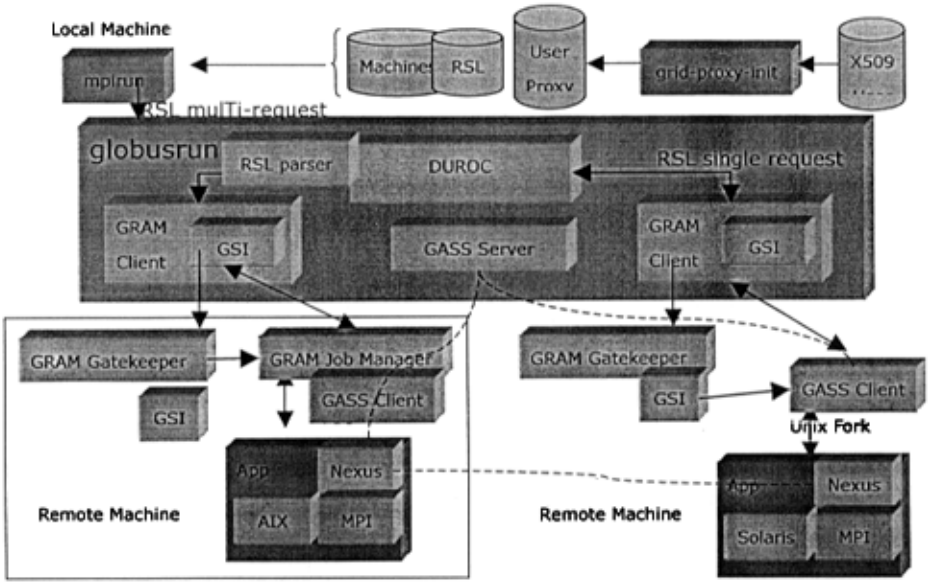

图 6-2 WebGrid 的系统实现拓扑结构图[50]

6.1.3 网格入口软件平台实现环境

我们利用 Globus 3.0.2 建立了网格应用开发平台，该平台目前主要由软件工程研究所的三台 PC 服务器组成，都安装了 Red Hat Linux 9.0，并且在网格资源调度器和 Web 服务器之间安装了防火墙，其基本结构如图 6-3 所示:

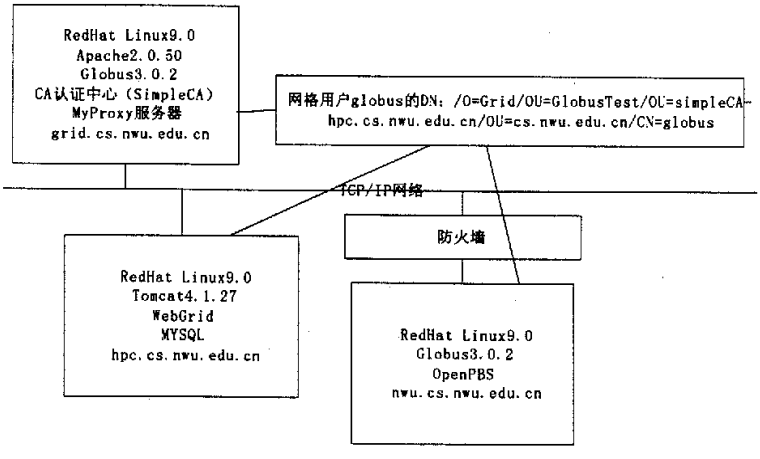

图 6-3 WebGrid 实验平台

- **网格管理节点 grid.cs.nwu.edu.cn:** 安装了网格软件 Globus，采用 SimpleCA 建立了 CA 认证中心，其它网格结点都需要安装它生成的分布式 CA 包，并且所有网格证书申请都需要发送到此管理结点，由管理结点的系统管理员负责签发网格证书；安装了 Myproxy 证书库，提供用户委派证书到库里和用户通过入口软件检索证书；Apache 服务器负责网格监控服务。
- **网格资源调度节点 nwu.cs.nwu.edu.cn:** 安装了 Globus 和节点 grid.cs.nwu.edu.cn 上 CA 分布式包；采用 OpenPBS 作为作业队列管理。
- **Web 应用服务器 hpc.cs.nwu.edu.cn:** 安装了 Web 服务器软件 Tomcat、入口软件 WebGrid、数据库 Mysql。

6.2 用户管理子系统的设计与实现

6.2.1 功能描述

网格用户管理主要提供的功能如下：用户注册，用户登录，证书签发，证书管理，用户基本信息管理。

6.2.2 用户界面设计

图 6-4 网格用户管理界面

6.2.3 详细设计

用户管理子系统的核心类图 6-5 所示。

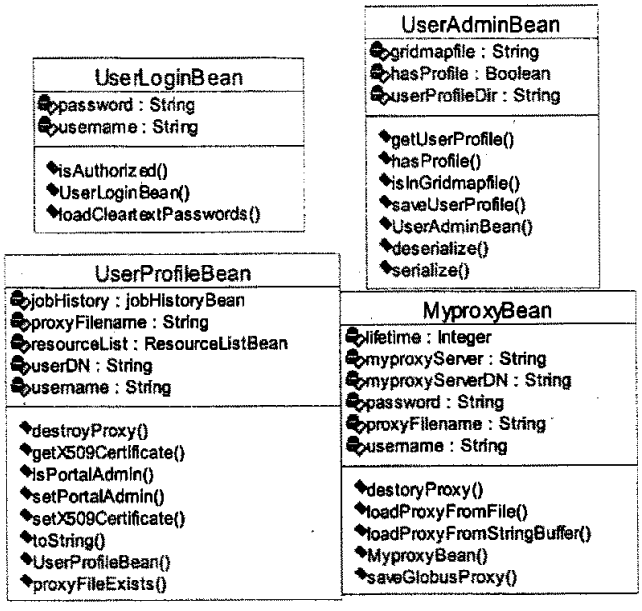

图 6-5 用户管理基本类图

6.3 作业管理子系统的设计与实现

6.3.1 功能描述

作业提交模块主要实现作业的提交，查询等功能。

6.3.2 用户界面设计

图 6-6 作业提交页面

图 6-7 交互式的作业结果页面

图 6-8 批处理方式作业结果查询页面

6.3.3 详细设计

作业管理模块最主要的类是 JobBean, GramSubmissionBean, GSISSHSubmissionBean, JobInfoBean, JobHistoryBean, 其中 JobBean 包含了提交作业所必须的属性，包括内存需求、执行程序的名称、参数、进程个数等等。JobBean 作为参数传给 GramSubmissionBean，GramSubmissionBean 将作业提交给 Gatekeeper, 当作业提交后，JobInfoBean 可以得到作业提交的信息，包括作业提交的时间表等等。用户可以作业的 ID 号来检验作业的状态、作业的输出结果等等。JobHistoryBean 用来存储多个作业的信息，提供已经提交作业的历史信息。

该模块用到的主要类图如图 6-9 所示：

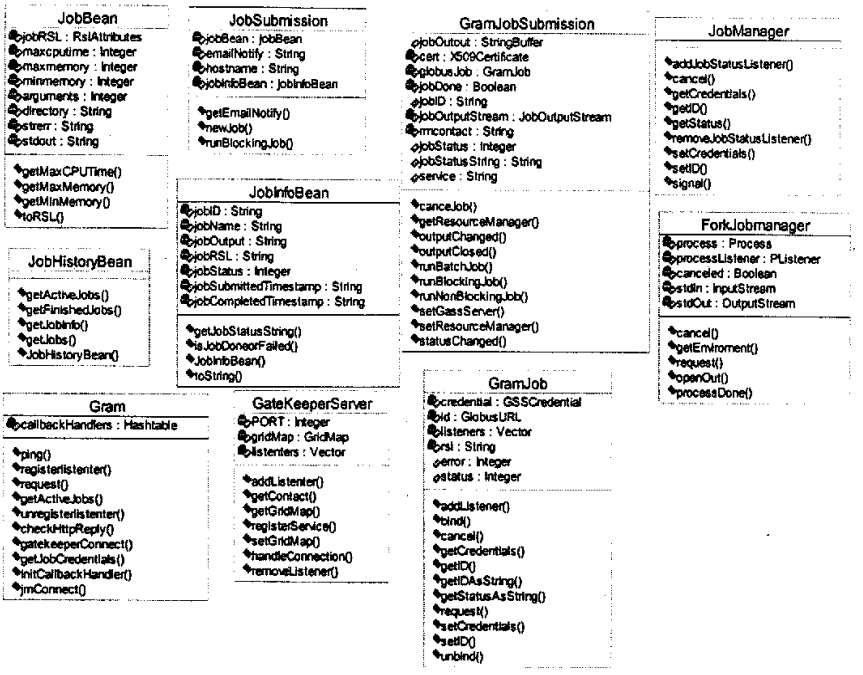

图 6-9 作业管理模块基本类图

6.4 监控子系统的设计与实现

6.4.1 功能描述

网格监控子系统的主要功能可归纳为：

（1）资源监控：对参与设计的各主机基本信息及状态进行监视，主要有：静态信息，包括设计单元总数、每台计算机的 IP 地址、计算机名、处理机数目、当前运行的操作系统名称、用户和组的状况、设计单元的特长信息。

动态但变化不频繁的信息，包括当前参与设计的单元个数，设计单元的关联系数、文件系统状况。

动态且变化频繁的信息，包括系统时间、CPU 占用率、内存使用情况、进程名、进程编号、进程对 CPU 和内存的消耗情况等。

（2）任务监控：监控用户所提交的任务的分配、执行情况，包括对任务状态和任务对资源的占用情况的汇报。

（3）异常报警：能判断是否为异常并针对异常做出处理。由于网格的广域

性、动态性以及设计环境的复杂性，作业执行过程往往因为一些突发情况而中断或终止，为了有效保证设计的顺利进行，设计了异常报警中间件，主要包括作业执行过程状态采集与分析，异常诊断与报警。作业执行过程状态采集与分析主要是获取当前任务的状态信息，综合各种参数，对状态做出准确判断，既不遗漏任何导致设计失败的异常状况，也不将一些正常状况误判。在动态性很大的网格环境中，所谓异常不能与相对稳定的机群环境相提并论。如：网格中参与计算的节点退出或加入是很平常的事，不能视为异常，在网格中间件的支持下，系统基本能适应诸如此类的变化。

（4）日志记载：日志的形式有文档型和数据库型。文档型日志多用在数据类型少、数据量小的情形；而当数据类型多、数据量大时多采用数据库型日志。在这里选用数据库型更合适。日志包括监控数据的历史记录和报警记录。因为监控数据的信息量非常大，不可能对所有数据都保留其记录，因此需要用户进行一些选择性的设置，如节点监控范围、监控数据种类、时间等。另外，当出现异常时，系统自动将与该异常类型相关的数据记录下来[51]。

6.4.2 用户界面设计

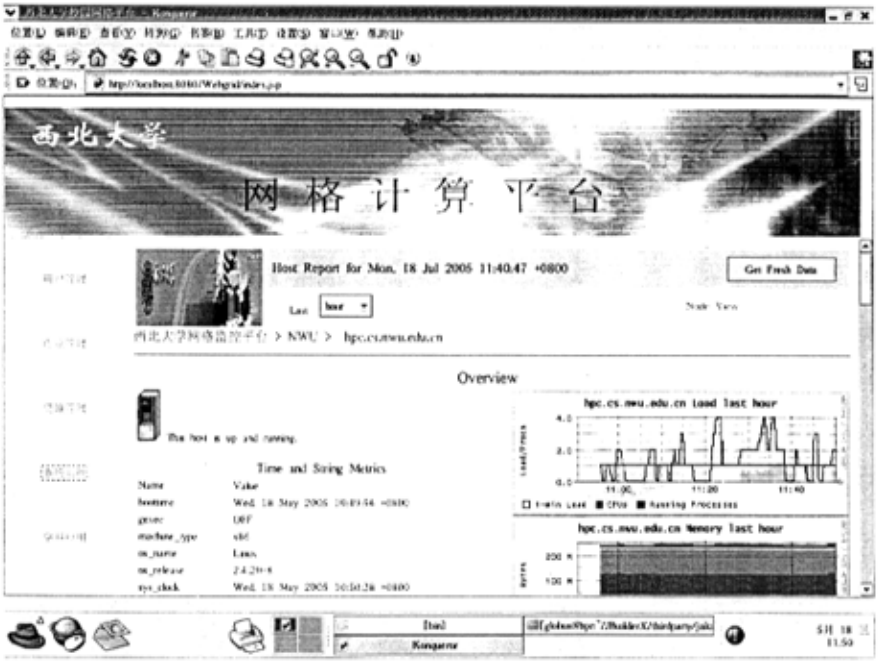

图 6-10 网格监控子系统界面

6.4.3 详细设计

6.4.3.1 监控数据的表示格式

对于在监控程序与信息提供者之间传输的数据，全部采用 XML 格式来描述[53]。XML 以通用的形式来描述结构化的数据，许多编程语言都有用于解析 XML 格式数据的接口，所以 XML 特别适合与多种编程语言联合使用。

6.4.3.2 服务端的设计

1. 各功能模块的划分与设计

监控服务模块比较复杂，要实现的功能繁多，可划分为多个子模块：

资源监控服务模块、任务监控模块、性能分析模块、日志查看模块、数据解析模块。

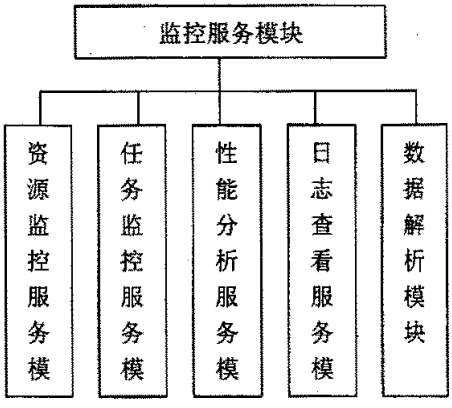

图 6-11 监控服务模块

（1）资源监控服务模块的设计：

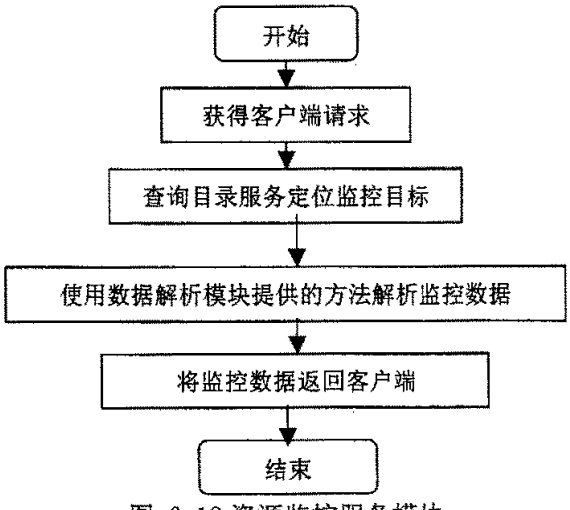

图 6-12 资源监控服务模块

（2）任务监控模块的设计

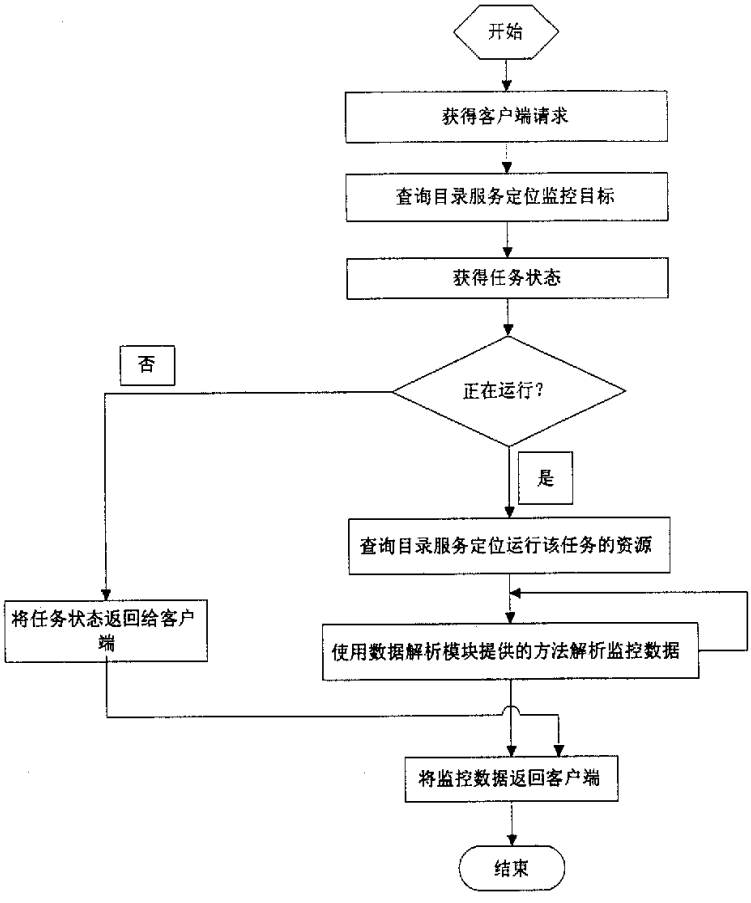

图 6-13 任务监控服务模块程序主要流程图

（3）数据解析模块

由于本地目录存储的数据采用 XML 进行定义，所以使用 XML 解析器，通过 DOM 接口对 JNI 编程接口得到的数据进行解析。解析流程如下：

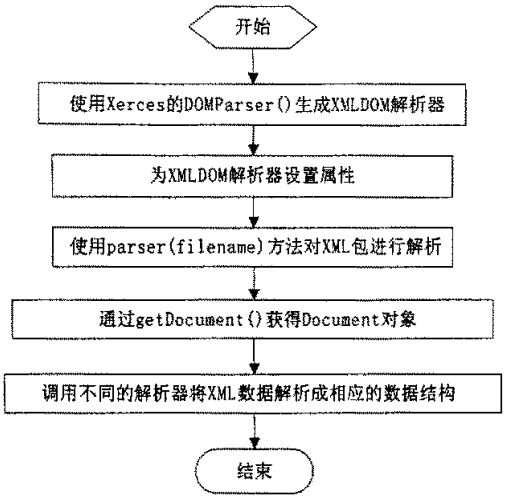

图 6-14 解析流程

2. 日志数据库的设计

该日志数据库存储的是网格系统的历史信息，主要提供系统管理员查看以了解某时间段系统的运行情况。普通用户还可以通过它查看自己提交的已结束的任务的历史情况。另外日志还记载报警信息。下面分别介绍这三方面所包含的数据库表：

（1）网格系统运行情况日志包含的数据库表：

资源情况数据库表：是对加入本网格系统的各资源情况的描述，主要字段有主机名、主机 IP 地址、操作系统、CPU 型号等。

系统负载情况数据表：记载各资源节点的负载情况，主要字段有主机名、时间、平均负载。

任务统计表：记载系统执行的任务量，主要字段有任务号、所属用户、状态、提交时间、完成时间等。

（2）任务历史情况数据表：主要字段有任务号、所属用户、状态、提交时间、完成时间等。

（3）报警日志数据表：主要字段有故障时间、故障类别、故障发生部件类别、地点等。

使用 JDBC 实现服务程序与日志数据库的绑定，这样服务程序就可以方便的访问数据库。

6.4.3.3 信息提供者的设计

1. 主机静态和动态信息的获取

以下以 Linux 和 Windows 平台为例来分析如何获得主机的静态和动态信息[56]。

守护进程的创建

监控系统应能监测到被监测系统整的运行期间的状态信息，所以信息提供者应是常驻内存的。守护进程实际上就是驻留内存中的处于运行状态的程序，创建守护进程实际上就是创建进程并使它不断运行，每个一段时间便执行一段应用程序。

Linux 中创建进程的方法是通过“一分为二”创建原有进程的子进程。原有的进程成为父进程。一个父进程可以创建多个子进程，子进程同样可以创建自己的子进程。这样形成一个树形结构，其根为 init 进程，它是系统启动时实际运行的第一个实际进程，其 PID 为 1。

主机信息的获取

系统提供如下主机信息结构：

进程信息的获取

- Linux 进程信息的获取

使用 Shell Script

Linux 系统下提供查看进程信息的命令，可利用这些命令编写 Shell Script 完成 Linux 平台下信息的搜集。然后在程序中通过 Java 的流重定向技术得到这些信息。通过解析程序将这些信息包装成 XML 数据包的格式。

可通过调用 C 库函数 system 执行 Shell 命令：

Int system(const char* command) 将 command 作为一个 Shell 命令执行。若不能生成 Shell 进程，返回－1，否则，返回 Shell 进程状态值。

可使用 JNI(Java Native Interface)技术实现 Java 和 C 的混和编程。

- Windows 进程信息的获取

在 Windows NT 和 Windows2000 中，可使用在 PASAPLDLL 中提供的 ProcessStatusHelper 函数帮助我们达到目的。

获取进程 ID 的函数 EnumProcesses()

BOOL EnumProcesses(DWORD *IpidProcess, DWORD cb, DWORD *cbNeeded)

通过进程句柄获取进程文件名的函数 GetModuleFileNameExA()

DWORD GetModuleFileNameExA(HANDLE hProcess, HMODULE hModule, LPTSTR lpstrFileName, DWORD nSize)

要使用以上函数，必须进行静态或动态加载库。由于静态加载（使用隐含链接，操作系统在加载应用程序的同时将加载应用程序所使用的 DLL），除了要有相应的 DLL 库文件外，还必须提供头文件和输入库文件，而 Windows2000 系统中，只提供 DLL 文件，没有相应的头文件和输入库文件，因此应采用动态加载。

第七章　总结与展望

7.1 结束语

当网格技术从实验室走向应用时，我们面临着巨大的挑战。这些挑战不进来自网格系统本身，如系统稳定性、健壮性等问题，而且，有很大一部分是来自网格用户。网格用户向网格系统提交任务以后，除了等待任务被执行完毕外，还很需要了解任务的执行情况：处于等待状态还是正在执行？执行进展如何？在必要时是否可以终止任务的执行？这就使得任务监控成为成熟网格中的必备服务。另外用户还需要了解网格系统中资源的使用情况和任务的执行情况，从而进行性能分析等等。任务调度是网格是否能高效使用资源、快速完成任务的关键构件，任务调度算法的优劣直接影响着网格计算系统的性能。

本论文围绕开发的基于 Web 的网格入口软件 WebGrid，提出了网格环境下任务监控的方法，介绍了基于遗传算法的任务分配策略，在 WebGrid 用户安全管理模块采用 MyProxy 机制，解决了 GSI 安全机制和 web 安全协议之间的不一致性，使两者之间平滑结合；作业调度模块采用了基于遗传算法的任务分配方法；系统监控模块中的任务监控部分采用了文中提出的任务监控方法。

7.1.1 主要内容包括

- 分析了当前国内外网格入口工具的研究现状。介绍了网格、网格入口软件、网格监控的基本概念、特点。对网格监控和任务调度进行了深入研究。
- 研究实现了基于 Web 的网格入口软件 WebGrid。
- 提出了网格环境下实现任务监控的策略。目前的网格中间件 Globus 对任务监控缺乏足够的支持，本文在现有技术基础上提出了实现任务运行期监控的方法。
- 在系统中实现了基于遗传算法的任务分配策略。

7.1.2 本文的主要创新点

- 在网格安全管理中采用 MyProxy 机制，解决了 GSI 安全机制和 web 安全

协议之间的不一致。

- 提出现有网格环境下实现任务监控的方法，开发了网格监控系统。
- 在系统中实现了基于遗传算法的任务分配策略。

7.2 进一步的工作

网格技术是一项具有挑战性的工作，关于它的研究很多，但还没有特别成熟的理论和模型产生。因此，本文所做的研究只是这个领域里的很少一部分，还有很多的问题需要进一步的深入探索。比较明确的下一步的工作有：

- 网格注册问题。现有的网格资源注册与注销都是通过静态的配置文件实现的，这样不适合网格动态环境的要求，在网格环境下，网格节点和各种资源都随时有可能加入或退出，所以必须找到一种机制支持网格资源的动态注册与退出。
- 作业调度问题的进一步研究。本论文对涉及到的作业调度问题的研究还很肤浅，基本上停留在试验阶段，对于任务分解策略和新的调度算法还需要做进一步的研究，有待提出新的任务分解模型和任务迁移策略[48]。
- 增加网格文件系统。拥有单一文件系统的网格将会给远程文件操作带来很大的方便，比较明显的是可以很容易的作到对远程文件的存取、转移和检索。目前网格中还没有建立文件系统，所有对文件的操作都需要在本地和远程两端进行拷贝，影响了网格系统的工作效率，也不利于文件的管理和操作，也影响到网格入口软件的信息服务管理[26]。
- 网格入口软件中的用户定制问题。我们所设计的 WebGrid 目前主要针对科学计算用户设计的入口软件，很多地方还很不完善，下一步的工作包括个性化的使用模式设计，对不同的用户提供不同的使用模式，以更充分提高系统的运行效率；扩展网格入口框架与其他网格框架实现无缝结合等[6][7]。

附录 攻读学位期间发表的主要学术论文

【1】张云锋，葛玮《基于服务质量的网格工作流算法优化》计算机科学 2004.9

【2】张云锋，葛玮 《An improvement on Algorithm Of Grid-Workflow Based on QoS》武汉大学学报英文版 2004.11

参考文献

【1】 Ian Foster. What is the Grid? A Three Point Checklist http://www-fp.mcs.anl.gov/~foster/ArTicles/WhaTisTheGrid.pdf, 2002.

【2】 都志辉，陈渝，刘鹏，《网格计算》 清华大学出版社 2002.8

【3】 Foster I, Kesselman, C Jeffrey M. Nick, Tuecke S. The Physiology of the Grid- OGSA. http://www.globus.org/research/papers/ogsa.pdf, 2002.

【4】 Foster I, Kesselman , The Physiology of the Grid An Open Grid Services Architecture for Distributed Systems Integration

【5】 Globus Project .http：//www.globus.org.

【6】 何戈，《WebCom:一个基于 Web 技术的计算网格入口工具包》计算机研究与发展 2004.1

【7】 John P. Morrison, Sunil John, David A. Power, Neil Caerkey and Adarsh Patil Centre for Unied Computing Dept. Computer Science, National University of Ireland, Cork, Ireland A Grid Application Development Platform for WebCom-G http://www.cuc.ucc.ie 2005 IEEE

【8】 顾见军，《网格 Portal 的设计与实现》硕士论文 四川大学 2003-4-30

【9】 任旭龙，《网格资源监控及其资源描述的研究》硕士论文 兰州理工大学 2003.6

【10】孙岱，《网格监控研究与监控服务系统的设计》 硕士论文 中南大学

【11】Baiyi Song, Carsten Ernemann, and Ramin Yahyapour User Group-based Workload Analysis and Modelling. Computer Engineering Institute, University Dortmund, 44221 Dortmund, Germany. 2005 IEEE

【12】林剑柠，《基于遗传算法的网格资源调度算法》计算机研究与发展 2004.12

【13】陈永柱，《中国气象应用网格门户的研究》硕士论文 国防科技大学 2004.3

【14】GridLab Project. http://www.gridlab.org.

【15】Jason Novotny, Michael Russell, Oliver Wehrens. GridLab and Application Portlets Design. https://gridport.npaci.edu/

【16】CoG Kit Homepage. http://www.globus.org/cog/.

【17】GridBus Project. http://www.gridbus.org.

【18】Rui Zhang, Steve Moyle and Steve McKeever, Stephen Heisig, Oxford University Computing Laboratory, IBM T.J. Watson Research Centre OGSA-based Grid Workload Monitoring 2005 IEEE

【19】刘东华，《面向网格的资源监控系统》硕士论文 中科院计算技术研究所 2002-06-01

【20】宋智礼 《应用网格技术实现校园网的资源监控系统》硕士论文 北方工业大学 2004-5-30

【21】李庆阳，《网格环境中的动态任务分配和调度算法的研究》硕士论文 黑龙江大学 2004-05-10

【22】郝克刚《NP 完全问题及有关的理论研究》 西北大学

【23】Ladislau Boloni, Damla Turgut and Dan C. Marinescuy, n-Cycle: a set of algorithms for task distribution on a commodity grid. Department of Electrical and Computer Engineering University of Central Florida

【24】孙济洲，《网格环境下任务调度的研究》硕士论文 天津大学 2004-01-01

【25】刘国恩，《基于 Globus 工具包的网格计算系统研究》硕士论文 北京航空航天大学 2004.2

【26】王小平，曹立明，《遗传算法》 西安：西安交通大学出版社 2002

【27】李敏强，寇纪松，林丹，李书全，《遗传算法的基本理论与应用》科学出版社 2002.3

【28】Ligang He, Stephen A. Jarvis, Daniel P. Spooner, David Bacigalupo, Guang Tan and Graham R. Nudd Mapping DAG-based Applications to Multiclusters with Background Workload. Department of Computer Science, University of Warwick 2005 IEEE.

【29】Vincenzo DiMarTino. MMililotTi Suboptimal scheduling in agridusing genetic algorithms Parallel Computing, 2004, 30:553-565

【30】王翠平，《分布式系统中的任务调度问题及遗传算法应用研究》硕士论文 青岛大学 2002.4

【31】钟求喜，陈火旺等，《基于遗传算法的任务分配与调度》计算机研究与发展 2000.8

【32】陈华，《网格的安全体系结构》硕士论文 西安电子科技大学 2004-01

【33】A GSS-API profile for security context establishment and message protection using WS-Secure Conversation and WS-Trust http://www.globus.org/ogsa/Security/

【34】Grid Certificate Extensions Profile Michael Helm, ESnet/LBL

【35】Mary R. Thompson, LBNLDoug Olson, LBNLRobert Cowles, Slashing Mullen, IBM Mike Helm, ESnet, CA-based Trust Model for Grid Authentication and Identity Delegation, Oct 2002

【36】李密，《基于 Globus 下的网格技术和安全分析》 硕士论文 北京邮电大学 2004.2

【37】徐杰，《基于代理的网格入口软件验证与授权机制研究》硕士论文 南京理工大学 2004.7

【38】陈亚玲，《基于代理的网格计算中间件的研究与实现》硕士论文 西安交通大学 2004.7

【39】姚怡，苏德富，周迎新，《网格安全代理系统-Myproxy》 广西大学

【40】Yunfeng Zhang, Ge Wei, 《An improvement on Algorithm Of Grid-Workflow Based on QoS》, Wuhan university journal 2004.11

【41】Menno Dobber, Ger Koole, Rob van der Mei Dynamic Load Balancing Experiments in a Grid, Vrije Universiteit Amsterdam, The Netherlands

【42】Vijay K. Naik Chuang Liu Lingyun Yang Jonathan Wagner. On-line Resource Matching for Heterogeneous Grid Environments. IBM T. J. Watson Res Center Computer Science Dept, University of Chicago IBM Software Group, Tivoli. 2005 IEEE

【43】杨新刚，《基于网格环境的资源存取的研究》硕士论文 四川大学 2003-04-20

【44】童端，《计算网格作业管理系统的设计与实现》硕士论文 西安交通大学 2004.3

【45】Marta Beltr´an and Jose Luis Bosque, Information Policies for Load Balancing on Heterogeneous Systems. DIET, ESCET, Rey Juan Carlos University, 28933 M´ostoles, Madrid, Spain 2005 IEEE

【46】Lars-Olof Burchard, Barry Linnert, J.org Schneider, Technische Universitaet Berlin, GERMANY. A Distributed Load-Based Failure Recovery Mechanism for Advance Reservation Environments 2005 IEEE

【47】张焱，裘聿皇，《基于遗传算法的考虑优先约束和负载平衡的多任务调度》 中国科学院自动化研究所

【48】Jennifer M. Schopf Mathematics and Computer Science Division, Argonne National Laboratory TEN ACTION WHEN GRID SCHEDULING

【49】张云锋，《基于服务质量的网格工作流算法优化》计算机科学 2004.11

【50】张云锋，《WebGrid———一个基于 Web 技术的计算网格工具包》计算机应用研究 2005.7

【51】A Software Installation Information Provider for MDS 2.x
http://www-unix.globus.org/toolkit/mds/gldap-imports/mds/info-providers/

【52】Creating New Information Providers
http://www-fp.globus.org/mds/creating_new_providers.pdf

【53】贾小珠，宋立智，赵玮，《深入浅出 XML》，人民邮电出版社 2001

【54】Gregor von Laszewski, Beulah Alunkal, Kaizar Amin,Jarek Gawor, Mihael Hategan, Sandeep Nijsure The Java CoG Kit User Manual Draft Version 1.1

【55】贾明，《严世贤 Linux 下的 C 编程》 北京：人民邮电出版社 2001

致 谢

论文完成之际，也是我将要离开学校的日子。回顾这三年的学习与生活，我不禁感慨万千。在这里，有收获的喜悦，有对校园的眷恋，也有对未来专业知识的渴求，但更多的是对给予我无私帮助的老师和同学的无限感激之情。

首先，我要感谢我的导师葛玮副教授。葛老师三年以来，不仅教了我许多专业知识，还教我做人的道理。尤其在我论文的写作过程中，他给了我很多中肯的建议和无微不至的指导，才使得我顺利地完成了论文的写作。

我要感谢的是郝克刚老师、华庆一老师、候红老师、鱼滨老师、龚晓庆老师、路小丽老师，他们的言传身教将会使我受用终生。

感谢软工所为我创造良好的上机与工作环境。

感谢我的朋友和家人，他们在学习和生活上给了我无私的帮助和关心。

感谢我的同学：刘峰、褚红伟、张瑞、郭胜旺等同学在日常的学习和生活中对我的无私帮助。

在此，谨以此文向以上关怀和帮助过我的所有领导、老师、同学表示衷心的感谢！