摘 要

网格计算是近年来得到快速发展的广域网络计算技术。网格计算环境相对于一般网络计算环境来说有着更为复杂的特征，如存在多管理域和站点自治，系统的动态性、异构性和通信延迟的不确定性更高等。

网格任务调度负责协调资源、调度作业，在网格环境里如何有效地实现任务调度是影响网格计算成功的最重要因素之一。近年来，网格环境下服务质量（QoS）问题渐渐受到关注，提供非平凡的（nontrivial）服务质量成为网格的一大特征。调度问题的实质，一方面是尽量满足用户 QoS 要求，另一方面则要尽力优化系统吞吐率、资源利用率等系统性能指标。因此，在网格任务调度中提供 QoS 支持就成为网格系统是否能够提供非平凡服务质量的关键。

网格任务调度中提供 QoS 支持的研究主要有两个方面的难点：一个是如何描述和度量网格 QoS；另一个是如何在调度中引入对网格 QoS 的支持。网格环境下的任务调度分为在线调度和批调度（离线调度）两种方式，在线调度算法以优化用户需求为调度目标，批调度以优化系统需求为调度目标。因此，在调度中引入 QoS 支持应根据调度方式的不同有所区别。

本文围绕在网格任务调度中提供 QoS 支持这一问题，开展了以下研究工作：

首先介绍了论文的研究背景，包括网格计算的概念、网格资源管理的概念以及网格任务调度在网格资源管理中的地位等；

然后，本文全面地综述了网格环境下的任务调度机制。其中，总结了网格任务调度的特点及目标；根据现有的研究状况对网格任务调度机制进行了分析；介绍了已有的调度系统；并指出网格任务调度中引入 QoS 支持的意义。

在本文的第三章，提出了基于层次的网格 QoS 描述方法。由于网格 QoS 在不同层次上的表现形式大相径庭，本文从应用、系统、资源这三个层次来看待基于 QoS 的网格资源管理，提出了从网格应用层、虚拟组织层、资源设备层这三个层次的网格 QoS 描述方法，并展现了三个层次的 QoS 之间的映射关系。

随后，为了解决任务调度中多个性能目标情况下的 QoS 度量问题，在基于层次的 QoS 描述方法的基础上，本文进一步提出了一种 QoS 度量方法，该方法将虚拟组织层各类 QoS 定量地描述出来，并与应用层用户满意度相联系，从而解决了网格 QoS 的统一度量问题。

之后，基于层次化的网格 QoS 描述与度量方法，本文的第四章分别针对在线调度和批调度提出了提供 QoS 支持的任务调度算法，使得在线调度能够综合性地考虑用户 QoS 需求，也使得批调度在优化系统性能的同时兼顾了用户 QoS 需求。模拟结果表明，提供 QoS 支持的在线调度算法在系统吞吐率与传统 MCT 算法平均相差 8%的情况下，用户满意度有平均 44%的提高；而提供 QoS 支持的批调度算法相对于传统 Min-Min 算法，系统吞吐率损失可以忽略不计，而用户满意度则平均提升了 17%。

在本文的最后给出了网格模拟器 GridQoS 的设计与实现细节。GridQoS 网格模拟器是基于 GridSim 模拟器的扩展实现，该模拟器能够模拟网格用户、网格任务、网格资源、网格信息系统等实体以及实体间的通信交互；同时，该模拟器提供对 QoS 描述的支持并提供对在线调度、批调度的模拟。该模拟器有效地解决了本文研究工作中的验证问题，同时它对于一般的网格环境模拟也具有普遍的应用意义。

关键词：网格，网格 QoS，QoS 度量，任务调度，在线调度，批调度，网格模拟器

Abstract

Grid computing is a fast developing wide area network computing technique. The Grid environment is more complex than in general ones, as resources are autonomous, owned by different individuals or organizations, and more geographically distributed, heterogeneous in nature.

Grid scheduling system is responsible for resource coordinated and job scheduling. In Grid environment, the efficiency of job scheduling is the key factor that affects the efficiency of Grid system. During these years, more and more attentions have been paid to QoS (Quality of Service) in Grid environment, and the ability to deliver nontrivial QoS became one of the key characters of Grid system. Usually, job scheduling means trying to satisfy user's QoS requirements and trying to optimize system performance, such as system throughput and resource utilization. Obviously, supporting QoS in Grid job scheduling is the key issue affects the Grid system deliveries nontrivial QoS.

The research for supporting QoS in Grid job scheduling has difficulties in two aspects: one is the description and measurement of Grid QoS, and the other is how to take Grid QoS into consideration during job scheduling. In addition, as job scheduling in Grid environment includes online scheduling mode and batch scheduling (off-line scheduling) mode, and online scheduling heuristics aim to optimize user's requirements while batch scheduling heuristics aim to optimize system's requirements, the way of supporting QoS in job scheduling should be different according to the different scheduling modes.

Focusing on the problem of supporting QoS in Grid job scheduling, this thesis carries out researches as follows:

Firstly, this thesis introduces the background of research, includes the concept of Grid, the concept of Grid resource management, and the role job scheduling acts in Grid resource management.

Secondly, this thesis summarizes the mechanism of job scheduling in Grid environment, which includes the summarizes for the characteristics and goals of Grid job scheduling, the analysis for Grid job scheduling mechanism according to the research up to date, the introduction of grid scheduling systems, and the emphasis on the problem of supporting QoS in Gird job scheduling.

In the third chapter of this thesis, a layered mechanism for Grid QoS description is proposed. Because of the QoS in different layers is quite different, this thesis proposes a description method for three layers includes Grid application layer, visual organization layer and physical layer. At the

same time, the mapping relationship among the three layer's QoS has been presented.

Moreover, in order to evaluate multiple performance factors, a QoS measurement method is also proposed in this thesis based on the layered QoS description method. This method solves the problem of unify Gird QoS measurement by the way of trying to measure the QoS metrics in visual organization layer and relating these metrics to the user satisfaction metric in application layer.

And then, based on the layered mechanism for QoS description and measurement, the QoS supporting job scheduling heuristics related to online scheduling and batch scheduling are proposed respectively in chapter four of this thesis. The heuristic for online scheduling enables the comprehensive consideration for user's QoS requirements, and the heuristic for batch scheduling enables the consideration for user's QoS requirements while aiming to optimize system performance. The simulation results from Grid simulator called GridQoS demonstrate that, for the QoS supporting online scheduling heuristic, compared to traditional MCT heuristic, its makespan decreases 8% in average while its user satisfaction increases 44% in average; for the QoS supporting batch scheduling heuristic, compared to traditional Min-Min heuristic, its makespan has almost no decrease while its user satisfaction improves 17% in average.

At last, GridQoS is designed and implemented for demonstration work. GridQoS is extended from GridSim simulator, and it supports the simulation of Grid entities like Grid user, Grid resource, Grid job and Grid information system, and also supports the communication behavior among Grid entities. GridQoS solves the simulation problem for this thesis effectively, and it can act as a simulator for general simulation of Grid environment.

Keywords: Grid, Grid QoS, QoS mearsurement, job scheduling, online scheduling, batch scheduling, Grid simulator.

图表目录

中国科学技术大学学位论文相关声明

本人声明所呈交的学位论文，是本人在导师指导下进行研究工作所取得的成果。除已特别加以标注和致谢的地方外，论文中不包含任何他人已经发表或撰写过的研究成果。与我一同工作的同志对本研究所做的贡献均已在论文中作了明确的说明。

本人授权中国科学技术大学拥有学位论文的部分使用权，即：学校有权按有关规定向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅或借阅，可以将学位论文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、汇编学位论文。

保密的学位论文在解密后也遵守此规定。

作者签名：张世美

2007年 6月 4日

第一章 绪 论

Internet 的出现使得人们能够大范围地共享各种信息，也使得人们比以往任何时候都更加渴望能够更广泛地共享各种资源。使用 Internet 作为低层，研究人员可以将很大范围上地理分布的异构计算机系统集合在一起形成一个大规模的计算平台。该领域的研究产生了一个新的软件体系结构，我们称之为网格(Grid)[1]。网格计算(Grid Computing)的概念来源于电网(Electrical Power Grid)，人们试图实现：类似于家用电器能够极为方便的从电网中使用电力资源，计算应用也能够便利地从一个大范围的分布的资源池中获取所需的各种计算资源。

近年来，国内外众多科研人员在网格计算方面开展了许多相关的研究工作，研究结果表明网格计算确实是一个可行的高性能广域分布式计算模型，同时这些结果也展示了许多急需解决的挑战性问题，其中，如何有效地管理网格资源、调度任务，如何向网格用户提供确保质量的服务都是影响网格计算是否成功的重要因素。

1.1 网格计算的概念及演变

网格计算是近年来得到快速发展的广域网络计算技术，人们试图将很大范围上地理分布的异构计算机系统集合在一起形成一个大规模的计算平台。逐渐地，这一目标深化为建立大规模计算和数据处理的通用基础支撑结构，将网络上的各种高性能计算机、服务器、信息系统、海量数据存储和处理系统、应用模拟系统、虚拟现实系统、仪器设备和信息获取设备（如传感器）等大型基础设施集成在一起，为各种相关应用开发提供底层技术支撑，构造一个功能强大、无处不在的计算设施。

网格计算是由最初的元计算发展而来[3]~[6]，目前为止，网格计算还没有一个十分精确的定义。Ian Foster 曾先后三次尝试给网格下定义。1999 年，Ian Foster 在与 Carl Kesselman 合编的著作[1]里认为：计算网格是用以提供可靠、一致、普遍而且便宜的高端计算能力的软硬件基本设施。那时候网格的范围还主要局限于计算网格，这也是网格的渊源所在。2001 年，文献[2]从更加宽广的范围，在更为抽象的层次上给出定义，认为网格主要涉及到动态、多机构的虚拟组织协调的资源共享和问题解决，关键的概念是在一组参与者（提供者和消费者）之间协商资源共享的安排并利用所得资源实现特定目标的能力。这时候的定义已经跳出了计算网格的范围，并且指出了网格的本质和根本目的是资源共享。2002 年，Ian Foster 在为 GRID TODAY[7] 撰写的特稿上给出了判断网格系统的三点标准，认为网格应该能够：

(1) 协调不服从于中央控制的资源；

(2) 使用标准、开放和通用的协议和接口；

(3) 提供非平凡的(nontrivial)服务质量。

这个标准是在 OGSA（Open Grid Services Architecture）[8]的背景下提出来的，强调的是实现资源共享的协议和服务。继 OGSA 网格体系结构之后，在 Globus Toolkit 4 [8]中，又

提出了一种新型的 WEB 服务框架，即 WSRF 框架，它通过尽可能多的使用现有 WEB 服务标准以降低服务本身的复杂性，Globus 项目希望 WSRF 能够成为网格服务的新标准。然而，WSRF 较 OGSA 并没有本质的变化，最大的不同在于面向服务的体系结构。这说明网格与 WEB 服务的整合趋势，也说明网格学术界希望网格在工业界逐渐拓展商业作用。

可见，网格还没有一个公认的、明确的定义，而且其内涵和外延也在不断的演变。下面是一个较为通用意义上的定义：

网格的核心问题是资源共享，网格就是在缺少中央控制、没有全知者(Omniscience)以及强的信任关系的情况下对于地理分布的各种大型基础设施的协同使用。

1.2 网格资源管理

网格计算环境相对于一般网络计算环境有着更为复杂的特征，如存在多管理域和站点自治，系统的动态性、异构性和通信延迟的不确定性更高，硬件和软件两个层次上都存在异构性等等。因此，实现有效的网格计算有很多需要解决的问题，具体包括资源管理、任务调度、系统安全、编程模式、性能评测和数据存取等。

1.2.1 网格资源管理的概念

网格资源是指网络上能够被共享和利用的任何能力[9]，包括传统意义上的物理资源，如计算资源、带宽资源、存储资源、传感器、特殊仪器等，也包括虚拟的服务，如数据库、数据传输、仿真等软件应用。

网格资源管理是网格的核心组件，其核心功能是识别资源需求，匹配和分配资源，调度和监视资源，从而尽可能高效地利用资源[10]。网格资源管理侧重于控制网格资源如何提供能力及可用服务给其它请求者，并不关心资源的功能，而是该功能工作的方式。

1.2.2 网格资源管理的特点及要求

传统计算系统的资源管理已经得到了充分的研究，如批调度器、工作流引擎和操作系统。网格资源管理与传统资源管理的目标都是高效、合理地利用资源，且都具有并发性、共享性和随机性的特点，但他们之间有着很大的不同，他们的根本区别在于网格资源具有广域分布性、异构性和动态性。

传统的资源管理系统在单台计算机或小规模局域网范围内上运作，对资源能够完全控制，故可在与外界隔离的情况下实现高效的管理机制和调度策略。网格资源管理的应用背景是开放的广域网，它对资源不能够完全控制，对资源的状态变化不可预料，而异构的资源也使得资源管理任务更加地复杂化。这些问题使得传统的资源管理工具无法胜任网格系统的资源管理与调度任务。

网格系统的特点及其对网格资源管理系统GRMS(Grid Resource Management System)的要求体现为以下几个方面：

- 资源的分布自治性。资源跨多个管理域，地理上分布、自治，规模庞大，GRMS需要提供资源、用户的全局命名空间；资源的共享与私有并存，不同的管理域有不同的本地资源管理系统LRM (Local Resource Management)，GRMS对资源的没有完全的信息和控制，需与LRM交互，协同使用资源。
- 资源的异构性。资源的种类繁多，有不同的类型和不同的性能特征，相对于传统的资源管理系统，其异构性更强。GRMS要解决异构环境下对相似资源的配置和管理，定义标准的资源管理协议。
- 技术的多样性。资源属于不同的机构，不同的机构对资源有不同的使用、调度策略和安全机制。GRMS需要通过一种标准机制对资源和用户需求进行描述；并且，资源管理框架要足够的灵活，易于扩展，以支持新的策略和机制。
- 参与者目标的不一致性。参与者众多，资源使用者与资源提供者往往有不一致甚至矛盾的目标、策略和需求模式。GRMS需要提供相关机制实现他们之间的协商及不同目标的平衡。
- 动态性与自适应性。用户需求和资源状态动态变化：资源提供者与资源消费者的身份、需求可能发生变化，资源信息具有不确定性，资源的配置和能力也在动态变化。GRMS需要具有一定的自适应性，有效且高效地利用资源，并具备一定的容错处理能力。
- 资源的协同性。很多网格任务需要跨域协同分配、使用多个资源[7]。GRMS需完成跨域的资源协同分配，这需要了解不同域的安全机制、资源特征，进行任务的跟踪，并处理各种形式的错误。

综上所述，网格环境的特殊性为网格资源管理带来了诸多新的挑战，需要进一步对资源管理系统的相关机制进行研究。

1.2.3 网格任务调度问题

在网格系统中，任务调度是资源管理的核心组成部分。任务调度的实质就是按照一定的策略为任务寻找合适的资源的过程。网格任务调度可以定义为将网格作业映射到各管理域的网格资源的过程，而调度通常是以整个执行时间（吞吐率）、资源利用率及使用者和应用需求等指标为优化目标。一个典型的调度器在网格环境中的应用示例如图 1-1 所示。

一个网格系统的有效性在很大程度上取决于它的调度执行的有效性和效率。由于网格作业本身的异构及需求的不同，其所需资源在广域上分布、本质上异构、由不同的个人或组织拥有、有不同的存取和花费模式、负载和可用性动态变化，这些复杂特征使得网格环境下的任务调度机制研究更具复杂性和挑战性，也使得任务调度成为网格研究中的一个热点与难点问题。

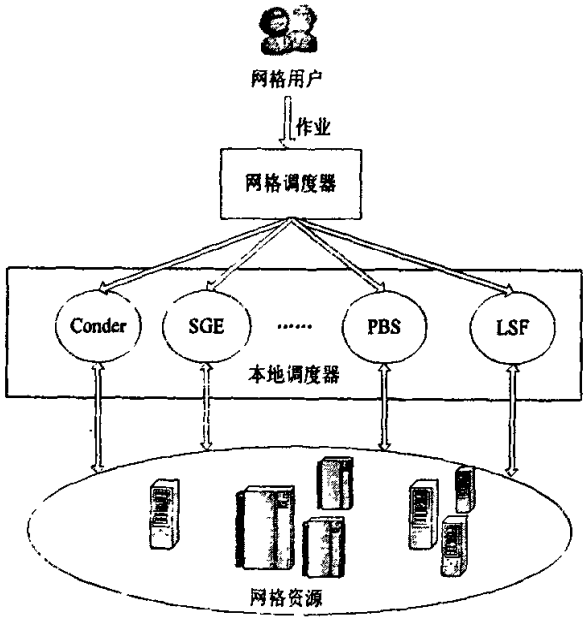

图 1-1 调度器在网格环境中应用示例

1.3 提供 QoS 支持的网格任务调度研究

1.3.1 网格 QoS 问题的提出

网格计算的初衷，是人们希望能够实现廉价、普遍的高性能计算，能够合作存取各种数据信息，能够提供广域多媒体应用等。随着对网格计算的进一步认识，人们发现，网格计算环境相对于一般网络计算环境有着更为复杂的特征，如存在多管理域和站点自治，系统的动态性、异构性和通信延迟的不确定性更高，硬件和软件两个层次上都存在异构性等。这些复杂性给网格系统的具体实现带来了很大的困难，尤其是在多个服务协作使用资源时，性能和效率也容易造成降级，因此，网格系统的服务质量难以得到保障。

随着网格应用的深入，人们不再满足于目前基于“尽最大努力”的网格基础设施，渐渐地，人们希望网格系统能够提供确保的服务质量，因此，网格 QoS 问题应运而生。在近年来的网格研究工作中，OGSA(open grid services architecture)[11],OGSI(open grid services infrastructure)[12]以及 Web 服务资源框架(WS-resource framework,简称 WSRF)[13]都要求所有的网格服务能够交付无缝的 QoS。

但是，目前的网格 QoS 研究尚处于起步阶段，其基本的问题定义、具体描述、参数量化等尚不清晰。

1.3.2 在任务调度中提供 QoS 支持的意义

在网格环境里如何有效地实现任务调度是影响网格计算是否成功的最重要因素之一。由于网格作业本身的异构及需求的不同，其所需资源在广域上分布、本质上异构、由不同的个人或组织拥有、有不同的存取和花费模式、负载和可用性动态变化，这些复杂特征使得网格环境下的任务调度机制研究更具复杂性和挑战性。与此同时，随着网格应用需求的扩展以及网格面向服务架构的提出，如何在网格系统中为用户提供 QoS 支持也成为当前网格研究中的一个热点问题。

在网格的应用场景下，从用户的角度而言，提交作业的过程总是伴随着对某些服务质量的期望（例如，最基本的，用户往往希望尽快得到作业执行结果）；同时，从系统的角度而言，处理作业的过程总是希望最大化系统吞吐率、最大化资源利用率。网格系统的一大特征是提供非平凡的服务质量，网格调度系统承担了协调资源、调度作业的任务，它是网格系统是否能够提供非平凡服务质量的关键。因此，研究在网格任务调度中如何提供 QoS 支持就具有了非常重要的意义。

1.4 本文的主要工作

基于上述考虑，本文主要研究网格环境下提供 QoS 支持的任务调度问题。主要工作介绍如下：

1. 全面地综述了网格环境下的任务调度机制。其中，总结了网格任务调度的特点及目标；根据现有的研究状况从调度组织模式、应用模型、状态估计、任务调度算法几个方面对网格任务调度机制进行了分析；介绍了已有的调度系统；并指出网格任务调度中引入 QoS 支持的意义。
2. 提出了基于层次的网格 QoS 描述方法。由于网格 QoS 在不同层次上的表现形式大相径庭，本文从应用、系统、资源这三个层次来看待基于 QoS 的网格资源管理，提出了从网格应用层、虚拟组织层、资源设备层这三个层次描述网格 QoS 的方法，并展现了三个层次 QoS 之间的映射关系。
3. 提出网格 QoS 的统一度量方法。评价一种资源管理算法抑或任务调度算法是否合理，针对单个性能目标是片面的，而应当综合性的针对多个性能目标。针对多个性能目标的资源管理的关键问题在于找到一种度量机制，该机制能够同时反映多个 QoS 目标，并体现其内在联系。在基于层次的 QoS 描述方法的基础上，本文进一步提出了一种 QoS 度量方法，该方法将虚拟组织层各类 QoS 定量地描述出来，并与应用层用户满意度相联系，从而解决了网格 QoS 的统一度量问题。
4. 提出网格环境下提供 QoS 支持的任务调度算法。本文基于层次化的网格 QoS 描述与度量方法，分别针对在线调度和批调度提出了提供 QoS 支持的任务调度算法，使得在线调度能够综合考虑用户 QoS 需求，也使得批调度在优化系统性能的同时

兼顾了用户 QoS 需求。模拟结果表明，提供 QoS 支持的在线调度算法在系统吞吐率与传统 MCT 算法平均相差 8%的情况下，用户满意度有平均 44%的提高；而提供 QoS 支持的批调度算法相对于传统 Min-Min 算法[44]，系统吞吐率损失可以忽略不计，而用户满意度则平均提升了 17%。

5. 设计与实现网格模拟器 GridQoS。网格环境中，由于资源的动态、异构性，以及资源和用户分布在许多组织中分别有自己的规则，所以几乎不可能可重复地、可控制地测试某个调度算法的性能，于是结果也是无法比较和估计的。GridQoS 网格模拟器是基于 GridSim 模拟器[49]的扩展实现，该模拟器能够模拟网格用户、网格任务、网格资源、网格信息系统等实体以及实体间的通信交互；同时，该模拟器提供对 QoS 描述的支持并提供对在线调度、批调度的模拟。该模拟器有效地解决了本文研究工作中的验证问题，同时它对于一般的网格环境模拟也具有普遍的应用意义。

本文的研究工作得到了国家自然科学基金项目——基于计算市场模型的网络资源管理研究（编号：60273041）和网格计算环境中信任感知的资源交易模型（编号：60673172），以及国家 863 项目——合肥网格节点的建设及若干典型网格应用的研制（编号：2002AA104560）的支持。

1.5 本文的组织结构

本文共分六章，组织结构如下：

第一章为绪论，主要介绍本文工作的背景和问题的提出。首先阐明网格的概念；然后概括性介绍网格环境下的资源管理、任务调度问题，并指出网格 QoS、提供 QoS 支持的调度问题的重要意义；最后，概括介绍本文的工作内容和组织结构。

第二章重点介绍网格环境下的任务调度机制。首先介绍网格任务调度的特点和目标，然后介绍现有的网格任务调度模型以及已有的调度系统，最后指出在任务调度中引入 QoS 支持的意义。

第三章研究了网格 QoS 的描述与度量问题，首先区分性地介绍网格 QoS 与网络 QoS，并具体描述网格应用的 QoS 需求；进而提出基于层次的网格 QoS 描述方法，并由此提出了一种统一的度量方式以解决网格 QoS 的度量问题。

第四章基于第三章网格 QoS 的描述、度量工作，分别针对在线调度、批调度提出了提供 QoS 支持的任务调度算法，并基于 GridQoS 进行了模拟和验证。实验结果表明，本文提出的调度算法相对于传统调度算法能够更优地适应用户和系统的需求。

第五章详细介绍了 GridQoS 网格模拟器的设计与实现。首先，介绍了 GridQoS 设计的起因，然后通过体系结构、具体实现细节对该模拟器进行详细介绍，最后以本文的模拟验证工作为例，给出模拟实例。

第六章总结全文，并对将来继续进行的工作进行展望。

第二章 网格环境下的任务调度机制

网格环境下的任务调度机制是影响网格计算成功的最重要因素之一。区别于本地调度，网格任务调度的对象往往跨越多个管理域，各管理域隶属于不同的组织，采用不同的策略操作其资源；而调度中涉及的网格用户和网格资源提供方的目的也可能不一致甚至相互抵触。由于网格任务调度面临的是一个NP完全问题，它引起了众多学者的关注,成为目前网格计算研究领域的一个焦点。本章将介绍网格环境下的任务调度机制。

2.1 网格任务调度的特点及目标

在网格系统中，任务调度系统是其重要的组成部分，它要根据任务信息采用适当的策略把不同的任务分配到相应的资源节点上去运行。由于网格系统的异构性和动态性，以及运行于网格系统之中的应用程序对于资源的不同需求，因此，网格任务调度相比本地调度更加复杂。

2.1.1 网格环境下任务调度的特点

网格任务调度系统具有以下几个特点：

1) 任务调度是面向异构平台的。网格系统是由分布在Internet上的各类资源组成的，包括各类主机、工作站甚至PC机。它们是异构的，可运行在UNIX，Windows NT等各种操作系统下，也可以是上述机型的机群系统、大型存储设备、数据库或其它设备。因此网格系统中的任务调度必须面向异构平台，并在这些平台上实现网格任务的调度。
2) 任务调度不干涉网格节点内部的调度策略。在网格系统中，各网格节点的内部调度策略是自治的，网格任务调度系统干预其内部的调度策略是没有必要的，也是不可能的。
3) 任务调度必须具有可扩展性。网格系统初期的计算规模较小，随着超级计算机系统的不断加入，系统的计算规模也必将随之扩大。因此，在网格资源规模不断扩大、应用不断增长的情况下，网格系统的任务调度必须具有可扩展性，不致降低网格系统的性能。
4) 任务调度能够动态自适应。网格中的资源不但异构且网格的结构总是不停地改变：有的资源出现了故障，有的新资源要加入到网格中，有些资源重新开始工作等。总之网格的动态性明显，因此任务调度系统必须适应网格的这种动态性，从可利用的资源中选取最佳资源为用户提供应用服务。

2.1.2 网格环境下任务调度的目标

简单地说，网格任务调度的目标就是要对用户提交的任务实现最优调度，并设法提高网

格系统的总体吞吐率。具体的目标包括：最优跨度(Optimal Makespan)、服务质量QoS (Quality of Service)、负载均衡(Load Balancing)、经济原则(Economic Principles)等。

1) 最优跨度。跨度是一个最主要、最常见的目标，指的是调度的长度，也就是从第一个任务开始运行到最后一个任务运行完毕所经历的时间。它也常常作为衡量系统吞吐率的指标。跨度越短说明调度策略越好。当用户向网格系统提交任务后，最大的愿望是网格系统尽快完成自己的任务。可见，实现最优跨度是用户和网格系统的共同目标。
2) 服务质量QoS。网格系统要为用户提供计算和存储服务时，用户对资源需求情况是通过QoS形式反映出来的。任务管理与调度系统在进行分配调度任务时，保障网格应用的QoS是完全应当的。
3) 负载均衡。在开发并行和分布计算应用时，负载均衡是一个关键问题。网格系统更进一步扩展了这个问题。网格任务调度是涉及交叉域和大规模应用的调度。解决好系统的负载均衡是一个非常重要的问题。
4) 经济原则。网格环境中的资源在地理上是广泛分布的，而且每个资源都归属于不同的组织，都有各自的资源管理机制和政策。根据现实生活中的市场经济原则，不同资源的使用费用也应是不相同的。市场经济驱动的资源管理与任务调度必须使消费双方（资源使用者和资源提供者）互惠互利，才能使网格系统长久地发展下去。

2.2 现有网格任务调度机制研究状况

现有的网格任务调度研究一般从组织模式、应用模型、状态估计、调度算法这几个方面展开，因此，我们从这几个方面出发，系统介绍网格任务调度机制。

2.2.1 网格任务调度的组织模式

目前网格计算环境中所使用的网格资源组织模型主要有如下几种：

1) 分层模型(Hierarchical Model) [15]~[17]。目前采用分层模型的网格项目有 Globus，Legion，DataGrid，AppLes。
2) 抽象所有者模型(Abstract Owner Model) [15]。目前还很少有实际的网格项目采用该模式。
3) 计算市场（经济）模型(Computational Market/Economy Model) [17]。网格计算市场提供了合适的工具和服务，允许资源请求者和资源提供者表达他们的需求，它可以和前两种模式结合使用来实现更有效的资源管理。
4) 混合模型是以上几种模型的组合，如在计算市场模型里可以采用分层模型的结构来组织分层的市场，抽象所有者模型里也可以采用市场的某些概念来协商资源。
5) 资源池模型。这种模型，如 Condor-G[18]，在一台中心服务器上记录计算环境中所有资源的信息，资源池中对资源的描述是无序的，因此在 Condor 环境中只支持工作级调度，

不支持任务级并行，也不支持任务之间的通信。

6) 网格资源空间的 EVP 模型。在织女星网格[19][20]中提出的 EVP 模型包括有效资源层(E)、虚拟资源层(V)和物理资源层(P)。

7) OGSA 资源服务模型。OGSA 已成为网格资源访问接口的事实上的标准，这种标准化接口所带来的优势是显而易见的，除了平台无关性外，接口与实现的分离给用户的使用和资源的管理提供了方便。

与资源组织模型相对应，当前的网格系统一般都使用分布式调度算法，即网格中多台机器能在执行调度的同时使用分布式算法实现资源信息发布。网格环境下的调度技术主要可以分为以下几类：基于“超级调度者”的方法、基于市场的方法、基于发现的方法、以及由这几种方法组合的混合技术[21]。

- 在基于“超级调度者”的方法中，网格环境下存在分层的多个调度器，它们互相协作执行资源管理。层次较高的资源控制者调度较大的单元，而较低层次的资源控制者调度较小的单元，逻辑上来说每层只有一个请求和任务队列。一个单层的控制者模式和一个集中的控制模式其实是相同的。在一个完全分布的方法中不存在专门的资源控制者机器，资源请求者和资源提供者直接确定资源的分配和调度，一般会存在多个不同的请求和任务队列。该种方法难于处理站点自治以及存在许多不同管理域和调度器的系统。更困难的问题是由不同调度器调度的资源的协作配置(coallocation)。由于诸如分布式的交互仿真类的应用为了达到要求的服务质量，要求多个资源的协作分配，协作配置是网格环境下的一个重要问题。解决此类问题是层次调度未来的进一步研究方向之一。
- 对于基于市场的方法，资源管理使用源于人类经济的原则执行。使用较多的技术是拍卖和多商品市场。在一个诸如网格的广域系统中，有可能必需使用充分分布式的拍卖机制来提供容错和可扩展性。使用这种技术的主要研究在于价格管理和资源的协作配置。
- 资源发现技术在一个分布式的数据库中维持资源属性和状态信息。这类方法之间的不同点在于它们更新、组织、或者维护分布式数据库的方法。广泛使用的 Globus 工具集使用一个集中式的发现算法的变种来寻找对于一个请求来说最合适的资源。若干基于代理的资源管理技术诸如 Infosleuth 也使用基于发现的技术。该方面的挑战是设计容错和可扩的高度分布式的发现技术。
- 混合技术使用一个双层机制。混合可以结合多重技术实现可扩和容错的方法。例如，网格可以在自治的管理域间使用一个分层的调度和在每一领域内使用发现技术。我们需要进一步的研究来权衡不同方案的组合。

由于网格计算环境是由所谓的元计算系统发展而来，目前的网格项目多采用基于“超级调度者”的层次式调度方法调度系统资源。基于市场的方法也得到越来越多的关注。

2.2.2 网格应用模型

将网格应用分解为多个子任务，每个子任务有各自的资源需求，然后提交到网格调度

系统执行调度，是目前网格调度系统采用的通用方法。网格应用模型对于网格调度和任务分配产生重要的影响，不同类型应用的约束条件不一样，尤其是网格环境由于没有完全集中的控制，在资源的分配和调度上对不同的应用模型的任务的处理方式不一样。它主要有两种不同的类型任务：①相互之间独立的任务，即 MetaTasks；②相互之间存在依赖和时序关系、因果关系的任务。

2.2.2 调度系统状态估计

调度问题是众所周知的 NP 问题，各种启发式算法需要了解系统目前的状态来作出调度决策。由于网格系统的动态性和大规模分布，系统的状态信息一般不可能完全得到。非预测模型不考虑历史信息，而仅仅使用启发式或者概率分配模型根据当前状态信息估计网格状态。预测模型通过对历史和当前信息的学习来推断系统未来的可能状态[22]。预测模型能更进一步分成启发式、出价模型和机器学习方法。启发式方法使用预先定义的规则基于系统目前和历史的信息作出状态估计。在出价模型方法中，资源使用市场动态购买和出售，因此可充分考虑资源可利用性和资源需求。在机器学习模式中，往往使用神经网络或者遗传算法来估计状态。

2.2.3 网格任务调度算法

目前在网格调度算法研究中，其目标主要是增加吞吐率，增加系统的使用率，实现经济系统和用户的约束条件，实现在整个系统中网格应用任务的完成时间最小，即使Makespan最小，找到一个这样的Makespan是NP完全问题。

网格环境下的任务调度按照任务间的通信关系可以分为：相互间存在通信的任务组的调度和相互独立的任务组(MetaTask)的调度两类。调度系统使用调度算法来确定请求以及任务调度的相关顺序。图 2-1 显示了调度算法的分类。对于相互间存在通信的任务的调度，最常用的方法是沿用传统的 DAG 图调度，即将相互间存在数据依赖的任务组当作一个有向无环图，根据此有向无环图作出调度决策。但对于大规模、多管理域的网格系统，DAG 图调度方式执行可能有相当的困难，实际上经常采用预置和合作配置方法来保证任务的执行。

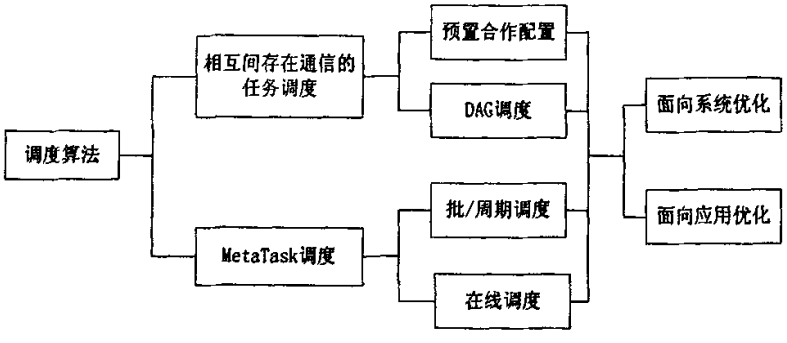

图 2-1　调度算法分类

MetaTask 调度根据调度的频度可分为周期性(批)调度以及事件驱动的联机在线调度。周期性(批)调度方法将资源请求和系统事件分为组，间歇地处理它们。调度的间隔可以是周期性的或者可能被一定的系统事件触发，关键在于任务调度是成批处理的而不是针对单个请求或者事件，事件驱动的联机调度在 GRMS 收到资源请求或者系统事件时执行调度。

批调度潜在地能得到更有效的网格资源利用率，因为能够考虑到更多的请求。对于周期性或者批调度，预测的状态估计也可以更好工作。但是由于服务层的问题不一定能立刻引起任务的调度，使用该方法很难实现硬 QoS 支持。联机方式反应更快，减少了任务的延迟时间，但很难对于不同类型的网格系统执行一个通用的联机 GRMS 方案。

大多数的调度算法都是面向系统的。面向系统的调度算法以最大化系统吞吐率为调度目标；而面向应用的调度算法则尽力优化某些特定的参数如应用完成时间。

实际的网格中可以执行调度组织、系统状态估计、和调度算法的不同组合。可以改变调度系统的一部分而不影响其它 GRMS 构件或其它调度构件。

2.3 现有网格调度系统介绍

下面分析现有的网格项目和系统实现的调度系统。

- Condor-G。Condor 是一种广泛使用的高吞吐率的批处理作业调度器，它与 Globus 项目合作开发了 Condor-G，它是一种访问远程批处理系统的应用网格的代理，利用 ClassAds 进行资源的发现和匹配，并采用抢先式再继续调度(Preemptive Resume Scheduling)来实现资源的动态变化。Condor 明确地管理客户和管理系统之间的任务服务等级协议(TSLA)。运行一个 Condor-G 作业需要六个步骤：①用户提交作业；②用户身份验证；③分析作业；④数据传输；⑤作业提交到作业队列；⑥作业执行。
- PBS，LSF 与 Silver/Maui [23]。PBS(Portable Batch System)是由 NASA 开发的灵活的批处理系统。它被用于集群系统、超级计算机和大规模并行系统，支持批处理作业、串行作业和 MPI、PVM 并行作业，提供了 FIFO 等多种作业调度策略，具有一定的作业监控能力和安全机制，但没有解决单一故障点问题，不提供检查点操作和进程迁移，要求对资源具有完全的控制。LSF 是 Platform 公司开发的负载分担设施(Load Sharing Facility)，作为本地调度系统其功能同 PBS 一样，即可作为网格调度体系结构中的底层调度用于完成批处理作业，实现负载平衡。如果以 GRAM 作为底层协议，也可以将 PBS，LSF 部署在网格层次上。Maui 为本地集群提供高性能作业调度，它作为现存资源管理的一个功能扩充模块，可以实现抢先式作业、公平调度、优先级管理、调度优化和 QoS 支持，它可以与 PBS、LSF 等进行集成。Silver 利用 Maui 实现了作业在多个集群间进行调度，能够提供负载平衡、协同分配和网格一级的 QoS 保证。
- 基于知识的元调度器(KB Metascheduler) [24][25]。KBMetascheduler 利用基于人工智能的多约束条件搜索技术来实现调度决策，它将作业的花费时间、用户要求、负载平衡、内存要求、缓存要求等信息用到专家系统中。该调度器是建立在 Globus Toolkit 的基础

上，利用了 Globus 的一些高级基础服务，如资源预留和信息服务。

- AppLes [26]。AppLes 是一个多用户分布异构环境下的高性能调度器，它关注应用的优先级，是一个多代理系统，每个代理负责一个特殊的任务。AppLes 代理利用异构应用程序模板允许用户指定应用的结构、特征和当前的实施、用户的指示(User Specification)提供关于用户的性能、时间和其它标准的信息。AppLes 考虑的是应用级的调度，应用是在基于主机系统的反馈基础上进行调度的。当系统发生变化时，AppLes 会重新考虑应用的调度过程，从而适应应用性能的变化。AppLes 包含的许多元素在基于性能信息的情况下支持资源分配，但是没有一个组件通过监控性能调整任务来支持反馈机制。
- Nimrod-G [27]。Nimrod-G 是澳大利亚 Monash 大学网格调度系统。它是一个模拟仿真系统，采用 Globus 中间件作为网格接口，执行系统对分布式参数计算问题基于经济模型提供一系列计算资源，能够基于最迟期限和预算(Deadline and Budget)进行调度工作。Nimrod-G 的计算经济目前还不是十分充分，它缺乏一种定义资源费用的全面策略。
- Vega Grid。织女星网格(VEGA)是由中国科学院计算开发的网格操作系统，引入服务网格的思想，资源调度中对于功能和性能的评价不只是采用传统计算机的评价标准(如速度、加速比、性能价格比等)，而是以用户满意度为目标，用类似 SLA(Service Level Agreement)的服务质量尺度来衡量。其作业管理机制对服务的使用仅限于单个的、孤立的服务的调用，缺乏动态地、易用地把多个服务组合起来，并且 VEGA 并不符合 OGSA 规范。

2.4 提供 QoS 支持的网格任务调度问题

一般来说，从用户的角度而言，提交作业的过程总是伴随着对某些服务质量的期望（例如，最基本的，用户往往希望尽快得到作业执行结果）。由于调度系统承担了协调资源、调度作业的任务，它一方面尽量满足用户 QoS 要求，另一方面则要尽力优化系统吞吐率、资源利用率等系统性能指标。因此，任务调度的有效性是网格系统能否提供非平凡服务质量的关键。

近年来，在任务调度问题的研究中，提供 QoS 的任务调度渐渐受到关注。文献[28]将网络吞吐率作为 QoS 参数嵌入 Min-Min 启发式算法，首次尝试在任务调度中考虑具体的 QoS 参数。但是，这项工作仅仅考虑了一维的 QoS 参数，并不能恰如其分地反映出用户的需求；文献[29][30]提到的 TITAN 系统，首次提出一种综合考虑系统吞吐率、任务完成时限、机器空闲时间三个指标的调度性能评测方式。该工作基于此评测模型，在不同情景下选择不同的传统调度算法。正如大多数调度系统，这项工作倾向于系统级的优化，并不能很好的考虑用户的需求。

综合上述研究情况，可见在任务调度中支持 QoS 这一研究领域尚不成熟。一方面，缺乏对网格 QoS 的全面、定量描述；另一方面，如何在现有的调度算法中支持 QoS 也有待进一步深入。因此，本文的主要工作围绕在网格调度中如何提供服务质量支持展开，在接下来

的第三章将对 QoS 的定性、定量描述，并在第四章详细介绍如何在调度算法中提供 QoS 支持。

2.5 本章小结

网格任务调度是决定网格系统能否提供非平凡服务质量的关键。本章详细介绍了网格环境下的任务调度机制。首先，介绍了网格任务调度的特点及目标；然后根据现有的研究状况，从调度组织模式、应用模型、状态估计、任务调度算法这几个方面详细描述了网格任务调度机制；之后了给出了已有的网格调度系统实例；最后描述了提供 QoS 支持的网格任务调度问题并指出其重要的研究意义。

下一章将具体研究网格环境下的 QoS 描述与度量问题。

第三章 网格环境下 QoS 的描述与度量

众多商业和科学计算应用都需要访问昂贵的高端资源，但这些资源往往是有限的。网格作为一种资源共享技术使得资源持有者与用户能够共享资源，而其中如何有效使用资源以满足特定应用需求渐渐受到网格研究的重视。对某一些需求独特的应用，比如，可视化应用，其图形显示需要仿真结果在特定时间段内返回，针对这些应用场景，网格研究者着力于资源选择和任务调度开展了众多的研究。尽管如此，直到“面向服务架构”的提出[8]，网格研究界才开始真正考虑网格环境下为用户提供确定的服务质量支持的问题，而提供非平凡的服务质量也成为了网格的一大目标[7]。

但是，就目前来说，大多数的网格环境仍然基于“尽力而为”（best effort）这一理念为用户提供服务，而共享资源的用户对服务质量的需求得不到正确、全面描述和有效满足。当有大量的请求抵达的时候，网格系统性能以及服务有效性的下降成为一个迫切需要解决的问题。针对这个问题，一个解决方式即是引入服务质量（QoS）机制使得服务提供者基于服务标准（诸如优先级、公平性、经济因素等）区分服务。

本章的工作集中在网格环境下服务质量（QoS）描述与度量。首先，回顾传统意义上 QoS 概念（第一节），然后阐明网格环境下 QoS 问题范畴（第二节），并对现有网格 QoS 研究状况进行介绍（第三节），之后提出基于层次的网格 QoS 描述方法（第四节）并基于此提出网格 QoS 度量方法（第五节），最后就本章内容进行小结。

3.1 传统意义上的 QoS 概念

从计算机系统诞生伊始，人们就一直孜孜不倦地致力于提高系统的服务性能和服务质量，因此，QoS 问题由来已久。而通常意义上的 QoS 问题往往是指计算机网络的 QoS 问题。

QoS（quality of service），即服务质量。它有多种等价或互补的定义形式。

RFC2386[31]中描述为：QoS 是网络在传输数据流时要求满足的一系列服务请求，具体可以量化为带宽、延迟、延迟抖动、丢失率、吞吐率等性能指标。此处的服务具体是指数据包（流）经过若干网络节点所接受的传输服务，强调端到端（end-to-end）或网络边界到边界的整体性。QoS 反映了网络元素（例如，应用程序、主机或路由器）在保证信息传输和满足服务要求方面的能力。

另一种描述[32]为：QoS 是指发送和接收信息的用户以及用户与传输信息的综合服务网络之间关于信息传输的质量约定。该约定可以被理解为服务提供者与用户之间的一份服务契约，即服务提供者承担支持给定的服务质量，当且仅当用户按照约定的信息流特征产生数据。换句话说，服务质量包括用户的要求和网络服务提供者的行为两个方面，是用户与服务提供者两方面主客观标准的统一。用户的要求是指用户在 Internet 上进行多媒体通信时所要求的服务类型以及相应的传输性能和质量等；网络服务提供者的行为则指 Internet 针对某一类服

务所能提供和达到的性能与质量。

QoS 控制的目标是为 Internet 应用提供服务区分和性能保证：服务区分是指根据不同应用的需求为其提供不同的服务；性能保证则要解决诸如带宽、丢失、延迟、延迟抖动等性能指标的保证问题。

3.2 网格环境下的 QoS 需求

“提供非平凡的服务质量(QoS，Quality of Service)”是判断网格系统的三大准则之一。实际的网格应用都需要使用异构的、类型多样的资源。对这些资源共享使用时，各个任务间需要协作，从而使得资源的调度分配、任务执行变得复杂。目前的网格基础设施大都建立在“尽力而为”的基础之上，网格系统的服务质量难以得到保障。解决这些问题，就需要在网格系统中引入QoS机制。网格QoS问题区别于传统QoS问题之处在于，网格能够提供端到端QoS控制，不只是提供网络QoS。

本节从网格应用入手，讨论应用的QoS需求，进而总结网格环境下的服务质量需求特征。

3.2.1 网格应用的 QoS 需求

网格技术对科学研究、社会生活等众多领域有着广泛影响，它的出现推动了许多高性能应用的发展，如：分布式超级计算、高吞吐率计算、数据密集型计算等。这里，从中国科学技术大学的高性能应用学科背景调研出发，并结合现有的网格应用情况，介绍几种典型的网格应用及其QoS需求：

- 海量数据集的远程数据可视化

  科学计算仿真要处理数十TB甚至PB的数据，有效地对这些数据进行远程研究，可通过把可视化分解为几个管道进程来实现。举个例子：计算可在一些机器上完成，计算结果可递交给另一台机器，由它将数据可视化并将结果以视频流的方式发送给所有客户。这些应用对带宽、延时和抖动极其敏感，涉及的网格资源包括存储、网络、CPU和可视化引擎等。

- 海量数据传输

  这类应用须将大量的数据从一个站点传输到另一个站点以存储和分析。它的QoS需求是数据传输必须在某个时间底线前完成，而不关心瞬时的传输速度。为此，需要对数据传输所涉及的存储系统、CPU和网络进行协调调度，以达到极高的总体传输率。

- 高性能合作环境

  这些环境包括虚拟沉浸、虚拟现实系统，需要高分辨率显示以及多站点连接和多交互形态，如音频、视频、跟踪和数据交换等。由于对交互性的需求，这些应用要求QoS机制允许描述和管理这些不同类型传输流的显式特征。

3.2.2 网格系统需要满足的 QoS 要求

为保证网格应用的端到端QoS，需要网格QoS机制能够满足以下要求：

- 不同特征混合流的支持

  远程可视化、科学数据分析、科学仪器在线控制和远程沉浸等领域中的网格应用经常混合着大量不同特征的多个流，这些特征包括带宽、延时、抖动、可靠性等。他们要求网格QoS机制能支持这些流，并允许具有不同特征流的共存，同时需要网络QoS技术为应用提供可靠的网络数据传输服务的保证。

- 统一的资源预留和分配

  高性能应用对一些特别的资源需求很大，如高带宽虚拟通道、科学仪器和超级计算机等。需要新的概念和构造，为属于不同管理域的多种异构资源实现统一形式的提前和直接预留，以及多资源联合分配，以满足网格应用使用资源和提供服务时的性能保证。

- 安全性

  网格QoS机制应当是安全的，可使用网格中统一的安全机制，对QoS服务的访问进行验证、鉴别和授权，防止非法用户侵入或篡改有关预留、算法等的信息。

- 轻便性

  网格QoS机制应是轻便的，尽量使用轻型的协议，不能过分加重网格资源的负担，特别要避免频繁的通讯而影响网络的性能。

- 强壮性和可扩展性

  由于网格是一个全球规模的基础设施，连接着巨大数目的动态网格资源，随时都有新资源加入和旧资源退出，这要求网格QoS机制能适应网格的动态变化，具有强壮性和可扩展性。

3.3 网格 QoS 的描述问题及其研究现状

网格QoS是一个综合指标，它用于衡量用户使用一个服务的满意程度，不仅仅停留于单一的性能参数要求。它往往描述了一个服务的某些性能特点，而这些性能特点是用户可见的。它以用户可理解的语言表述为一组参数，是对服务提供者服务水平的一种度量和评价。

现有网格服务质量保证研究中，对如何描述和度量具体的QoS参数，还没有统一的方法，本节重点介绍这方面的相关工作。

Foster等学者在文献[33][34]中提出了支持资源预留和协同分配的网格资源管理体系架构GARA (general-purpose architecture for reservation and allocation)。由于GARA面向以计算为中心的网格环境，在GARA的有关QoS的控制接口中，针对的是最终的物理资源。但在以服务为中心的OGSA和WSRF中，物理资源对用户是透明的，用户只能针对网格服务或虚拟组织中的逻辑资源提出QoS需求。因此，单纯针对某个层次进行QoS研究并不完整。一方面，

需要区分QoS的层次，另一方面，需要将针对服务的QoS需求映射转换为物理资源的QoS参数。所以，需要对网格QoS层次结构中各层之间QoS参数间的映射关系进行研究。

在文献[35]中，Roy等学者提出了一种新的消息传递架构MPICH-GQ，它综合了QoS描述、预留和实现技术，从基于消息传递编程的角度给出了网格QoS的一些实现机制，并涉及到了网格QoS层次的问题。在MPICH-GQ的系统架构中，通过MPI QoS Agent实现应用层QoS参数到低层次QoS参数的映射转换，但他们没有对网格QoS的层次结构进行更进一步的研究。

文献[36]将网格QoS的参数划分成Accounting QoS，Service QoS，Provisional QoS，Service Reliability和Service Security这5种类型。这种划分方案反映了网格QoS的一些特性，较全面地考虑了网格QoS参数在服务代价、安全和可靠程度等方面的需求，但它也存在着值得商榷的方面。比如，在用户首先要保证服务代价的情况下，Accounting QoS就要定义为对QoS起主要的决定性作用的硬QoS，即属于Service QoS。因此，Accounting QoS就会与Service QoS发生重叠，从而无法保证5种类型的QoS参数是正交的。

文献[37]~[39]提出了分布异构网络的性能量化框架，该框架对分布异构网络中的优先级、版本、完成期限和安全等QoS参数进行了度量。但是该框架仅代表性地度量了几个网络中的QoS参数，缺乏度量网格QoS参数的高效机制。然而，基于此框架可以将虚拟组织层的QoS参数度量出来，以利于它们相互之间的比较。

文献[28]在对Min-Min算法改进的基础上，提出了以QoS为导向的Min-Min网格资源分配算法，并且设计了相应的网格仿真系统。以QoS为导向极大地缩短了任务的响应时间，但是，该算法仅仅考虑了一维的QoS参数，这不能恰如其分地反映出用户的需求。在实际的网格系统中，常常需要多维的QoS参数来描述用户的需求。

文献[29][30]提到的TITAN系统，首次提出一种综合考虑系统吞吐率、任务完成时限、机器空闲时间三个指标的调度性能评测方式。该工作基于此评测模型，在不同情景下选择不同的传统调度算法。正如大多数调度系统，这项工作倾向于系统级的优化，并不能很好的考虑用户的需求。

综上所述，网格的早期工作提出了资源管理体系结构，它仅解决了跨越多个管理域带来的异构性问题，在满足用户需求和提高资源分配性能方面还缺乏深入研究（比如，大部分的资源调度算法仅仅考虑了时间因素，某些以QoS为导向的调度算法也只考虑了一维的QoS参数）。可见，目前对网格QoS的研究成果尚且不多，且存在一些值得研究的问题，如网格QoS所处的层次比较模糊、网格QoS的分类正交性较差、缺乏合理的QoS参数度量方案等。

3.4 基于层次的网格 QoS 描述方法

研究QoS必须分清层次，因为QoS在不同层次上的表现形式大相径庭。一个具有普遍意义的方法是从应用、系统、资源这3个层次来看待基于QoS的资源管理[40]。从应用的角度来看，应用需要利用一定量的资源来达到自己预想水平的QoS；从系统的角度来看，资源管理系统应当包含应用与资源等实体以及调度算法、安全算法等元素；从资源的角度来看，每个

资源只关心其自身的访问算法和运行于其上的应用，而并不关心其它资源的情况，资源与资源之间是相互独立的。

综上，在本文的工作中，网格QoS将从三个层面进行描述：低层的资源设备层、中间件虚拟组织(VO, Virtual Organization)层和上层的网格应用层，如图3-1所示。

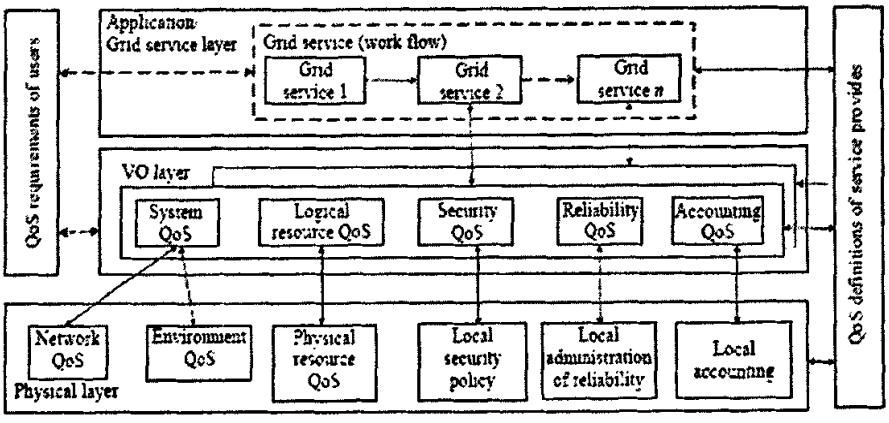

图 3-1　网格 QoS 层次结构

3.4.1 网格应用层的 QoS 描述

网格应用通过访问合适的网格资源来达到预定水平的QoS。它们不关心如何达到预定水平的QoS，也不关心在达到这一水平的过程中其它应用的情况。网格应用可能是一个单一的服务，也可能是由一系列的子服务所抽象成的网格服务工作流。对于网格应用是由子服务工作流组成的情况，服务提供者在确定网格应用的QoS参数时，需要从各子服务相关参数定义的语义出发进行统一抽象，确保高层服务能够获得一致的服务质量。因此，应用层需要抽象出网格应用的模型，以表达网格应用的结构及其分配资源的最小单元。另外，利用服务质量曲线将应用层QoS参数与用户满意度相关联，以描述资源提供者的QoS水平，它是VO层QoS度量的依据。

服务质量曲线描述如下：

定义1. 用户满意度：区间[0,1]内的实数，值越大表示用户对服务越满意，取值1表示用户需求得到完全满足；取值0表示用户的最低要求未被满足。

定义2. 用户满意函数：用来描述某个特定的QoS参数与用户满意度之间的映射关系。u_{ij} 表示第i个网格服务中的第j个QoS参数与用户满意度之间的映射关系。用户满意函数的定义域因不同的QoS参数而异，值域为[0,1]内的实数。

用户满意函数因用户提供的应用不同而有所不同，网格资源管理系统可以根据用户满意函数的性质来提供尽量优质的服务。另外，用户满意函数也因QoS参数的不同而呈现不同的性质。例如，如图3-2(a)所示，用户对CPU、带宽的需求越高越好，因此用户满意函数可能是一条无限逼近于1的曲线；用户对外存的要求可能有一个最小值，超过这个值用户就得到

完全满足，所以，此时用户满意函数可能是折线型的，用户满意度在最小值前为0，最小值后为1，如图3-2(b)所示；在采用页式管理的操作系统中，用户对内存的需求与页面的大小有关系，分配给用户内存大小必须增加到一个页面大小时才增大用户满意度，所以，用户满意函数可能是阶梯型的折线，如图3-2(c)所示。

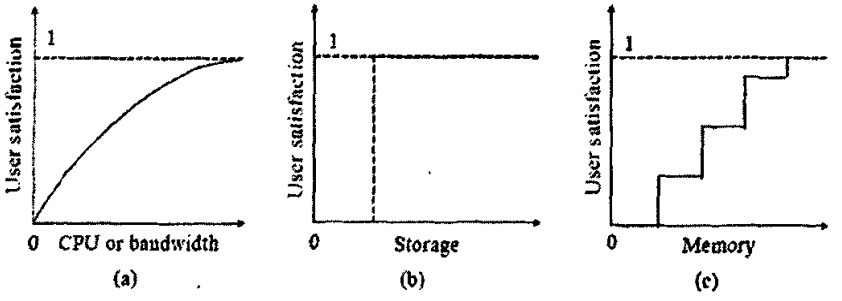

图 3-2　用户满意函数的例子

3.4.2 虚拟组织层 VO 的 QoS 描述

中间层是承上启下的VO层。VO层将用户提出的QoS需求映射到某一类的网格QoS，并将物理层中相似的QoS参数整合为一类。这样，VO层不仅屏蔽了物理资源的异构性，还降低了物理层QoS的复杂性。为此，我们对VO层网格QoS的分类进行了研究。由于VO是连接网格服务和物理资源的纽带，是网格特性得以体现的关键，因此，从VO层对网格QoS参数进行分类，一方面可以很详细地表达网格QoS参数的特性，另一方面可以很方便地将VO层的QoS参数分别映射转换成网格应用层和物理资源的QoS参数。

根据QoS参数所表达的QoS特性的不同，我们在VO层将网格QoS参数分成了逻辑资源类、系统类、安全类、信任类和记账类5种类型。

- 逻辑资源QoS

  逻辑资源QoS是由服务提供者提供的用来描述虚拟组织中直接提供网格服务的逻辑资源的QoS参数。它综合考虑了物理资源本身的整体服务性能、资源本地任务的负载情况以及资源的共享算法等因素，是物理资源在网格环境中服务性能的抽象。逻辑资源QoS是网格服务QoS的主要决定因素，也是网格QoS不同参数种类中最活跃的参数类型。

- 系统QoS

  系统QoS是用来描述对网格服务的服务能力有影响的系统环境方面的QoS参数。主要包括网络性能和本地资源节点环境两个方面的QoS。

- 安全QoS

  安全QoS是由服务提供者提供的用来表明网格服务自身安全级别和网格服务访问控制算法等方面的QoS参数，其中网格服务安全级别QoS用来满足网格用户对服务安全方面的QoS需要，网格服务访问控制算法QoS用来满足服务提供者安全管理方面的需要。通过服务访问控制算法QoS，服务提供者一方面可以确定合法的用户以及不同角色用户的

不同使用权限，从而实现区分访问的控制，另一方面可以实现与本地安全管理算法的无缝映射。

- 信任QoS

为了保证网格用户的QoS需要，要求虚拟组织中网格服务的信息是准确、可靠的，但由于网络性能和网格资源的动态变化以及可能存在的少数服务提供者故意虚报信息的情况，使得虚拟组织中由服务提供者提供的信息与实际信息有一定的偏差。偏差的大小将直接影响网格服务的健壮性和对QoS的保证能力，因此，需要对服务提供者提供的信息进行可信性评价。

- 记账QoS

为了构建大规模的网格系统，就必须存在一种激励机制，以促进资源提供者贡献自己的资源并从中获利[41]。记账QoS就是用来描述服务代价及其管理算法方面的QoS参数，它由服务提供者提供。当然，不同的服务提供者对服务代价的管理算法不尽相同，如对不同类型的用户收取不同的费用、对不同的QoS需求收取不同的费用、对不同时间段提供的服务收取不同的费用等。网格用户在选择服务时也会考虑到服务代价的问题，他们试图选择性能价格比最高的服务。所以，当构建大规模的网格环境时，甚至当网格真正成为公共设施时，记账QoS必将成为不可或缺的要素。

3.4.3 资源设备层的 QoS 描述

最底层为资源设备层。该层捕获各个物理资源的QoS属性以支撑虚拟组织层的各类QoS。由于网格环境中的物理资源不仅要响应网格任务的资源请求，还要响应本地任务的资源请求，并且本地任务的优先级在一般情况下要高于网格任务的优先级。因此，在共享资源可用的情况下，本地任务的到来将会导致资源对网格任务服务性能的下降甚至网格任务的中断。假设有一个物理资源整体服务性能很好，但该物理资源的本地任务到达率很高，使得该物理资源可被网格用户使用的时间很少，同时使得提交到该资源节点的网格任务的中断率很高。所以，从网格资源的角度来看，虽然该物理资源整体服务性能很好，但对于VO中映射到该物理资源上的逻辑资源而言，该逻辑资源的服务性能就比较差了。从这里我们可以看出，逻辑资源的服务性能是随着对应物理资源的本地任务负载情况的变化而动态变化的，但物理资源本身的整体服务性能是相对稳定的。因此，物理资源的QoS特性并不能直接代表VO中逻辑资源的QoS特性。逻辑资源QoS应该是综合考虑了物理资源本身的整体服务性能、资源本地任务的负载情况以及资源的共享算法等因素，是物理资源在网格环境中服务性能的抽象。

另外，由于逻辑资源所映射的物理资源要依附于一个物理资源节点，并且物理资源节点的支撑环境，如资源节点的操作系统环境、节点的响应速度和吞吐量、节点内存的大小等，它们虽然不能直接决定网格服务的整体QoS，但也会产生一定程度的影响。为了准确地对逻辑资源的QoS进行描述，在逻辑资源QoS参数中，我们只是描述了虚拟组织中直接提供网格

服务的逻辑资源的QoS，而没有对逻辑资源所在资源节点的环境QoS进行描述。因此，资源环境QoS在虚拟组织层应被抽象成系统QoS的一个组成部分。类似地，网络性能对网格服务的QoS也不能起直接决定作用，但可以产生一定的影响，网络QoS在虚拟组织层也应该被抽象成系统QoS的一个组成部分。所以，系统QoS应该是网络QoS和资源环境QoS两个方面QoS参数在虚拟组织层的抽象。

因此，虚拟组织层中各QoS参数在资源设备层上的具体映射为：系统QoS被映射成资源环境QoS和网络QoS，逻辑资源QoS被映射成物理网格资源QoS，安全QoS被映射到本地安全管理算法模块，信任QoS被映射到信任信息统计模块，记账QoS被映射到本地记账模块。

由于物理资源对网格用户是透明的，用户只能在应用层/网格服务层或者虚拟组织层提出QoS需求。在如图3-1的层次结构模型中，我们用虚线表示用户可以根据自己的需要选择在不同的层次上提出QoS需求。其中，在应用层/网格服务层，用户在功能性QoS方面可以提出简略的QoS请求；而在虚拟组织层，用户只能提出具体的QoS参数请求。

3.5 网格 QoS 的度量

由于受到网格中的资源和算法机制方面的限制，计算速度、存储量和系统吞吐率等多种 QoS 需求很难同时满足（有时候这些需求甚至是相互抵触的，例如，高速度的计算单元往往意味着更昂贵的价格，而用户往往希望在高速度的同时享有低廉的价格）。因而，评价一种资源管理算法抑或任务调度算法是否合理，针对单个性能目标是片面的，而应当综合性的针对多个性能目标。针对多个性能目标的资源管理的关键问题在于找到一种度量机制，该机制能够同时反映多个 QoS 目标，并体现其内在联系。

另外，用户通常只能够理解应用层的 QoS。因此，他们往往在应用层提出一些并不具体的 QoS 需求，如完成时间的要求、价格的要求、安全性要求等，这些要求最终都可以表现为用户对系统服务的满意度。对网格系统而言，需要将这些 QoS 需求转化为具体的虚拟组织层的 QoS 参数，并进而与底层的资源环境相联系。基于这种想法，本节提出一种度量机制，它将层次结构模型中最重要的 VO 层的各类 QoS 定量地描述出来，并与应用层用户满意度相联系，从而提供优化网格资源管理的量度和目标。

3.5.1 系统 QoS 的度量

系统QoS是用来描述对网格服务的服务质量有影响的系统环境方面的QoS参数。由于不同作业可能对具体的底层环境有不同要求，比如，对串行计算环境、并行计算环境的要求，或者对具体操作系统的要求；而服务提供者提供的资源环境可能从机器构成、操作系统构成等方面有一定区别，因此，对系统环境有特殊需求的作业需要被分配到特定的资源环境中。在这种情况下，系统QoS是硬性的要求，在进行资源选择的第一步，就需要根据用户要求进行系统环境的硬性筛选。如果资源不符合用户的系统QoS要求，用户对其满意度直接记为0；

如果符合，再根据其它的QoS参数计算用户满意度。

3.5.2 逻辑资源 QoS 的度量

逻辑资源QoS是直接反映服务性能的一个综合因素，它包含很多具体的参数，如计算速度、存储容量、传输带宽等。而不同网格服务的逻辑资源QoS对具体的QoS参数有所侧重，比如计算密集型服务可能不考虑存储容量这个参数，因为它对存储能力的要求比较低，很容易得到满足；反之，存储密集型服务可能不考虑计算速度。但是，无论一个网格服务侧重于什么样的QoS参数，这些服务性能方面的参数都综合形成了逻辑资源QoS。

在某网格应用中，第i个网格服务Si的逻辑资源QoS表示成n维的布尔向量 ϕ_i^L，n为所有逻辑资源QoS参数的种类。若 $\phi_i^L[k]=1$，则表示该网格服务用到了第k个分量所对应的QoS参数；反之，若 $\phi_i^L[k]=0$，则第k个分量所对应的QoS参数没有被用到。

如果网格服务不能满足用户的最低要求，用户满意度就为0；只有当用户所递交任务的逻辑资源QoS所包含的所有具体QoS参数的最低要求都被满足时，才能认为该网格系统有能力接受该任务。我们用特征值 l_i' 来表征逻辑资源QoS向量 Φ_i^L 包含的所有值为1的分量所对应的QoS参数的最低要求是否都被满足：若都被满足，则 $l_i'=1$；否则，$l_i'=0$。

u_{ij} 为第i个网格服务的 $\phi_i^L[j]$ 分量所对应QoS参数的用户满意函数，逻辑资源QoS的用户满意度计算公式如下：

$$l_i=(\sum_{j=1}^{n}\phi_i^L[j]\cdot u_{ij})/n \qquad 3\text{-}1$$

逻辑资源QoS的总的度量公式为 $l_i\cdot l_i'$。当用户提出的QoS需求有一个不能被满足时，特征值 $l_i'=0$，$l_i\cdot l_i'$ 也为0，至少有一个用户的QoS需求的最低要求没有被满足，所以用户满意度为0。

3.5.3 安全 QoS 的度量

我们可以仿照建立逻辑资源QoS度量公式的方法来建立安全QoS的度量公式：以n维布尔向量 $\phi_i^S[j]$ 表示网格应用中的第i个网格服务 S_i 的安全QoS所用到的QoS参数；以 s_i' 为特征值，s_i 为安全QoS的用户满意度；同样地，安全QoS的度量公式就为 $s_i\cdot s_i'$。

3.5.4 记账 QoS 的度量

记账QoS是用来描述服务代价及其管理算法方面的QoS参数。网格资源的提供者和消费者之间协商价格是一个很复杂的问题，Buyya等学者参照经济学的原理对网格中的价格协商机制进行了研究[27]。本文假设网格消费者已经以某种机制选定了某个网格资源的提供者，

认可了网格资源的提供者所提供的价格。与逻辑资源QoS和安全QoS所不同的是，记账QoS是一个一维向量，根据其用户满意函数就可以直接得到记账QoS的用户满意度，于是，可以用相似的方法对记账QoS定量描述。

以计算网格为例，时间因素（以秒计算）取决于作业本身长度（MI：兆指令数）和资源的处理速度(MIPS)，即有：

$$time = job\_length / processing\_speed \tag{3-2}$$

价格则取决于执行时间和单位时间的价格，即：

$$\begin{aligned} cost\_exec &= time * cost\_per\_sec \\ &= (job\_length / processing\_speed) * cost\_per\_sec \\ &= job\_length * (cost\_per\_sec / processing\_speed) \\ &= job\_length * cost\_per\_MI \end{aligned} \tag{3-3}$$

用户的期望价格为cost_expect，实际价格cost_exec由3-3决定，用户满意度c表示为：

$$c = (cost\_request - cost\_exec) / cost\_request \tag{3-4}$$

可见，c为在[0,1]内变化的实数。假设c'为特征值，记账QoS的度量公式就为$c_i \cdot c_i'$。

3.5.5 信任 QoS 的度量

在对信任 QoS 进行度量之前，有必要明确信任本身的含义。信任机制主要包括信任定义、信任模型和信任管理等内容。信任是一个非常复杂的主观概念，目前并没有一致的定义。我们的讨论基于 Azzedin 在[42]中提出的特性，对信任定义如下：

信任是对一个实体履行某种行为的信心，与该实体可靠性、诚信和性能相关；信任是一个主观概念，可以用信任值表示信任程度的高低；信任值是一个相对的度量，并且随实体的行为和时间而动态变化。

信任值的表示可以基于连续的数值或者基于信任等级。

在信任值度量之前，通常的做法是建立信任模型。信任模型也被称为信任度评估模型，其主要功能是对实体之间的信任关系进行评估，提供信任值的度量方法。信任评估依赖于结点的反馈信息、反馈信息的数目、反馈信息的可信度以及实体交互的上下文等等。信任模型主要分为两类：基于直接经验和基于推荐信任。前者的信任值取决于实体的自身经验，后者依赖于其它实体的间接信任。

信任管理是支持信任模型有效运作的管理系统，保证更加安全的交易。典型的信任管理系统主要有基于 PKI 的集中式管理、基于局部推荐的信任管理和全局可信度管理。

由于信任机制包含信任的方方面面，不同的信任模型对实体可信度有不同的侧重点，为了简化模型，在下面的讨论中，我们主要考察基于直接经验的信任机制，即用户对资源的信

任取决于直接交互，由调度系统在高层的信息中心为每个资源维护一个全局的资源信任向量。特别地，我们假设某个资源的信任值是[0,1]内的实数，0 代表完全不信任，1 代表完全信任，值越大表示资源可靠性的越强。每一次交易完成后，用户评估资源交易的结果，如果成功则增加资源的信任值，如果失败则采取相反的动作。信任值的增加将会增加后继用户对该资源的信心，信任值的减少则会降低后继用户对该资源的信心。当然，这种方式可能导致最初取得高信任值的资源能够接收到更多的作业，甚至导致最初信任值相对较低的资源一直得不到作业提交。但是，这一问题可以通过调整该资源其它方面的服务质量来吸引用户（比如降低价格等），本文将不对此作更多的讨论。

设资源的初始信任值为 T_0。系统根据历史交易的结果，记录下交易的正面经验数 p 和负面经验数 n，再根据当前的交易结果是否成功，分别用公式 3-5、3-6 更新信任值 T：

$$updateOnSuccess(T_{p+n+1}) = T_{p+n} + (1 - T_{p+n}) * \frac{p}{p+n} * IncreaseFactor \qquad 3\text{-}5$$

$$updateOnFailure(T_{p+n+1}) = T_{p+n} - T_{p+n} * \frac{n}{p+n} * DecreaseFactor \qquad 3\text{-}6$$

其中，递增因子IncreaseFactor和递减因子DecreaseFactor为[0,1]之间的实数，影响信任值的变化力度，并且随用户的偏好有不同的权值，乐观的人趋向于取较大的递增因子和较小的递减因子，悲观的人则相反。容易证明，更新后的信任值仍然在[0,1]之中，同时，可以观察信任值的增长规律。信任度较高的用户在积累了一定的信任值后，其增长速度较慢，但是惩罚的力度加大了，这是为了防止高信任值的实体恶意降低服务质量。

3.5.6 整体服务水平的度量

在对虚拟组织层的QoS进行分别度量后，需要对他们表现出的整体服务水平进行评价。这里通过用户满意函数计算出用户满意度，度量出网格系统提供给该任务的服务质量水平。

用户满意度计算公式如下：

$$U = w_l \cdot l + w_s \cdot s + w_c \cdot c + w_t \cdot t \qquad 3\text{-}7$$

其中，l, s, c, t分别为逻辑资源QoS值、安全QoS值、记账QoS值、信任QoS值，w_l, w_s, w_c, w_t 分别为四者的权重，U 为该服务总的用户满意度。

3.6 本章小结

对于希望提供非平凡服务质量的网格系统而言，网格 QoS 的研究具有重要的意义。在当前网格资源管理、任务调度研究中，用户的需求往往得不到全面描述和有效满足，如何描述、度量网格 QoS 是其中一个亟待解决的问题。

本章通过对传统网络 QoS 以及网格应用的 QoS 需求进行描述，阐明了网格 QoS 的问题范畴，提出了基于层次的网格 QoS 描述方法，并详细介绍了各种 QoS 的度量方法。

本章重点致力于网格 QoS 的描述和度量的研究，它是后续章节在网格任务调度中提供 QoS 支持的工作基础。

第四章 提供QoS支持的网格任务调度算法

网格任务调度是网格资源管理的核心部件。网格任务调度的实质就是为任务寻找合适的资源，使得在满足用户需求的前提下，任务完成时间尽可能小而资源利用率尽可能高。其中，用户的需求必须通过网格QoS参数来描述，这就涉及到网格QoS描述与度量的问题。针对这一问题，我们在第三章提出了网格QoS描述与度量机制，并作了详细阐述。本章基于这一工作，重点描述提供服务质量支持的网格任务调度算法。首先，从总体上描述网格调度问题并给出相应的形式化描述方法（第一节）；而后在传统的在线调度（第二节）和批调度（第三节）算法中加入服务质量支持，并分别通过GridQoS网格模拟器进行模拟和验证。

4.1 网格任务调度问题描述

本节讨论了网格调度问题的应用类型、一般性描述以及调度过程中任务执行时间预测等相关问题。

4.1.1 术语

采用的相关术语描述如下：

任务t_i在资源m_j上的期望执行时间e_{ij}定义为m_j在无负载的情况下完成t_i所消耗的总时间。其中，e_{ij}包含t_i代码转移至m_j的时间。任务t_i在资源m_j上的期望完成时间C_{ij}定义为m_j完成t_i（在完成所有先期任务之后）的实际时间。定义m为网格环境下的总资源数，K为该环境下进行一次调度算法评测研究的所采用的一组任务的数量。令任务t_i的到达时间为a_i，定义b_i为t_i开始执行的时间。根据以上定义有：

$$C_{ij} = b_j + e_{ij} \qquad 4\text{-}1$$

4.1.2 调度问题的一般性描述

调度问题的一般化描述通常可以借助于Graham分类方法[43]。这是一个三元符号域：$\lambda|\beta|\gamma$：其中，λ描述资源环境；β描述应用特征和约束条件；而γ域表示用于优化调度的目标函数。具体介绍如下：

λ：由于网格环境的特征在于异构、分布，因此，网格资源环境可以表示为不同硬件体系结构、不同操作系统的独立资源的集合。其具体的特征描述可以从以下几个方面考虑：

- 硬件参数，包括体系结构、CPU处理能力、存储能力等；
- 软件支持特征，包括操作系统、支持的编程语言类型等；

- 网络环境，由于资源分布在不同的地理位置，其网络带宽参数是描述其环境的重要特征；
- 共享特征，网格资源可能是独占的（即每次只接受一个用户），也可能是共享式的（若干用户根据一定的算法共同使用资源）；

β：从任务的应用特征角度，任务的相关信息包括：

- 任务的应用类型。

  网格环境中任务调度所要面对的应用类型主要包括以下三种：

  - ✓ 单一任务应用：是指该应用只包含唯一一个子任务 j_1，该子任务是串行任务或者通信密集型的并行任务，该类任务从性能角度考虑只能在单一的目标资源上执行。
  - ✓ 密集并行应用：是指该应用可以按照参数空间划分为多个独立、互不相干的子任务$(j_1, j_2, \cdots, j_n)$，一般来说各个子任务所处理参数空间的大小（也就是所含负载量的大小）都是相同的。
  - ✓ 工作流型应用：是指该应用由多个子任务$(j_1, j_2, \cdots, j_n)$组成，且各子任务之间具有严格的前后序约束，构成任务图。

  从一般的网格调度研究范畴考虑，本文的工作不讨论有相关性的任务序列，即不讨论工作流应用的情况。

- 任务到达率。

  该参数反映了用户向网格系统提交任务请求的频率，它直接影响到调度系统的调度方式。比如，在任务到达率较低的情况下，一般采用在线调度方式，而在任务到达率较高的情况下，批调度方式由于能够在每一次映射时间点综合考虑所有资源、任务信息，在系统吞吐率方面有更好的表现[44]。

- 任务应用需求，也即 QoS 需求。

  任务的应用需求是对用户 QoS 需求的刻画，它同时也是对调度系统的约束条件。根据第三章的工作，网格 QoS 可从应用层、虚拟组织层、资源设备层三个层次来描述。但从网格用户的角度，他们只能够从应用层提出一些较高层的应用需求，这些应用需求是用户能够理解的，它们不一定能够直接反映对底层系统具体参数的要求。比如，用户对完成时间的限制，对任务总开销的限制，对系统安全性的要求等等。这些要求可以通过一定的算法映射为虚拟组织层的 QoS 参数，并进而映射到资源设备层。

γ：目标函数。调度系统通过不同调度算法来达到不同的优化目标。从调度系统出发，面向系统的优化指标一般采用系统吞吐率，系统吞吐率指标可采用 makespan[45]衡量（makespan 是异构计算系统中常用的吞吐率衡量指标，定义为一组任务中第一个任务开始执行时间到最后一个任务结束时间之间的差值），这里形式化定义为：

$$\max t_i \in K(C_{ij}) \qquad 4\text{-}2$$

另外，从任务的应用需求出发，作为优化指标的参数包括：

- ✓ Completion Time：即任务完成时刻，这是用户通常非常关心的参数；
- ✓ Total Cost：$CostPerSecond * e_{ij}$，即总的价格开销；
- ✓ 以及根据任务其它具体应用需求的QoS评测参数；

4.1.3 任务执行时间预测问题

无论是批调度或实时调度，都假设能够预测每个任务在每个计算单元上的执行时间。任务执行时间预测问题是调度系统实现中的一个重要部分。C_{ij}表示任务t_i在资源m_j上的期望完成时间。C_{ij}可通过计算或通过假设已知的数据库获得。我们可以通过两种方式计算C_{ij}的值。

➢ 短期预测模型

在首轮调度中，调度器对所有任务的执行时间进行初始猜测。一旦任务完成，调度器将根据记录的实际执行时间通过修改预测算法改进预测的精确度。在AppLeS系统[46]中采用了该方式，它的研究表明，其平均的预测误差为11个百分点。

➢ 长期预测模型

在文献[47]中给出了一种长期的性能预测模型用以在网格环境中预测任务完成时间。近年来，工作[48]基于长期性能模型提出了一种名叫网格结果服务（Grid Harvest Service，GHS）的性能评价和预测系统。实验结果表明，GHS的预测误差能够控制到10个百分点以内。

为了问题简化，本文采用预测矩阵，记录不同任务在不同机器上的预测完成时间。矩阵行对应不同任务，列对应异构机器，值对应e_{ij}。表1给出了一个预测矩阵例子。

由表1得到e_{ij}，每个机器对应一个空闲剩余时间r_j，根据式（1）得到任务预测完成时间C_{ij}。每完成一个任务，r_j、C_{ij}进行相应更新。

表4-1　任务完成时间预测矩阵示例

	m0	m1	m2
t0	10	6.7	5
t1	20	13.3	10
t2	30	20	15

4.2 提供服务质量支持的在线调度算法

通过对传统的调度算法的分析，本节针对在线调度提出了提供服务质量支持的调度算法，并基于GridQoS网格调度模拟器进行了模拟验证。

4.2.1 传统的在线调度算法

对于在线调度模式，每个任务在做资源映射和任务调度时只被考虑一次，一旦任务开始执行，映射关系将不再改变。当任务到达率（即任务抵达服务系统的速率）很低时，系统可以在任务到达时立即处理。如果这时候采用周期性的调度方式，使得每个已到达的任务需等到下一次调度时间到来才予执行，就会耽误任务的执行时间，因此，在这种情况下调度器采用在线调度模式是合理的。

一些经典的在线调度启发式算法包括最小完成时间算法、最小执行时间算法等。

- 最小完成时间（MCT）启发式算法

  其做法是将任务分配到能够最早完成的资源上。由于考虑到资源本身的负载等问题，该算法可能导致某些任务并非在具有最小执行时间的资源上执行。MCT 算法往往用作在线调度模式的评测基准，即其它在线调度启发式算法通过与 MCT 算法进行性能比较来衡量算法的优劣。在 MCT 算法下，每当有任务到达时，所有的资源将被检测一遍以决定具有该任务最小完成时间的资源，因此，算法时间复杂度为 O（m）。

- 最小执行时间（MET）启发式算法

  其做法是将任务分配到具有该任务最小执行时间的资源上。与 MCT 相比，该算法未考虑资源本身的负载情况，因此可能导致资源负载的严重不平衡。而其优点则是，它使得每个任务都能够在具有其最小执行时间的资源上执行。同样的，该算法的时间复杂度也为 O（m）。

近年来，随着计算网格市场模型的提出[27]，出现了一些新的调度算法。这类算法以市场为基础，从用户的角度出发，同时考虑完成时间、计算成本这两种 QoS 因素。但是，由于 QoS 考虑比较单一，其调度模型的应用基础受限于计算网格市场环境，因此，需要提出更全面和实用的提供服务质量保证的在线调度算法。

4.2.2 提供服务质量支持的在线调度算法

在需要提供非平凡的服务质量的网格环境下，传统的在线调度算法由于局限于单一的 QoS 参数（完成时间、执行成本），因此，有必要研究更全面和实用的提供服务质量支持的在线调度算法。

用户提交任务的同时提交 QoS 请求：$\left((a_1,a_2,\cdots,a_n),(l_1,l_2,\cdots,l_n),(s_1,s_2,\cdots,s_n),c\right)$，其中 a 代表系统 QoS，a_i 代表系统 QoS 的第 i 个分量；l 代表逻辑资源 QoS，l_i 代表逻辑资源 QoS 的第 i 个分量；s 代表安全 QoS，s_i 代表安全 QoS 的第 i 个分量；另外，c 代表记账 QoS。

提供服务质量支持的在线调度算法任务提交的数据结构如表 4-2 所示，requestQoS 表示用户期望的服务质量。

表 4-2　任务请求的数据结构

任务请求数据结构
jobID（任务号）
userID（任务隶属的用户）
Length （任务长度）
requestQoS（服务质量要求）

提供服务质量支持的在线调度算法伪码描述如图 4-1 所示：

```
1)  do 直到所有任务处理完成
2)    for 任务组 Tv 中每一个任务 t_i
3)      获取任务 QoS 请求为 ((a_1,a_2,...,a_n),(l_1,l_2,...,l_n),(s_1,s_2,...,s_n),c);
4)      从资源信息中心获取可用资源的列表 Rv;
5)      根据系统 QoS (a_1,a_2,...,a_n) 的硬性需求，过滤一部分资源;
6)      如果剩余资源数为 0，报告 t_i 处理失败 1)，返回 2)；否则，继续;
7)      for 剩余的所有资源
8)        根据第三章的方法计算逻辑资源 QoS、安全 QoS、记账 QoS 度量值 l,s,c;
9)        根据公式 3-7 计算用户满意度 U;
10)     end for
11)     寻找最优的映射匹配 match(i,l) = f(r_1,r_2,...)
12)     将任务 t_i 发送至资源 r_l ;
13)     更新资源 r_l 的负载信息;
14)     记录任务 t_i 相关执行信息;
15)   end for
16) end do
```

图 4-1　提供 QoS 支持的在线调度算法伪代码

如图 4-1，首先根据任务 t_i 系统 QoS 硬性过滤资源，然后对剩余的资源，计算 t_i 对每一个资源的用户满意度，之后，按照选择方法“f”，选择最优的资源 r_l。最后，将任务 t_i 发送至资源 r_l，并对资源信息进行相关的更新。其中，资源选择方法“f”可以是将任务优先提交到具有最大用户满意度的资源上。假设任务规模为 S，资源数目为 m，则整个算法的时间复杂度为 O(Sm)。需要说明的是，这里没有引入对信任 QoS 的考虑。由于信任 QoS 需要引入资源失效、任务重传的机制，为了问题的简化，我们对此暂不作考虑。

4.2.3 模拟实验与结果分析

本文基于GridQoS（将在第五章作详细介绍）网格模拟器模拟实现了提供服务质量感知的在线调度算法。在此过程中，将该调度算法与MCT算法进行比较，从系统吞吐率、用户满意度等几项评测指标详细分析了提供服务质量感知的在线调度算法的性能表现。

4.2.3.1 模拟环境与参数设置

资源场景设置：

1. 资源数量为1000。
2. 资源的QoS描述信息包括：单CPU处理能力，以MIPS（每秒百兆指令周期数）描述；安全级别，通过三个安全级别描述；单位处理时间价格。
3. 模拟的资源类型分别有“Compaq AlphaServer”，“Sun Ultra”，“Intel Pentium”，“SGI Origin”，“HP Integrity”，“HP Superdome”等机型。

任务参数设置：

1. 任务到达时间，由于在线调度是任务在到达时立即执行任务调度，因此，我们假设模拟中处理的任务其到达时间为0.0s，调度时刻也为0.0s；
2. 任务数量取在10~100之间变化。
3. 任务的QoS要求描述为：资源类型要求、任务完成时间（jobRequestTime）、任务安全级别（jobRequestSafety）、任务执行价格（jobRequestCost）。其中，由于任务完成时间能够进一步映射成为对CPU处理能力以及资源负载情况的要求，因此，将其作为逻辑QoS处理。
4. 用户偏好：u, v, w分别描述jobRequestTime, jobRequestSafety, jobRequestCost三项参数的权值，它们反映了这几项参数对用户的重要程度。

评测参数：

1. 进行比较模拟的算法内容包括：Strict Time，单纯考虑完成时间的在线调度；Strict Time，即单纯考虑任务价格的在线调度；Strict Safety，单纯考虑安全性的在线调度；Comprehensive QoS，代表本文提出的提供QoS支持的在线调度算法。
2. 评测参数：作业平均用户满意度（UserSatisfaction），系统吞吐率（makespan）。

4.2.3.2 模拟内容

（一）各种在线调度算法的用户满意度比较

假设所有资源数量一定（1000）的情况下，模拟Strict Time, Strict Cost, Strict Safety, CompreQoS四种在线调度算法。前三种算法分别以单一QoS为优化目标，类似于MCT在线调度方法，将任务调度到具有最优QoS的资源，其中Strict Time算法以作业完成时间为优化目

标，即为MCT算法。CompreQoS则是本文提出的提供QoS支持的在线调度算法。它们的用户满意度比较如表4-3和图4-2所示。

由表4-3和图4-2可见，CompreQoS的用户满意度明显优于其它三种算法：相对于其它三种算法中用户满意度最好的Strict Time算法，性能平均提升在20%左右。随着用户数量的增加，CompreQoS表现出的用户满意度一直比较稳定，而其它三种算法则性能下降明显。在这项模拟内容中，任务逐渐增多，而与此同时资源数量保持不变。对于CompreQoS算法，由于均衡的考虑了用户的所有要求，任务数量的增多并没有给用户满意度造成大的影响。在这个过程中，任务数量的增加提高了任务对资源争抢的可能，因此，任务执行时间会受到影响。但由于QoS权重的引入，用户的满意度需要综合考虑多项性能指标，因此，前三种算法由于仅单纯考虑一项性能参数，用户满意度下降明显。

表 4-3　在线调度算法用户满意度比较

	10	20	30	40	50	60	70	80	90	100
Strict Time	0.74	0.60	0.51	0.47	0.45	0.43	0.42	0.41	0.40	0.38
Stirct Safety	0.72	0.65	0.61	0.55	0.50	0.48	0.45	0.43	0.40	0.38
Strict Cost	0.71	0.51	0.45	0.42	0.40	0.39	0.38	0.37	0.37	0.37
CompreQoS	0.74	0.70	0.69	0.69	0.68	0.68	0.66	0.64	0.64	0.62

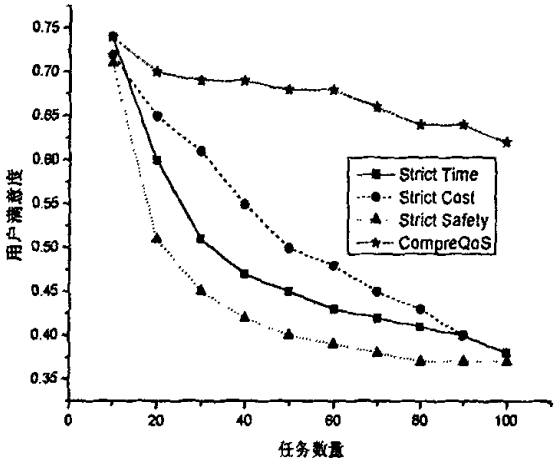

图 4-2　在线调度算法用户满意度比较

（二）在线调度算法的系统吞吐率比较

在资源数量一定的情况下，模拟Strict Time, Strict Cost, Strict Safety, CompreQoS四种在线调度算法。算法系统吞吐率Makespan如表4-4和图4-3所示。四种调度算法中，Strict Time表现出最好性能，这一点是因为该算法以任务完成时间为优化目标，因此能够相对地使任务更快完成。相比Strict Time算法，CompreQoS性能表现相差平均为8%，可见，虽然为了全面综合各项性能参数使得系统吞吐率有一定损失，但损失很小，相比用户在其它性能上获得的收益，我们认为这种损失是值得的。

表 4-4　在线调度算法 makespan 比较

	10	20	30	40	50	60	70	80	90	100
Strict Time	9.7	11.7	11.7	11.7	12.5	16.5	22.5	25.5	30.5	32.5
Stirct Safety	28.1	45.5	46.0	56.8	67.4	84.9	102.3	108.6	120.9	138.4
Strict Cost	26.3	40.2	40.2	57.2	68.3	86.1	101.6	110.1	120.0	138.1
CompreQoS	10.8	11.9	11.9	11.9	15.0	18.6	22.6	29.3	35.3	37.5

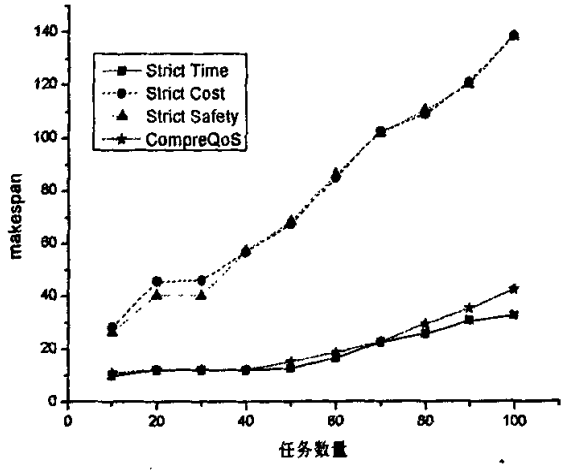

图 4-3　在线调度算法 makespan 比较

4.3 提供 QoS 支持的批调度算法

通过对传统的批调度算法的分析，本节针对批调度提出了提供服务质量支持的网格批调度算法，并基于GridQoS网格调度模拟器进行了模拟验证。

4.3.1 传统的批调度算法

在批调度模式中，任务是作为调度队列进行统一映射和调度的。每一次通过一定的启发式算法对任务队列执行资源映射称之为一次映射事件。调度器在每一次映射事件发生时都将对批队列中的任务加以映射和调度。由于批调度的启发式算法全面考虑了整个任务队列的信息，这种调度模式可能比在线调度模式能够做出更优的调度决策。当任务的到达率很高时，在每两个映射事件之间以及每次映射执行计算的时刻，将有足够多的任务到达使得机器保持繁忙（这里，我们假设每次进行映射执行的计算时间相比平均任务执行时间小到可忽略）。需要指出的是，当任务到达率很低，以至于在每一次映射事件发生时，任务队列中仅有或甚至没有一个任务，批调度模式将退化为在线调度模式。这时候，考虑到映射事件带来的系统实现上的复杂性，采用在线调度将更具合理性。

批调度的映射事件有两种触发方式：

- 固定时间间隔算法触发（比如，每10秒钟对任务队列执行一次批调度）。在这种情形下，映射事件可能出现冗余：一种可能性是在此期间没有新到达的任务；另一种可能是在上一次映射事件之后尚没有任务完成，因此，机器的负载信息并没有发生变化。这些情况可以通过引入状态检测机制来避免。
- 固定任务数量算法。当下列两种相反的情形分别发生时，该算法将触发一次对任务队列 R_j 的映射事件：一种是当到达的任务使得 R_j 中的任务数大于或等于预先指定的数目 κ；另一种则是 κ 数目的任务已经到达，而当某一任务完成时未完成的任务数目已大于或等于 κ。

在本文的研究中，算法的测试是基于单次映射事件，其中，任务队列大小设置为S。

这里介绍三种具有代表性的批调度启发式算法：Min-Min算法，Max-Min算法，Sufferage算法[44]，具体介绍如下：

- Min-Min算法首先对每个任务找出具有它最早完成时间的资源 r_l，再找出具有最早完成时间的任务 t_k，然后将 t_k 分配到 r_l 上。
- Max-Min启发式算法与Min-Min相似，其不同点在于：每个任务分别找到最早完成时间的机器后，将具有最大值的最短完成时间的任务先分配。也就是说，Min-Min算法，先完成短任务，后执行大任务，而Max-Min算法则相反。
- Sufferage算法的思想是，如果不分配机器 r_l 给任务 t_k，将对 t_k 的期望完成时间产生最大影响，则该任务应该先分配资源。将一个任务的sufferage值定义为该任务的最早完成时间和次早完成时间的差值。每次分配资源的时候，该值最大的任务最先分配。

文献[28]提出了一个批调度算法的普遍性描述方法，这里用伪代码描述，如图4-4所示。

```
1) 由公式 C_ij = r_j + e_ij 生成初始任务执行时间预测矩阵;
2) do 直到所有任务完成映射
3)     for 任务队列 T_v : (t_1, t_2, ..., t_n) 中的每一个任务;
4)         计算 metric_i = f_1(c_1, c_2, ...);
5)     end for
6)     选择最佳的映射 match(k,l) = f_2(metric_1, metric_2, ...);
7)     将任务 t_k 发送至资源 r_l;
8)     从任务队列 T_v 中删除 t_k;
9)     更新任务执行时间预测矩阵;
10) end do
```

图4-4　批调度算法的一般化伪代码表示

对照图4-4，Min-Min、Max-Min、Sufferage等批调度算法可以通过定义不同的“f1”以及最优匹配选择方法“f2”生成。比如，Min-Min、Max-Min算法中，“f1”定义为最小完成时间，也就是对每个任务找出具有最早完成时间的机器。对“f2”，Min-Min算法选择具有

最小完成时间的任务，而 Max-Min 则选择具有最大完成时间的任务。也就是说，Min-Min 算法，先完成短任务，后执行大任务，而 Max-Min 算法则相反。

假设任务数为 S，计算资源数为 m。，图 4-4 所示的伪代码时间复杂度为 $O(S^2m)$。

传统的批调度算法都是以优化系统吞吐率makespan为目标，并不考虑单任务的用户需求。因此，对于用户有具体QoS需求的情况，批调度系统往往不能兼顾。提供非平凡的服务质量是网格系统的重要标志，而批调度是网格调度算法中的重要手段，因此，研究如何在批调度中引入对具体QoS参数的考虑具有重要的意义。

4.3.2 服务质量感知的批调度算法

任务与任务之间，由于对具体服务质量有不同要求，有一些任务可能对服务质量要求较高，而一些任务对服务质量则要求较低。在网格环境中，资源往往是有限的，传统的批调度算法是将任务以特定的启发式算法成批调度，其优化目标在于提高系统的吞吐率，这些调度算法都假设任务具有相同的 QoS 需求，并没有优先级之分。但在实际的应用场景中，由于用户的个人需求不同，任务往往存在不同的 QoS 需求。单纯以优化系统吞吐率为目标可能导致一些对 QoS 有更高要求的任务由于那些具有低 QoS 要求的任务占用资源而不能得到及时、有效的处理，因此，有必要在进行批调度时对任务进行区分。对此，本文采用一个优先级分类机制解决这个问题。

任务请求的数据结构如表 4-5 所示：

表 4-5　任务请求的数据结构

任务请求数据结构
jobID（任务号）
userID（任务隶属的用户）
Length（任务长度）
time_arrive（任务到达时间）
requestQoS（服务质量要求）

用户提交任务同时提交 QoS 请求：$\left((a_1,a_2,\cdots,a_n),(l_1,l_2,\cdots,l_n),(s_1,s_2,\cdots,s_n),c\right)$，其中 a 代表系统 QoS，a_i 代表系统 QoS 的第 i 个分量；l 代表逻辑资源 QoS，l_i 代表逻辑资源 QoS 的第 i 个分量；s 代表安全 QoS，s_i 代表安全 QoS 的第 i 个分量；另外，c 代表记账 QoS。

为了获取网格资源的平均 QoS 水平，我们令资源信息中心维护一组平均资源 QoS 向量：$\left((l_{r1},l_{r2},\cdots,l_m),(s_{r1},s_{r2},\cdots,s_m),c_r\right)$，其中，$l_r$ 代表平均逻辑资源 QoS，l_{ri} 表示平均逻辑资源 QoS 的第 i 个分量；s_r 代表平均安全 QoS，s_{ri} 表示平均安全 QoS 的第 i 个分量；c_r 代表平均记账 QoS。资源信息中心负责定期获取现有的资源状态，对平均 QoS 进行定时更新。

对任务 t_i，对应资源平均 QoS 水平，按照公式 3-7 可计算该任务对资源的平均 QoS 的用户满意度（与 4.2 节的考虑相似，为了问题的简单化，这里暂不考虑信任 QoS），即

$$U_i = w_l \cdot l + w_s \cdot s + w_c \cdot c \quad 4\text{-}3$$

另外，由于：

$$w_l + w_s + w_c = 1 \quad 4\text{-}4$$

$$l \in [0,1], s \in [0,1], c \in [0,1] \quad 4\text{-}5$$

因此，$U_i \in [0,1]$。

随着U_i的增大，用户对资源平均QoS的满意度越高，这表明，该用户对资源QoS的要求越宽松；相反地，随着U_i的减小，用户对资源平均QoS的满意度越低，这表明，该用户对资源QoS的要求越苛刻。可见，U_i在[0, 1]之间的分布表明了用户对资源QoS要求的高低。我们将U_i的分布区间[0, 1]划分为n个子区间，如图4-5。

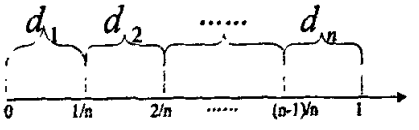

图4-5　区间[0,1]划分示例

U_i值越小的任务对资源QoS要求越高，如图4-5，对U_i处于区域d_1的任务，它们的QoS要求程度要高于分布在其它区域的任务；而对U_i处于区域d_n的任务，QoS要求相对最低。由于对资源QoS要求越高的任务其资源选择的条件越苛刻，为了避免具有高QoS属性的资源被低QoS要求的任务占用，在映射中具有更高QoS要求的任务应该具有更高的优先级。根据U_i值所属的这n种情况，任务由此被分为n类优先级，调度则按照优先级高低依次调度。提供QoS支持的批调度算法伪代码形式如图4-6。

1）由公式$C_{ij} = r_j + e_{ij}$生成初始任务执行时间预测矩阵；
2）**do** 直到所有任务完成映射
3）　**for** 任务队列T_v中的每一个任务t_i
4）　　根据资源信息中心获取资源平均QoS；
5）　　计算任务对应资源平均QoS的用户满意度U_i；
6）　　根据U_i值为t_i确定其所属的区间d_x；
7）　**for** 区间a_1中每一个任务 t_j
8）　　根据资源信息中心获取可用资源列表；
9）　　根据t_j的系统QoS需求$(a_1, a_2, \cdots, a_n)$，硬性过滤资源；
10）　　如果剩余资源列表长度为0，报告任务t_j处理失败，返回7）；否则，继续；
11）　　对剩余资源计算 $metric_i = f_1(c_1, c_2, \cdots)$；
12）　**end for**
13）　选择最佳的映射 $match(k,l) \approx f_2(metric_1, metric_2, \cdots)$；

14）　将任务t_k发送至资源r_l；

15）　从任务队列T_v中删除t_k；

16）　更新任务执行时间预测矩阵，更新资源平均QoS；

17）　对区间$d_2 \sim d_n$，重复步骤7)～16)；

18）**end do**

图4-6　提供QoS支持的批调度算法伪代码

首先计算每个任务的U_i值，根据U_i值定义方法，U_i值越小任务优先级越高。根据U_i值的不同，批队列$T_v:(t_1,t_2,\cdots,t_n)$中的任务被划分为n类，n类任务按照优先级高低分别使用传统批调度算法调度。其中，在执行“f1”选择方法前，任务所考虑的资源应为根据其系统QoS要求进行了硬性过滤后所剩余的资源。

假设任务数为S，计算资源数为m。步骤1）的时间复杂度为O(Sm)，步骤3）至6）的时间复杂度为O(S)，步骤（7）至步骤（17）的时间复杂度为$O(S^2m)$。因此，算法总的时间复杂度比传统批调度算法没有增加，仍然为$O(S^2m)$。

4.3.3 模拟实验与结果分析

我们基于GridQoS网格调度模拟器模拟实现了提供服务质量感知的批调度算法，同时，将此调度算法与Min-Min算法进行比较，从系统吞吐率、用户满意度两项评测指标详细分析了提供服务质量感知的批调度算法的性能表现。

4.3.3.1 模拟环境与参数设置

与4.2.3.1相似地，环境描述如下：

资源场景设置：

1. 资源数量为100。
2. 资源的QoS描述信息包括：单CPU处理能力，以MIPS（每秒百兆指令周期数）描述；安全级别，通过三个安全级别描述；单位处理时间价格。
3. 模拟的资源类型分别有“Compaq AlphaServer”，“Sun Ultra”，“Intel Pentium”，“SGI Origin”，“HP Integrity”，“HP Superdome”等机型。

任务参数设置：

1. 任务到达时间：jobArriveTime，该时间描述了任务在队列中的停留时间，它是批调度中重要的考虑因素。模拟实验中，假设任务到达时间在[0s，10s]之间随机分布，它也表示批调度映射事件是以10s为时间周期。
2. 任务数量取在100~1000之间变化。
3. 任务的QoS要求描述为：资源类型要求、任务完成时间（jobRequestTime）、任务安全级

别（jobRequestSafety）、任务执行价格（jobRequestCost）。

4. 用户偏好：u, v, w分别描述jobRequestTime, jobRequestSafety, jobRequestCost三项参数的权值，它们反映了这几项参数对用户的重要程度。

评测参数：

1. 进行比较模拟的算法内容为：Min-Min批调度算法，QoS Min-Min批调度算法。
2. 评测参数：作业平均用户满意度（UserSatisfaction），系统吞吐率（makespan）。

4.3.3.2 模拟内容

（一）批调度算法用户满意度比较

假设所有资源都是可靠的情况下，模拟Min-Min批调度算法，QoS Min-Min批调度算法。其中，QoS Min-Min算法即是本文所提出的提供QoS支持的批调度算法，对应图4-6，其“f1”“f2”匹配算法采用Min-Min中的匹配算法。特别地，QoS Min-Min中的优先级划分为5级，分别描述用户紧急、偏紧急、一般、偏宽松、宽松五种需求状态。

它们的用户满意度如表4-6和图4-7所示。

如图4-7，QoS Min-Min算法在用户满意度的表现上明显优于Min-Min算法。随着网格任务的增加，资源数量保持不变，QoS Min-Min的用户满意度变化并不明显，而Min-Min则急剧下降。

表 4-6　批调度算法用户满意度比较

	100	200	300	400	500	600	700	800	900	1000
Min-Min	0.69	0.65	0.60	0.58	0.56	0.55	0.50	0.45	0.43	0.35
QoS Min-Min	0.73	0.67	0.68	0.67	0.67	0.67	0.67	0.66	0.65	0.60
Improvement	5.8%	3.1%	13.0%	15.5%	19.6%	21.8%	24.0%	46.7%	51.2%	71.4%

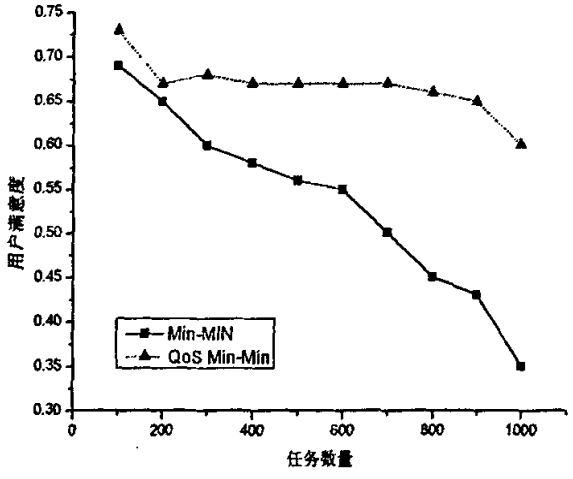

图 4-7　批调度算法用户满意度比较

（二）批调度算法系统吞吐率比较

引入对资源可靠性的考虑，资源存在作业提交失效重传的情况。在这种情况下，两种批调度算法性能表现如表4-7和图4-8所示。

表 4-7　批调度算法 makespan 比较

	100	200	300	400	500	600	700	800	900	1000
Min-Min	10.0	40.5	15.8	21.2	28.8	35.1	50.3	65.9	77.2	91.3
QoS Min-Min	10.0	40.3	15.9	20.5	28.1	35.4	50.2	66.8	78.0	91.7
Decrease	0.0%	0.5%	0.6%	-3.3%	-2.4%	0.8%	-0.2%	1.4%	1.0%	0.4%

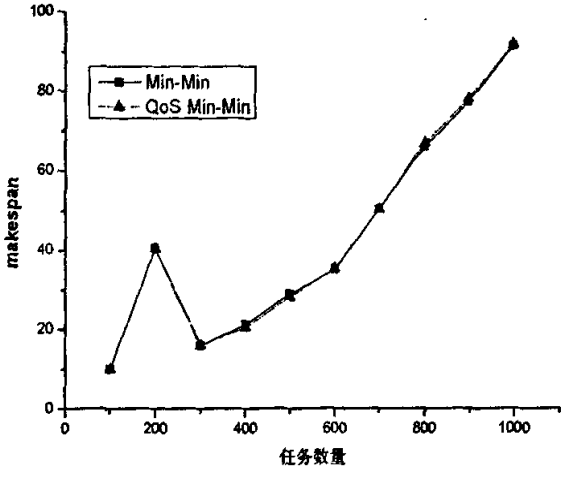

图 4-8　批调度算法 makespan 比较

可见，QoS Min-Min相比Min-Min在系统吞吐率上并没有明显损失，因此，可以认为，QoS Min-Min仍保持了批调度算法对系统性能的优化特点。

4.4 本章小结

网格任务调度在网格系统中起着举足轻重的作用。在网格调度中提供QoS支持是近年来网格研究的热点与难点。本章基于第三章网格QoS的度量的工作，分别针对网格在线调度与批调度提出了提供QoS支持的任务调度算法。模拟实验的结果证明，本文提出的调度算法相比传统调度算法，能够更全面的适应用户需求和系统需求。

下一章将详细介绍本文提出的网格模拟器GridQoS的设计和实现。

第五章 GridQoS 网格模拟器设计与实现

模拟、仿真、实测是验证研究工作的几种手段。在网格研究领域，针对网格任务调度方面研究的验证一直是研究工作中的一大难点。由于网格环境本身的大规模、异构、分布等特点，几乎不可能可重复地、可控制地测试某个调度算法的性能，因此，我们考虑采用模拟的手段对本文的研究工作予以验证。

本文调度算法方面的验证工作是基于 GridQoS 网格模拟器得出的，本章则详细介绍了 GridQoS 网格模拟器的设计和实现工作。首先，介绍设计 GridQoS 的起因（第一节）；然后，从体系结构（第二节）、具体实现（第三节）对模拟器进行描述；最后，以本文研究工作的核心算法实现为例，介绍基于该模拟器的应用方法（第四节）。

5.1 设计 GridQoS 的起因

在网格系统中，资源管理和调度的实现是一项复杂的工程。为了检验相关调度算法的效率，需要在各种情况下测试各种性能，比如资源的数量、用户的需求。网格环境中，由于资源和用户是分布在许多组织中，并且有自己的规则，所以几乎不可能可重复地、可控制地测试某个调度算法的性能，于是结果也是无法比较和估计的。这个时候，出现了一些网格系统模拟器，用以模拟实现和评测各种调度算法。由于网格系统模拟的具体复杂性，到目前为止，并没有足够强大的网格模拟器出现(比如，没有功能上相当于网络模拟器 NS 的模拟器)。

在现有的网格模拟器中，GridSim[49]是具有较大影响力的一个项目，它是基于 Java 离散事件的网格模拟工具包，支持异构网格资源、用户和应用模型的建模和模拟。GridSim 提供建立任务和把任务映射到资源及其管理的接口原语。但是，GridSim 的首要目标是通过模拟来研究基于网格计算经济模型的有效资源分配方法，它在实际的调度算法模拟中只支持简单的在线调度算法，同时，它并不提供对用户的全面的 QoS 描述。有鉴于此，为了模拟验证本文提出的提供服务质量支持的调度算法，我们基于 GridSim 模拟器进行了扩展，设计与实现了能够满足本文工作应用需求的网格模拟器 GridQoS。

GridQoS 设计目标包括：

- ✓ 提供对用户、作业、资源等实体的模拟；
- ✓ 支持对作业实体 QoS 描述；
- ✓ 支持对资源异构性的模拟；
- ✓ 能够模拟各实体间的通信和交互；
- ✓ 能够模拟网格在线调度算法；
- ✓ 能够模拟网格批调度算法；
- ✓ 能够在调度算法中引入对 QoS 的支持；
- ✓ 支持面向系统（吞吐率等）以及面向用户（完成时间等）的各种评测指标。

5.2 GridQoS 的体系结构

如图 5-1 所示，GridQoS 按层次化和模块化的方式设计，各层次介绍如下：

第一层，Java 接口和 Java 虚拟机（JVM），JVM 支持单个或多处理器，包括集群。

第二层，利用第一层接口，建立基本的离散事件基础结构，采用比较流行的SimJava。

第三层，主要利用第二层的离散事件服务模拟系统实体。其中，核心网格实体包括资源、信息服务等；另外，提供应用程序模型，统一访问接口，应用模拟原语和建立更高层次实体的框架。

第四层，集中于对网格调度器的模拟，这一层次为模拟各种调度算法提供支持。

第五层，作为最上层，主要用于不同情境下的资源和应用建模。它利用第三层和第四层提供的服务评估调度算法。这一层次提供对用户服务质量需求的描述能力，同时提供对调度算法的评测能力。并且，这一层次为网格模拟人员提供系统交互，使得模拟人员能够指定输入、输出内容。

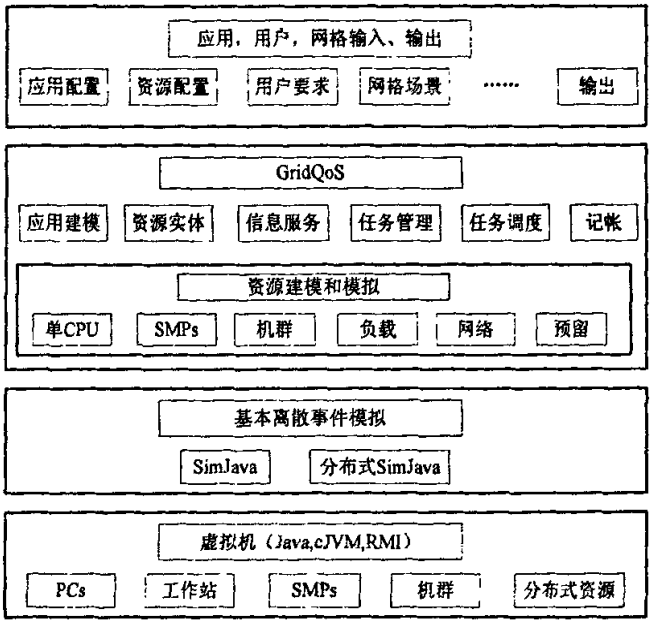

图 5-1　GridQoS 体系结构

5.3 GridQoS 的实现

本节从技术细节角度详细介绍 GridQoS 网格模拟器的实现。

5.3.1 SimJava 离散事件模型介绍

GridQoS的离散事件模拟是基于SimJava[50]离散事件模型。SimJava 的设计目标是基于C++的 ASE++库，建立一个纯文本的 Java 离散事件模拟库。SimJava模拟包含许多能够并

行运行且具有独立线程的实体。这些实体通过端口（Sim_port类）连接，并借助端口发送和接收事件实体（Sim_event类）来实现通信。整个模拟的过程通过实体生成的跟踪消息记录并保存至文件。

构造一次模拟包括以下步骤：

1) 编程实现模拟实体的行为，通过继承Sim_entity标准类和重写body()方法实现；

2) 向Sim_system对象中添加实体实例，通过方法Sim_system.add(entity)实现；

3) 使用方法Sim_system.link_ports()将实体的端口连接起来；

4) 最后，通过调用Sim_system.run()设置模拟动作。

举一个模拟的例子。模拟两个实体：发送者、接收者，发送者分发100条消息给接收者，等待确认，并在每两次分发之间等待10个模拟时间单位。

主函数如下：

```
1)  import eduni.simjava.*;
2)  class Example {
3)    public static void main(String args[]) {
4)      Sim_system.initialise();
5)      Sim_system.add(new Source("Sender", 1, Source.SRC_OK));
6)      Sim_system.add(new Sink("Receiver", 2, Sink.SINK_OK));
7)      Sim_system.link_ports("Sender", "out", "Receiver", "in");
8)      Sim_system.run();
9)    }
10) }
```

该main函数基本步骤实现为：首先，通过调用Sim_system.initialise()初始化Sim_system对象；随后，有两个实体被分别添加，它们通过new方法调用各自的构造器；之后，这些实体通过调用Sim_system.link_ports()联系起来，它将“Sender”实体的“out”端口与“Receiver”实体的“in”端口连接；最后，通过调用Sim_system.run()运行模拟，直到所有的事件完成则运行退出。

两个实体类从Sim_entity类继承，发送者代码实现如下：

```
1)  class Source extends Sim_entity {
2)    private Sim_port out;
3)    private int index;
4)    private int state;
5)    public static final int SRC_OK      = 0;
6)    public static final int SRC_BLOCKED = 1;
7)    public Source(String name, int index, int state) {
8)      super(name);
9)      this.index = index;
10)     this.state = state;
11)     out = new Sim_port("out");
12)     add_port(out);
13)   }
```

```
14)     public void body() {
15)         Sim_event ev = null;
16)         int i;
17)         System.out.println("About to do body S");
18)         for (i=index; i<100; i++) {
19)             sim_schedule(out,0.0,0);
20)             sim_wait(ev);
21)             state = SRC_BLOCKED;
22)             sim_hold(10.0);
23)             state = SRC_OK;
24)             sim_trace(1,"C Src loop index is "+i);
25)         }
26)       System.out.println("Exiting body S");
27)     }
28)  }
```

构造器方法首先调用Sim_entity的构造器，即super(name)，从Sim_entity继承的所有类都需要做同样的调用；之后，初始化私有的两个数据成员：state和index；最后创建端口“in”并将它添加至端口列表。

标准的Sim_entity实体是不做任何事情的，要使得一个实体实现任何有意义的动作需要重写body()方法。发送者body()功能包含了所有simjava的重要方法：

- ✓ sim_schedule(Sim_port port, double delay, int tag)向端口连接的实体发送一条消息，要求在从现在开始之后delay长度的模拟时间单元内完成，并伴随标志位tag；
- ✓ sim_wait(Sim_event ev)等待一个使用了sim_schedule()的发送事件（本例中是等待接收器实体）；
- ✓ sim_hold(double t)使实体阻塞t模拟时间单元；
- ✓ sim_trace(int level, String msg)为跟踪文件添加msg。

5.3.2 GridQoS 网格实体逻辑设计

一个模拟仿真环境需要抽象出所有相关实体以及它们在实际系统中的交互行为。基于对GridSim的扩展，GridQoS模拟器支持的实体抽象包括：网格用户、网格作业、网格资源、网格信息服务，以及两大功能模块：作业管理模块和调度模块。其实体的逻辑交互如图5-2所示：

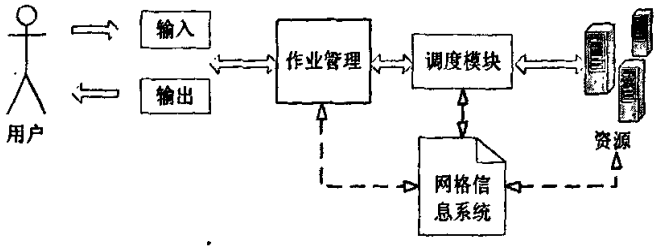

图 5-2　GridQoS 实体模拟流程

实体间的逻辑交互流程可以概括为以下步骤：

1) 用户提交任务至任务管理模块，作业管理模块负责接收作业和维护作业的需求参数；
2) 作业管理模块将任务提交至调度模块；
3) 调度模块通过资源信息中心获取资源信息；
4) 调度器根据映射结果，将任务调度到指定的资源；
5) 资源将任务执行结果发送至任务管理模块；
6) 任务管理模块根据任务执行结果将任务完成情况反馈至资源信息中心，资源信息中心将据此更新资源相关信息。

对各实体及模块的详细介绍如下：

网格用户(user)

每一个用户实体都代表了一名网格用户，他具有如下的数据结构：

- ✓ 用户ID：UserID
- ✓ 任务队列：GridletList
- ✓ 调度算法：表明对具体服务质量的偏好，比如：价格最小化或者时间最小化；
- ✓ 活动频率：即该用户创建作业的频率，这一点也正是作业到达率的反映。

网格任务（Gridlet）

这里的网格任务考虑元任务（任务之间没有通信关系），它是调度的最小单元，它维护了如下数据结构：

- ✓ 任务ID：GridletID
- ✓ 用户ID：UserID
- ✓ 任务描述信息：任务长度（length），任务到达时间，输入输出参数；
- ✓ 服务质量需求描述：硬件要求，软件要求，完成时间限制，价格限制，安全级别要求等。

资源(resource)

资源实体的每个实例代表了一个网格资源，网格资源提供如下信息：

- ✓ 资源ID：resourceID
- ✓ 资源描述信息：resourceCharacteristics，该信息包含了对资源具体服务内容的描述
  - 机器种类；
  - 处理器数目；
  - 处理器单位时间价格；
  - 处理器处理速度；
  - 资源安全级别；
  - 负载信息；

- 共享资源或独占资源信息；
- 存储器空闲空间信息。

网格信息服务(Grid Information Service，GIS)

提供资源注册服务，保存网格可用资源的列表，供调度模块查询，他维护如下信息：

✓ 可用资源列表：ResourceList；

✓ 资源平均QoS向量averageQoS[]，该向量反映了资源的平均服务水平。

作业管理模块

作业管理模块负责接收提交的作业以及获取作业处理结果，它维护如下信息：

✓ 未处理作业列表：UnfinishedtList；

✓ 处理后的作业列表：ReceivetList。

输入和输出(Input and Output)

GridQoS实体之间的信息传输通过输入和输出实体，他们的I/O通道或端口就是到输入和输出实体的连接。独立的 Input 和 Output 实体可以模拟双工通信和多用户并行通信，也可以模拟通信延迟。

5.4 基于 GridQoS 模拟实现网格调度算法的实例

本节详细描述了本文研究工作中调度算法的模拟实现。需要说明的是，为了顾及批调度对系统整体性能的优化要求，我们采用了集中式的解决方案，即调度模块对作业进行集中式映射和调度。

整体上的流程如图 5-3 所示，黑色的箭头代表主要的执行流程，其中，资源初始化、调度、任务结果处理三个部分都需要与网格信息服务这一实体交互。

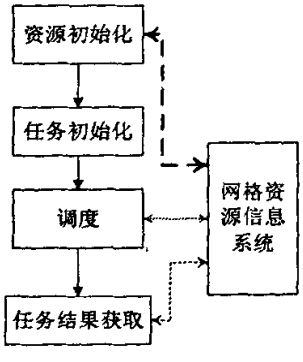

图 5-3　GridQoS 模拟调度算法流程图

5.4.1 资源场景设置

程序的第一步是资源场景的初始化，这一步通过ResourcesBehavior类实现。主要步骤如下：

1) 实例化ResourceBehavior类；
2) 调用initResourcesScenario(RESOURCE_NUM)方法，初始化资源QoS参数；
3) 向网格信息系统注册资源信息；
4) 调用printResourceList方法，存储初始化后的资源，以保证可重复地使用相同参数资源进行模拟实验。

一个包含5个资源实例的资源信息列表如表5-1所示：

表 5-1　资源信息列表示例

资源名称	资源ID	体系结构	CPU MIPS	每秒花费	安全级别
Resource_0	5	HP Superdome	408	2.932540752146214	2
Resource_1	9	Sun Ultra	411	2.8875024506102083	2
Resource_2	13	HP Integrity	439	2.9859543298619777	1
Resource_3	17	Compaq AlphaServer	416	2.7334761308096303	1
Resource_4	25	Intel Pentium	419	3.102617585221737	0

5.4.2 任务初始化

任务初始化主要步骤如下：

1) 初始化用户实体；
2) 在用户实体的构造函数中调用createGridlet方法，初始化任务队列；
3) 在任务队列中实例化任务，初始化每个任务QoS信息；
4) 存储任务信息，以保证可重复地使用相同参数的任务进行模拟实验。

一个包含5个作业信息的作业信息列表如表5-2所示：

表 5-2　作业信息列表示例

作业ID	体系结构	作业长度	安全	完成时间要求	花费要求
Gridlet_0	Compaq AlphaServer	1721.9222339651858	1	31.183582714984055	97.82052583832514
Gridlet_1	HP Superdome	1028.8511728901274	1	53.46354831828167	55.22273359486379
Gridlet_2	HP Integrity	529.79499553245	2	9.891152476094607	28.37475335435658
Gridlet_3	Sun Ultra	326.3423959360962	1	6.808400434770348	14.34901027251403
Gridlet_4	HP Superdome	2582.943240880998	0	57.459077468555805	135.74162146530 58

5.4.3 调度算法实现

调度核心算法是在用户实体的body( )方法中实现的，为了对一个任务队列进行实验并能够模拟其到达率，我们假设只创建一个用户实例，调度算法作用于该用户提交的任务队列。具体实现分为在线调度方式和批调度方式。

➢ 在线调度算法：

核心伪代码如下：

```
1 while (arrival_job j){
    // get resources list from grid information service and filter the unmatched
2   if (GIS is available) {
3     resources = getResourceList();
4   }
5   matchedResources = filterNotmatched(j, resources);
6   //submit the job to the resource with maximal expected utility
7   max = 0;
8   foreach (x in matchedResources){
9     if (evaluateUtility(j, x)>max){
10      target = x;
11    }//end if
12  }//end for each
13  gridletSubmit(j, target);
14  // update related information after job submitting
15  updateGridletInfor(j);
16  updateResInfor(target);
17 } // end while
```

主要步骤描述如下：

1) 用户实体向网格信息系统GIS发出资源信息请求，获取资源列表（步骤3）；
2) 根据作业的系统QoS需求进行硬性过滤(步骤5)；
3) 根据具体调度策略，根据作业需求求出具有最大效用值的资源（步骤8~11）；
4) 通过GridletSubmit()方法，将任务提交至指定资源（步骤13）；
5) 根据提交的情况修改任务、资源信息（步骤15、16）。

➢ 批调度算法：

核心伪代码如下：

```
1 while (arrival_job j && time<INTERVAL){
2   gridletlist.add(j)
3 }
4 foreach(j in gridletlist){
5   if (GIS is available) {
6     averageQoS = getAverageQoS();
```

```
7    }
8    userSatisfaction = averageUserSatisfaction(j, averageQoS);
9    rank = caculateRank(userSatisfaction);
10   Gridletlist[rank].add(j);
11  }
12 foreach(list k in Gridletlist[])
13   while (Gridletlist[k].size() != 0){
14      foreach(j in Gridletlist[k]){
15          // get resources list from grid information service and filter the unmatched
16          if (GIS is available) {
17              resources = getResourceList();
18          }//end if
19          matchedResources = filterNotmatched(j , resources);
20          //record the resource with minimal processing time
21          min = 0;
22          foreach (x in matchedResources){
23              if (evaluateUtility(j, x)<min){
24                  target = x;
25              }//end if
26          }//end for each
27      }//end for each
28      j = getMinTime(Gridletlist[k]);
29      gridletSubmit(j, target);
30      // update related information after job submitting
31      updateGridletInfor(j);
32      updateResInfor(target);
33      Gridletlist[k].remove(j);
34   }//end while
35 }//end for each
```

主要步骤描述如下：

1) 在规定的批处理事件间隔时间INTERVAL内初始化批队列gridletlist（步骤1~3）；
2) 从GIS获取平均资源性能，对批作业求对资源平均性能的用户满意度（步骤8）；
3) 根据平均性能的用户满意度计算批作业所属的优先级别（步骤9）；
4) 将作业根据其优先级放入对应的批队列中（步骤10）；
5) 将批队列按优先级从高到低依次进行批调度（步骤12）；
6) 按照Min-Min算法，将作业调度到其匹配的资源上（步骤13~34），其中资源过滤策略相似于在线调度算法。

5.4.4 执行结果获取

获取任务执行结果的步骤如下：

1) 通过GridletReceive()方法接收从资源实体返回的任务执行结果；
2) 修改完成任务的资源的负载信息，同时修改网格资源信息系统对应的资源信任值；
3) 通过recordStatistics()方法将执行的信息记录至任务实体中；
4) 将已完成的任务添加至receiveList队列；
5) 通过评测参数要求，打印任务执行结果。

5.5 本章小结

本章详细介绍了验证本文研究工作的基础：GridQoS模拟器的设计与实现。我们从设计目标、体系结构、具体实现细节等方面着手，介绍GridQoS模拟器。在本章的最后，我们以本文研究工作为例，给出了模拟器进行调度算法模拟的实现实例。

第六章 结束语

6.1 本文工作总结

网格计算的初衷，是人们希望能够实现廉价、普遍的高性能计算，能够合作存取各种数据信息，能够提供广域多媒体应用等等。随着对网格计算的进一步认识，人们发现，网格计算环境相对于一般网络计算环境有着更为复杂的特征，如存在多管理域和站点自治，系统的动态性、异构性和通信延迟的不确定性更高，硬件和软件两个层次上都存在异构性等等。因此，实现有效的网格计算有很多需要解决的问题，具体包括任务调度和资源管理、系统安全、编程模式、性能评测和数据存取等。

任务调度承担了协调资源、调度作业的任务，作为网格资源管理的一个部分，它是网格系统中的关键环节。随着网格 QoS 的渐受关注，如何在网格系统中提供 QoS 支持成为近年来的研究热点。而如何在任务调度中提供 QoS 支持，则成为网格系统是否能够提供非平凡服务的关键。研究任务调度中的 QoS 问题，涉及到网格 QoS 的描述与度量，也涉及到具体的任务调度算法。基于此，本文研究网格环境下提供服务质量支持的任务调度问题，主要内容包括以下几个方面：

1. 介绍网格的基本概念，继而介绍网格资源管理的概念以及其中的任务调度问题，并提出提供 QoS 支持的网格任务调度问题；
2. 综述网格环境下的任务调度机制，包括网格任务调度特点和目标、现有的网格任务调度机制研究状况、现有的调度系统等，并在最后描述了提供 QoS 支持的网格任务调度问题及其研究意义；
3. 从概念、网格应用的 QoS 需求等方面介绍网格 QoS 问题，研究并提出网格 QoS 描述方法，并进而提出网格 QoS 度量方案；
4. 研究网格环境下任务调度问题，提出提供服务质量支持的任务调度算法，并对其进行模拟验证；
5. 研究网格环境模拟问题，基于 SimJava 离散事件模型，设计与实现网格模拟器 GridQoS，并给出具体的模拟实例。

6.2 本文的主要贡献

本文所做的主要贡献如下：

1. 提出了基于层次的网格 QoS 描述方法和度量机制

本文从应用、系统、资源这三个层次来看待基于 QoS 的网格资源管理，提出了从网格应用层、虚拟组织层、资源设备层这三个层次的网格 QoS 描述方法，并展现了三个层次 QoS 之间的映射关系。

为了综合性的评价多个性能目标，本文提出了一种 QoS 度量机制，该机制将虚拟组织层各类 QoS 定量地描述出来，并与应用层用户满意度相联系，从而解决了网格 QoS 的统一度量问题。

2. 提出了网格环境下提供 QoS 支持的任务调度算法

本文基于层次化的网格 QoS 描述与度量方法，分别针对在线调度和批调度提出了提供 QoS 支持的任务调度算法，使得在线调度能够综合性地考虑用户 QoS 需求，也使得批调度在优化系统性能的同时兼顾了用户 QoS 需求。模拟结果表明，提供服务质量支持的在线调度算法在系统吞吐率与传统 MCT 算法平均相差 8%的情况下，用户满意度有平均 44%的提高；而提供服务质量支持的批调度算法相对于传统 Min-Min 算法，系统吞吐率损失可以忽略不计，而用户满意度则平均提升了 17%。

3. 设计与实现网格模拟器 GridQoS

GridQoS 网格模拟器是 GirdSim 网格模拟器的扩展实现，该模拟器能够模拟网格用户、网格任务、网格资源、网格信息系统等实体以及实体间的通信交互；同时，该模拟器提供对 QoS 描述的支持并提供对在线调度、批调度的模拟。该模拟器有效地解决了本文研究工作中的验证问题，同时它对于一般的网格环境模拟也具有普遍的应用意义。

6.3 今后的研究工作

随着网格环境下资源共享的广泛化，网格 QoS 以及网格调度问题的研究在未来依然是一个非常重要的方面。本文的工作由于时间等诸多限制，尚有许多有待完善之处，因此在今后的研究工作中，我们将主要围绕以下一些方面展开：

1. 网格 QoS 各层次间映射机制的研究与实现。

本文的研究工作中，各层次间的 QoS 都假设为一些简单的映射，但在实际的网格环境中，由于资源的广泛性以及大量用户的参与，如何让各层次的资源与用户达成对 QoS 一致的理解往往需要通过 SLA（服务等级协议）实现。因此，引入 SLA 解决各层次间 QoS 映射问题将是下一步的研究重点。

2. 调度中硬 QoS 保证问题

提供严格服务质量保证的网格系统中，如果系统无法按照 QoS 协议完成已接受的任务，网格服务商将会承担赔偿的风险，而迄今为止，本文在调度中的 QoS 研究仍然是提供软 QoS 支持，如何在调度中提供硬性 QoS 支持，使得网格系统能够严格满足用户需求，将是下一步研究工作的重点。

3. 仿真工作

本文的工作是基于 GridQoS 模拟器进行的模拟验证，模拟的资源、任务参数都是随机生成。随着网格应用的广泛化，如何引入实测数据对研究工作进行验证将变得可行。因此，采集实测数据并修改模拟器使之能够模拟实际应用需求也是下一步研究工作的重点。

参考文献

[1] Ian Foster and Carl Kesselman, editors. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, Inc., 1999. ISBN 1-55860-475-8.

[2] Ian Foster, C. Kesselman, and S. Tuecke. The anatomy of the Grid: Enabling scalable virtual organizations. *International J. Supercomputer Applications*, 15(3), 2001.

[3] Friedhelm Ramme. Building a Virtual Machine Room - a Focal Point in Metacomputing. Future Generation Computer Systems (FGCS), 11:477-489, 1995.

[4] H. G. Dietz, W. E. Cohen, and B. K. Grant. Would you run it here... or there? (AHS: Automatic Heterogeneous Supercomputing). In Int. Conf. Parallel Processing, Vol II: Software, pages 217-221, 1993.

[5] Mark A. Baker and Geoffery C. Fox. Metacomputing: Harnessing Informal Supercomputers. High Performance Cluster Computing. Prentice-Hall, May 1999. ISBN 0-13-013784-7.

[6] Larry Smarr and Charles E. Catlett. Metacomputing. Communications of the ACM, 35(6): 44-52, June 1992.

[7] Grid today. http://www.Gridtoday.com

[8] http://www.Globus.org/toolkit

[9] Foster I, Kesselman C. The Grid 2, Blueprint for a New Computing Infrastructure [M]. San Francisco: Morgan Kaufmann Publishers Inc. 2004.

[10] Jarek N, Jennifer M, Jan W. Grid Resource Management: State of the Art and Future Trends [M]. Norwell: Kluwer Academic Publishers, 2004.

[11] Foster I, Kesselman C, Nick JM, Tuecke S. The physiology of the grid: An open grid services architecture for distributed systems integration. 2002. http://www.gridforum.org/ogsi-wg/drafts/ogsa_draft2.9_2002-06-22.pdf

[12] Tuecke S, Czajkowski K, Foster I, Frey J, Graham S, Kesselman C, Maquire T, Sandholm T, Snelling D, Vanderbilt P. Open grid services infrastructure (OGSI) Version 1.0. 2003. http://forge.gridforum.org/projects/ggf-editor/document/draft-ogsi-service-1/en/1

[13] Czajkowski K, Ferguson DF, Foster I, Frey J, Graham S, Seduknin I, Snelling D, Tuecke S, Vambenepe W. The WS-resource framework (Version 1.0). 2004. http://www-106.ibm.com/developerworks/ library/ws-resource/ws-wsrf.pdf

[14] http://www.csse.monash.edu.au/~davida/nimrod/nimrodg.htm

[15] Rajkumar Buyya, Steve Chapin, and David DiNucci, Architectural Models for Resource Management in the Grid, The First IEEE/ACM International Workshop on Grid Computing (GRID 2000), Springer Verlag LNCS Series, Germany, Dec. 17, 2000, Bangalore, India.

[16] S. Chapin, M. Clement, Q. Snell, "A Grid Resource Management Architecture," Draft of

Scheduling Working Group of Grid Forum. Available from http://www.Gridforum.org Oct. 1999.

[17] Karl Czajkowski, Ian Foster, Nick Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. A Resource Management Architecture for Metacomputing Systems. In Dror G Feitelson and Larry Rudolph, editors, JSSPP'98, number 1459 in LNCS, pages 62-82. Springer-Verlag Berlin. Heidelberg, 1998.

[18] Tannenbaum T，Foster I，Livny M，et al. Condor-G：A Computation Management Agent for Multi—Institutional Grids[J]. Cluster Computing，2002，5(3)：237—246.

[19] W Li，z Xu，F Dong，et al. Grid Re~urce Discovery Based on Routing transferring Model [C]. Th e 3rd ACM / IEEE International Work shop on Grid Computing. 2002. 145—156.

[20] 李伟，徐志伟. 一种网格资源空间模型及其应用[J]. 计算机研究与发展，2003，40(12)：1757—1763.

[21] T. L. Casavant and J. G Kuhl. A Taxonomy of Scheduling in General-Purpose Distributed Computing Systems, May 1986.

[22] Warren Smith, Ian Foster, and Valerie Taylor. Predicting Application Run Times Using Historical Information. In Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing, 1998.

[23] Nabrzyski J, Jennifer M S, Weglarz J. Grid Resource Management: State of the Art and Future Trends[M]. Boston: Kluwer Academic Publishers. 2003. 36—44.

[24] Krysztof K. JareK N，Julinusz P. User Preference Driver Muhiobjective Resource Management in Grid Environments[C]. The 1st International Symposium on Cluster Computing an d the Grid Brisban e，2001. 114—122.

[25] Jarek Nabrzyski. Knowledge-based Scheduling Method for Globus [EB / OL]. http：// www.man.~zanan.pl/metacomputing/ai—meta/0busnew/index.htm，1999.

[26] Spring N，Wolski R. Application Level Scheduling of Gene Sequence Comparison on Metacomputers [C]. Melbourne：the 12th ACM Int' 1 Conf. on Supercomputing. 1998. 141 —148.

[27] Rajkumar Buyya, David Abramson, and Jonathan Giddy. Nimrod/G: an architecture for a resource management and scheduling system in a global computational Grid. In Proceedings of the Fourth International Conference on High Performance Computing in Asia-Pacific Region, pages 283-289, Beijing, China, 2000.

[28] He, X., Sun, X, and von Laszewski, G., QoS guided min-min heuristic for Grid task cheduling. Journal of Computer Science and Technology, 18, 2003, pp.442-451.

[29] D. P Spooner, S. A Jarvis, J Cao, S Saini, G R Nudd, "Local Grid Scheduling Techniques using Performance Prediction," IEE Proc. Comp. Digit. Tech., 150(2):87-96, 2003.

[30] L. He, S. A. Jarvis, D. P. Spooner, X. Chen, and G. R. Nudd. Dynamic Scheduling of Parallel Jobs with QoS Demands in Multiclusters and Grids, Pittsburgh, USA. In 5th IEEE/ACM International Workshop on Grid Computing (Grid2004), 2004.

[31] Crawley E, Nair R, Rajagopalan B, Sandick H. A framework for QoS-based routing in the internet. IETF RFC 2386, August 1998

[32] Hutchison D, Coulson G, Campbell A, et al. Quality of service management in distributed systems. In: Sloman M, ed. Network and Distributed Systems Management, Chapter 11. Addison Wesley, 1994.

[33] Foster I, Kesselman C, Lee C, Lindell B, Nahrstedt K, Roy A. A distributed resource management architecture that supports advance reservations and co-allocation. In: Proc. of the 7th Int'l Workshop on Quality of Service, Vol 13. 1999. 27-36. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=766475

[34] Foster I, Roy A, Sander V. A quality of service architecture that combines resource reservation and application adaptation. In: Proc. of the 8th Int'l Workshop on Quality of Service. 2000. 181-188.

[35] Roy A, Foster I, Gropp W, Karonis N, Sander V, Toonen B. MPICH-GQ: Quality-of-Service for Message Passing Programs. San Diego: IEEE Computer Society Press, 2000. 19-30.

[36] Al-Ali R, ShaikhAli A, Rana O, Walker D. Supporting QoS-based discovery in service-oriented Grids. In: Proc. of the Int'l Parallel and Distributed Processing Symp. (IPDPS 2003). Nice: IEEE Computer Society, 2003. 101-109

[37] Irvine CE, Levin T. Toward quality of security service in a resource management system benefit function. In: Proc. of the 15th Annual Computer Security Application Conf. IEEE Computer Society, 2000. 133-139. http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=843738

[38] Irvine CE, Levin T. An approach characterizing resource usage and user preferences in benefit functions. Technical Report, NPS-CS-99-005, NPS, 1999.

[39] Kim JK, Kidd T, Siegell HJ, Irvine CE, Levin T, Hensgen DA, St. John D, Prasanna VK, Freund RF, Porter NW. Collective value of QoS: A performance measure framework for distributed heterogeneous networks. In: Proc. of the Int'l Parallel and Distributed Processing Symp. (IPDPS). San Fransisco: IEEE Computer Society, 2001. 137-150.

[40] Chatterjee BSS, Sydir MDJJ, Lawrence TF. Taxonomy for QoS specifications. In: Proc. of the 3rd Int'l Workshop on Object-Oriented Real-Time Dependable Systems (WORDS'97). Newreport Beach, 1997. 100-107.

[41] Buyya R, Abramson D, Venugopal S. The Grid economy. Proc. of the IEEE, 2005,93(3):698-714.

[42] Azzedin, F. and M. Maheswaran, Towards Trust-Aware Resource Management in Grid Computing Systems, in 2nd IEEE/ACM International Symposium on Cluster Computing and the Grid. 2002. p. 452-452.

[43] Graham, R.L., et al., Optimization and approximation in deterministic sequencing and scheduling: a survey. Ann. Discrete Math, 1979. 4: p. 287-326.

[44] Maheswaran M, Ali S, Siegel H J, et al, Dynamic mapping of a class of independent tasks onto heterogeneous computing systems. In the 8th IEEE Heterogeneous Computing Workshop (HCW '99),San Juan, Puerto Rico, Apr. 1999, pp.30-44.

[45] M. Pinedo, "Scheduling: Theory, Algorithms, and Systems," Prentice Hall, Englewood Cliffs, NJ,1995.

[46] Henri Casanova, Graziano Obertelli, Francine Berman and Rich Wolski. The AppLeS parameter sweep template: User-level middleware for the Grid. In Proc. the Super Computing Conference (SC'2000), Dallas, Texas, Nov. 2000.

[47] Gong L., Sun X.H., and Waston E. Performance modeling and prediction of non-dedicated network computing. IEEE Trans. on Computer, September, 2002, 51(9): 1041-1055.

[48] Sun Xian-He, Wu Ming, GHS: A performance prediction and task scheduling system for Grid computing. In Proc. of 2003 IEEE International Parallel and Distributed Processing Symposium (IPDPS 2003), Nice, France, April, 2003.

[49] Rajkumar Buyya, and Manzur Murshed, GridSim: A Toolkit for the Modeling, and Simulation of Distributed Resource Management, and Scheduling for Grid Computing, The Journal of Concurrency, and Computation: Practice, and Experience (CCPE), Volume 14, Issue 13-15, Pages.

[50] http://www.dcs.ed.ac.uk/home/hase/simjava/

致 谢

到致谢的这一刻，有千言万语难以言尽。我很庆幸有缘份在三年前的那个十月来到科大的网络中心，三年的时光转瞬即逝，而其间的经历却刻骨铭心。

首先必须要感谢的是我的导师杨寿保教授，感激他在我工作、生活上点点滴滴的指引：当我初入实验室时，他鼓励我积极接触实验室里各个研究方向；当我开题时，他听我做开题试讲并逐一指正不妥之处；当我第一次写论文，文字笨拙、逻辑混乱时，他不断的指正和引导我，让我的论文终于被接受和认可；当我在项目工作上勤奋或懒惰时，他及时肯定或提醒我，使我不敢懈怠、更加努力；……。我的每一点成长，都离不开杨老师的关怀与帮助，而杨老师认真严谨的治学态度、雷厉风行的行事作风、平易近人的长者风范也将是我一生的楷模。

与此同时，要感谢我们863项目组负责人陈华平教授，他认真负责的工作态度保证了项目工作的顺利进行，而他对项目工作远景的认识、对研究工作的深刻见解深深地影响和鼓舞着我。

感谢冯征师兄、杨法娜师姐以及池轶师姐，从在我大四的时候起，她们就常常给予我帮助；感谢刘鹏展师兄、申凯师兄，他们对技术的专注和执着、对网格研究的热忱深深地影响和鞭策着我，他们一直是我的良师益友；感谢郭磊涛、孙伟峰、大鹏、绍林、王菁、张蕾、董闻等众多师兄师姐，他们各自都有着独特的魅力并都拥有着一颗善良、热忱的心，感谢他们在学习、研究、生活上对我的帮助；也感谢实验室里众多师弟师妹，他们的出现让实验室充满了生机与希望，也让我深深体会到责任的意义。

感谢实验室里我最好的伙伴：赵叶红、姜燕、徐钽、史凌志。他们总是无微不至地帮助、关怀和鼓励我。无论是成功或失败，我们都一起分享，而我们一起玩耍、生活、学习、工作的日子总让我常生“但愿人长久”之叹。也感谢与我同级的好朋友韦冬、路卫娜，在三年相处中，他们曾给予我那么多的帮助。

总之，感谢实验室的众多兄弟姐妹，我们彼此之间的情谊将是我一生的财富！

另外要感谢0411的同学们，在那些一起学习一起找工作的日子里，我们曾彼此鼓励、共同进步。

也要感谢网络中心的顾老师、夏老师、封老师、孟老师等，他们总是热心地在工作上、生活上为我提供帮助。

末了，要深深地感谢我的父母，他们总是一如既往地帮助、鼓励和包容我，总是乐于与我分享生活中的点滴；最后，也感谢我的男朋友，他的鼓励、他富有创意的种种想法、他对事物的深刻见解都极大地推动着我的进步。

张然美

二〇〇七年五月一日

读研期间发表及录用的论文

网格环境下一种 QoS 感知的批调度算法

张然美，杨寿保，申 凯，郭磊涛

《小型微型机系统》，已录用

读研期间参与的科研项目

✧ **2007.1 至今，国家 863 项目：**

基于应用调度的网格服务环境及若干网格应用的研制 (2006AA01A110)

- 应用调研与项目申请；
- 应用研制部署以及简单元调度实现；
- 研究网格服务环境中的任务调度机制。

✧ **2006.2 至今，国家自然科学基金项目：**

网络计算环境中信任感知的资源交易模型 (60673172)

- 研究资源选择算法；
- 调研并参与模拟器实现。

✧ **2004.9～2005.12，国家自然科学基金项目：**

基于计算市场模型的网络资源管理 (60273041)

- 参与结题工作。

✧ **2004.9～2005.12，国家 863 项目：**

合肥网格结点建设及若干典型网格应用研制 (2002AA104560)

- 基于 Jetspeed 项目进行合肥网格结点 Portal 开发；
- 调研与开发网格应用；
- 部署和维护合肥网格结点监控系统。