摘要

网格计算是典型的分布式计算，网格系统是典型的分布式系统。近年来对于网格环境下各种技术的研究已经成为计算机领域的一个热点问题，其中的任务调度，任务协同以及资源协同等问题更是引起了中外学者的关注，并且产生了很多基于网格环境下的任务调度算法。

目前存在的网格环境下的任务调度算法大部分都是静态的启发式算法。事实证明网格环境下的任务调度是 NP 完全问题(除极少数特殊情况外)。而任务调度的最直接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。

本文首先比较分析了网格计算和对等计算(P2P)的异同点，并在网格计算中应用 P2P 技术，提出了一种新的资源组织管理模型－P2P_Grid 模型。该模型中的超级 Peer 将大型的网格系统划分为若干个小规模的子网格系统，每个超级 Peer 是其所属子网格系统的控制中心；而不同子网格系统间的超级 Peer 是对等的。这样每个超级 Peer 对于局部任务既可以完成集中式调度，又可以与其他空闲超级 Peer 协同工作完成分布式调度。

接着，在该模型的基础上，改进了经典的 Min_min 任务调度算法，命名为 P_G_Min 算法。该算法利用了 P2P_Grid 模型既集中又分布的特点，对于用户所提交的任务总是在其所属的子网格系统内进行调度，而超级 Peer 也总是将局部网格系统的资源分配给待调度的任务，这样既降低了任务调度的完成时间，同时提高了系统资源的利用率。

最后，我们在 GridSim 模拟器上对算法进行了仿真试验。我们首先在模拟网格平台上实现了 Min_min 算法，并对其进行了改进，提出了 QOS－Min_min 算法，进行了仿真试验，给出了试验数据和比较结果。最后对 P_G_Min 算法进行了仿真试验，与传统的 Min_min 算法和 QOS-Min_min 算法进行分析比较，分析发现 P_G_Min 算法较 QOS-Min_min 算法和 Min_min 算法在任务调度平均完成时间和系统资源利用率两个性能上都有很大的改善。

关键字：网格计算，对等计算，超级 Peer，任务调度，资源调度

Abstract

Grid Computing is a typical Distributing Computing and Grid Computing system is a typical Distributing system. It has been a hot point about Grid Computing various technologies in the Computer filed. Many key technologies were aroused attention of researchers such as task scheduling, task cooperation and resource cooperation, etc. At the same time many task scheduling algorithms based on Grid Computing had been brought.

At present, the great mass of existed task scheduling algorithms based on Grid Computing are static heuristic algorithm. The task scheduling based on Grid Computing has been proven to be NP-Complete for most cases (besides rarely especial case). The direct aim about task scheduling is to realize optimization scheduling and try to improve Grid System throughput.

Firstly, the similarities and differences were compared between Grid Computing and Peer-to-Peer, applied P2P to Grid Computing and brought a new resource organization management model-P2P_Grid model. Super-Peer divided large-scale Grid System to some small-scale sub-Grid system in the new model, and each Super-Peer was control center of sub-Grid system that it belonged; all Super-Peers in the different sub-Grid system were peer to peer. So, each Super-Peer not only could complete concentrated scheduling for local tasks but also could cooperate with other idle Super-Peer to complete distributing scheduling.

Secondly, a new task scheduling algorithm based P2P_Grid model was improved, named P_G_Min algorithm. The new algorithm took advantage of P2P_Grid model's characteristic to schedule the tasks that users submitted in sub-Grid system, at same time Super-Peer always designed local resources to scheduling tasks. As a result, the completion time of tasks scheduling to be reduced and the utilization of system resource to be improved.

Finally, three algorithms were tested on GridSim simulator. First, Min_min algorithm was tested by using the simulator, then the Min_min algorithm was modified and QOS-Min_min algorithm was draw out. QOS-Min algorithm was also simulated on GridSim simulator and testing data was registered, comparing outcome with conventional Min_min algorithm. At last, the P_G_Min algorithm was compared with traditional Min_min algorithm and QOS-Min algorithm by simulate experimentation. It shows that the new algorithm has a better quality of system load balancing and the utilization of system resource.

Key words: Grid Computing, Peer-to-Peer (P2P), Super-Peer, Task Scheduling, Recourse Scheduling

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。尽我所知，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含本人为获得江南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。

签名：[illegible]　日期：2007 年 5 月 29 日

关于论文使用授权的说明

本学位论文作者完全了解江南大学有关保留、使用学位论文的规定：江南大学有权保留并向国家有关部门或机构送交论文的复印件和磁盘，允许论文被查阅和借阅，可以将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、汇编学位论文，并且本人电子文档的内容和纸质论文的内容相一致。

保密的学位论文在解密后也遵守此规定。

签名：[illegible]　导师签名：[illegible]

日期：2007 年 5 月 29 日

第一章 绪 论

1.1 引言

网格计算和对等计算都是新型的分布式系统，它们都是分布式的子集[1]，将两种技术结合起来研究，是分布式系统下的任务调度的一个新的探索方向。

1.2 研究背景

随着网络的不断发展，传统的网络技术已经不能满足用户日益增长的网络资源的需求，对于更广范围内的网络资源的共享的要求愈来愈迫切，基于这一点，一种异构的网络体系结构产生了，即分布式系统。分布式系统的产生也带来了其一系列相关技术的产生，比如网格技术。在分布式系统中，存在许多关键技术，其中任务调度算法就是其研究热点之一。

网格正逐步成为一种新的技术和基础设施[2]，可以充分利用集成的资源形成一个大规模的计算池，其目的是为了在分布、异构、自治的网络资源环境上构造动态的虚拟组织，并在其内部实现跨自治域的资源共享与资源协作，有效地满足面向互联网的复杂应用对大规模计算能力和海量数据处理的需求。网格计算的理想目标[3]是使网络上的所有资源易于协同工作，服务于不同的网格应用，实现资源在跨组织(自治域)之间应用的共享与集成。

实际上，网格计算是分布式计算的一种，类似于 MPP(极度并行计算机)。 这种计算模式是利用互联网把分散在不同地理位置的电脑组织成一个“虚拟的超级计算机”其中每一台参与计算的计算机就是一个“节点”，而整个计算是由成千上万个“节点”组成的“一张网格”。由于网格资源具有广域分布、异构、动态等特性，所以，好的任务调度和资源调度策略显得尤其重要。

无论是早期的计算网格还是目前的服务网格，都面临着如何有效地管理和调度网格任务和资源问题[4]。网格调度与本地调度的主要区别在于其调度的对象跨越多个管理域，不同的组织采用不同的策略操作其资源，而且资源用户和资源提供方的目的可能不一致甚至相互抵触。网格调度问题的最一般的目标函数是 Makespan，即调度系统有效地分配网格资源，实现在整个系统内网格应用任务的完成时间最小。通常是发现适合于给定任务的潜在资源集合，从那些资源中选择合适的资源子集，这些资源满足一个预先定义好的调度约束，找到一个这样的 Makespan 是 NP 完全问题[5]。

目前，国内外在这一领域已经做了大量的研究工作，并且形成了许多重要的理论。但是从对现在已经存在的异构环境下的任务调度算法的分析中可以看出这些算法本身都存在各自的缺陷，对于不同的任务调度算法，其各自的侧重点不同，但是总的来说，

一个好的任务调度算法其关键的技术参数是在最短的时间内实现最优分配策略，从而提高系统资源的利用率以及保持较好的系统负载平衡。

1.3 选题意义及国内外研究现状

网格计算(Grid Computing)是当前互联网研究中的一个热点，也是并行和分布处理技术的一个发展方向。在网格计算中，任务管理、任务调度和资源管理是网格必须具备的三个基本功能。用户通过任务管理系统向网格提交任务、为任务指定所需资源、删除任务并监测任务的运行状态。用户提交的任务由任务调度系统按照任务的类型、所需资源、可用资源等情况安排运行日程和策略。而资源管理系统监测网格资源状况，收集任务运行时的资源占用数据等[6]。由于网格计算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[7]。

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，使得任务调度变得极其复杂，不好的任务分配策略，将会增加任务的执行时间、降低整个网格系统的吞吐量。

任务调度是网格计算的基本功能，它面临的问题是一个 NP 完全问题。到目前为止，人们提出了很多网格系统的调度技术和算法。但是，不少调度算法还没有完整的理论依据、一些调度技术的结论还只是来自仿真结果，至今还没有形成网格计算的任务调度理论。

本课题所讨论研究的任务调度算法同样是在网格环境下进行的。网格计算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。在这样的一个前提下，本课题所提出的任务调度算法也是在保证该目标的基础上，将整个网络的负载平衡提高。由于网格是一个不断变化的系统，新节点的不断加入，拓扑结构的变化必然会带来资源分布的不均匀，那么在执行了一定的任务调度后，可能会造成整个网格系统负载极为不均衡。基于这样的思想，本课题所提出的任务调度算法是在以网格环境为基础，并提出一种新的资源组织管理模型上进行模拟的。新的模型将传统的集中式调度和分布式调度合二为一，对于局部网格，使用集中式调度；对于不同局部网格之间，使用分布式调度，这样就可以保持较好的网格系统负载平衡。

但对于网格系统，任务的调度是十分困难的，很多理论已经证明网格任务调度是一个 NP 完全问题，目前为止我们还不可能找到一个最优调度方案。而对于本课题中所提出的新的网格环境下的模型希望可以对网格任务调度的研究提供一些新的思想和新的思路，这对于研究先进的调度算法将是一个挑战和研究方向，因此选题具有一定的理论研究意义和现实价值。

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) [8-10]。采用分层模型的网格项目有 Globus[11], Legion[12], DataGrid[13],

AppLes[14]。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)[16, 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[18]。这种模型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[19, 20]。在织女星网格中提出的 EVP 模型包括有效资源层(E) 、虚拟资源层(V)和物理资源层(P) 。⑦OGSA 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，网格中的调度系统一般使用分布式调度策略，主要有如下几种调度模型：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，该方法难以解决协作配置问题。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。资源发现模型在一个分布式的数据库中维持资源属性和状态信息，该模型需要进一步研究具有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，目前还需要进一步研究权衡不同方案的组合。

1.4 论文的研究内容和结构

本文主要以网格环境为研究平台讨论了分布式系统下的任务调度算法。主要工作如下：

(1) 从概念、特点、目标、关键技术等几个方面分别详细分析了网格计算和对等计算这两种技术，并在此基础上对这两种技术进行了互补性的研究。对网格计算和对等计算结合进行了可行性研究讨论。

(2) 通过可行性研究，提出了本课题的分布式模型－P2P_Grid 模型。对该模型的特点、内部结构，资源的组织管理方式进行了深入的分析讨论；

(3) 使用 GridSim 模拟器模拟了网格环境，并在模拟器上仿真了 Min_min 算法。同时，对 Min_min 算法进行了改进，提出了 QOS-Min_min 算法和基于 P2P_Grid 模型的 P_G_Min 算法，并分别进行了仿真试验，比较三种算法的试验结果，给出了结论。

文章结构如下：

第一章：绪论。如上所述，介绍课题背景，选题意义及国内外的研究现状以及本文的研究内容和论文结构安排。

第二章：网格计算概述。本章主要包括了：网格的概念；网格的起源及研究现状；网格的基本要求、特点和关键技术。

第三章：P2P 技术概述。本章主要包括了两个部分：第一部分，对 P2P 的技术背景、概念、研究现状和关键技术等问题进行了阐述；第二部分，对 P2P 和网格计算进行的互补性比较研究，并提出了本课题所建立模型－P2P_Grid 模型。从研究背景、模型提出了理论依据、模型的资源管理模式等几个方面给于了深入的分析。

第四章：网格环境下任务调度算法。本章主要包括了三个部分：第一部分，介绍了网格任务调度特点、目标和目前存在的问题；第二部分，对目前存在的几种经典的网格任务调度算法进行的分析比较；第三部分，针对本课题提出的调度模型进行了详细的阐述，并给出了算法的简单思想。

第五章：基于 P2P_Grid 模型的任务调度算法的研究。本章主要包括了三个部分：第一部分，介绍了传统的启发式算法－Min_min 算法，并在 GridSim 模拟器上进行了仿真试验，给出了试验数据；第二部分，改进了 Min_min 算法，提出了 QOS-Min_min 算法，进行了仿真试验，并给出了与 Min_min 算法的比较结果；第三部分，提出了基于新模型的任务调度算法 P_G_Min 算法，进行了仿真试验，并对三个算法进行了分析比较，给出了比较结果；

第六章：总结和未来的工作。本章是对论文所做工作的完整总结，提出了进一步研究探索的方向。

第二章 网格计算概述

2.1 网格的概念

这一章首先对网格的基础知识做一个介绍。如前所述，网格是继 Internet 之后的又一次重大的科技进步，它代表了一种先进的技术和基础设施[22]。那么到底什么是网格呢？到目前为止，我们仍然不能给出一个精确的概念来定义网格，现存的各种网格的定义都是从某种实际应用的角度出发对其定义的。但是，从普遍意义来讲，一种广义的抽象的网格概念定义为：网格就是一个集成的资源与计算环境，或者说一个计算资源池。网格能够充分吸纳各种计算资源，并将它们转化成一种随处可得的、可靠的、标准的同时还是经济的计算能力。除了各种类型的计算机，这里的计算资源还包括了网络通信能力、数据资料、仪器设备，甚至是人等各种相关的资源。而网格计算就是基于网格问题的求解。

相对于网格的广义的定义，我们给出网格计算的广义定义。那么什么是网格计算呢？网格计算是指基于网格问题的求解。因此，本文所讨论的问题就是关于网格计算的问题。

而狭义的网格定义中的网格资源主要是指分布的计算机资源，进而狭义的网格计算的概念就是将分布的计算机组织起来协同解决复杂的科学与工程计算问题。

下面是关于网格的一些有代表性的观点[22, 23]：

(1) 网格就是方便资源管理、有效支持广域分布的、多领域的科学与工程问题解决的中间件系统。

(2) 网格就是在动态变化的、拥有多个部门或者团体的复杂虚拟组织(Virtual Organization)内，灵活、安全地协同资源共享与问题求解。

(3) 网格是建造分布式科学计算环境的一种一体化的集成方法，该环境包括计算、数据管理、科学仪器及人类的协作。

(4) 网格是一种无缝的、集成的计算与协作环境。

2.2 网格的起源及其研究现状

2.2.1 网格的起源

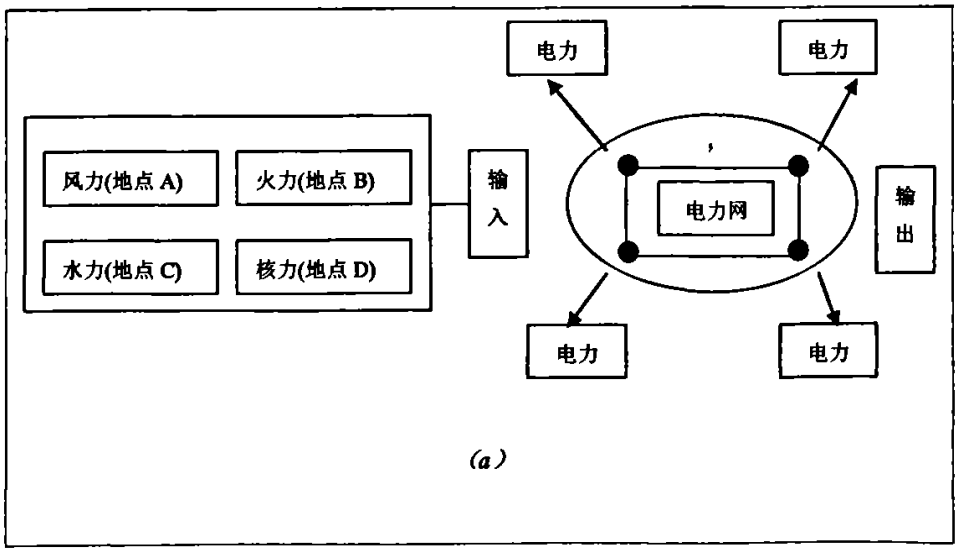
网格计算的起源是由于单台高性能计算机已经不能胜任一些超大规模应用问题的解决。于是人们想象分布在世界各地的超级计算机的计算能力能否通过利用广域互连技术使其像电力资源那样输送到每一用户，来求解一些大规模科学与工程计算等问题，从

而形成了计算网格(又称网格计算系统)。网格计算是作为虚拟的整体而使用在地理上分散的异构计算资源，这些资源包括高速互连的异构计算机、数据库科学仪器、文件和超级计算系统等。使用计算网格，一方面能使人们聚集分散的计算能力，形成超级计算的能力，解决诸如虚拟核爆炸、新药研制、气象预报和环境等重大科学研究和技术应用领域的问题，另一方面能使人们共享广域网络中的异构资源，使各种资源得以充分利用。

过去人们往往很自然的把计算资源和特定有形的计算机联系起来，而网格就是在剥去了各种具体计算资源的外在的“形”的基础上，将其内在的“神” 即计算能力抽取出来，形成一种分布在网上的抽象的计算能力，在实现了“形”和“神”分离的同时，将原来有形的，专用的计算能力转化为一种无形的更为通用的计算能力，正如同电力网将具体的各种类型的发电机的电力转化为一种我们认为根本没有什么差别的统一的电力一样。

网格和电力网都有各自资源的消费者和资源提供者，对于电力网来说资源提供者就是发电站，对于网格来说资源提供者就是计算机等；对于电力网来说资源消费者就是各种消耗电能的设备，而对于网格来说资源消费者就是使用网格计算能力求解问题的用户。不论是电力网还是网格，它们都有覆盖范围广泛，而且组成资源多样的特点。正如同电力网中需要大量的变电站等设施对电网进行调控一样，网格中也需要大量的管理结点来维护网格正常运行。与电力网相比，网格的结构更复杂，需要解决的问题也更多，但是它也会带给我们更大的便利和帮助。

图 2−1 给出了电力网和网格组成的简单对比示意图[22]



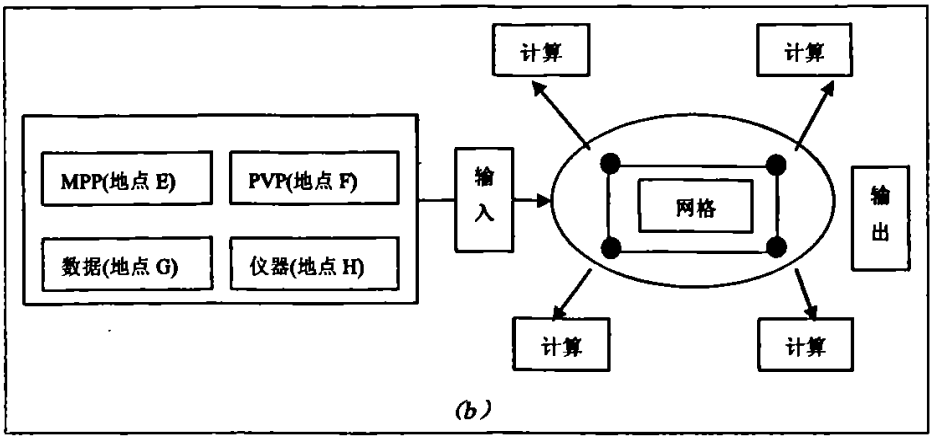


图 2—1 电力网与网格对比

(a) 电力网构成示意图 (b) 网格构成示意图

2.2.2 网格的研究现状

网格计算被誉为继 Internet 和 Web 之后的“第三代信息技术浪潮。”从美国、欧洲、日本等发达国家到印度等一些发展中国家，都启动了大型网格研究计划并得到产业界大力支持。英国政府已投资 1 亿英镑，研发“英国国家网格”，美国政府用于网格技术的基础研究经费达 5 亿美元。美国军方正在规划实施一巨型网格计划一“全球信息网格”，预计 2020 年完成。

我国的网格计算研究起步不久。由中国科学院牵头的“国家高性能计算环境(NHPCE，National High Performance Computing Environment)”项目和由清华大学牵头的“先进计算基础设施(ACI，Advanced Computational Infrastructure)，北京上海试点工程”两个网格计算项目已取得初步成果。由科技部支持创办的五个国家高性能计算中心已经运行。目前我国的网格计算研究主要集中于中科院计算所、国防科大、江南计算所、清华大学等几家在高性能计算方面有较强实力的研究单位。这些单位在高性能计算研究方面有很好的技术积累和很强的科研能力。

迄今为止，网格计算还没有正式的标准，但在核心技术上，相关机构与企业已达成一致由美国能源部及 NASA 等政府研发机构为中心推动，美国 Argonne 国家实验室与南加州大学信息科学学院(IS)合作开发的项目 Globus 协议已成为日前网格技术的典型代表和网格计算事实上的规范与标准。Globus Toolkit 作为自由软件已经在因特网上公开。包括 Entropia，IBM，Microsoft，Compaq 等在内的十二家计算机和软件厂商已宣布将采用 Globus Toolkit，作为一种开放架构和开放标准基础设施。Globus Toolkit 提供了构建网格应用所需的很多基本服务，如安全、资源发现、资源管理、数据访问等。目

前所有重大的网格项目都是基于 Globus Tookit 提供的协议与服务建设的。此外，包括 Global Grid Forum、对象管理组织(OMG)，W3C，以及 Globus，org 等标准化团体都参与了网格计算标准以及全球大网格(GGG)标准的制定工作。

2.3 网格的基本要求

对于网格提供的计算能力，有四个基本的要求，它们分别是可靠性要求，标准化要求，易访问性要求和价格低廉的要求[24]。

网格的可靠性是指网格提供的计算能力必须保证是持续、稳定和安全的，不应该因为网格内部个别资源的变化而对网格应用造成影响，即网格内部局部资源的变动对网格应用应该是透明的，就如同我们日常使用电灯的时候不应该因为个别发电厂临时出现什么故障而造成整个电网电力来弥补本地的电力不足，网格也应该能够保证提供持续、稳定的计算能力。网格还应该满足各种形式的安全要求，比如数据传输的加密，权限的认证，避免非法入侵和非法使用等，如果没有安全性保障，这种先进的计算服务就不能得到广泛的推广。

网格的标准化要求一方面是指网格资源之间应该有一个统一的可以相互访问的接口或者协议标准，因为只有这样才能够实现网格资源的互操作从而实现充分的资源共享，标准化是共享的前提；标准化的另一个含义是指网格对用户提供的计算能力应该满足一定的标准，有一种比较统一的形式，从而便于以一种统一的方式进行访问，对于访问者来说，不能因为时间、地点、具体的访问系统等的不同要求不断改变访问形式，访问形式应该有一致性，当然一致性的前提是网格必须提供给用户一个相对稳定的标准化接口。

网格的易访问性要求是指用户可以在任何时间，任何地点，以自己习惯的统一的形式访问和使用各种网格资源。网格计算能力可以通过网格输送到任何角落，随处可得。也就是说，在网格上是没有资源处在什么位置的概念的，只有“在网格上”或者“不在网格上”的区别，无论你在什么地方，网格资源都在你的身边。人们以前在解决特定问题时或许不得不到特定的地点来进行，比如到某一个单位去登记和使用特殊的仪器设备等，但是在网格解决问题上时，不应该因为访问者或者资源所在地位置不同而受到限制。

网格费用的低廉性要求是网格能够被普遍接受和推广的前提，不管网格有多少优点，如果大多数的使用者无法承受其费用，网格就不可能被普及，它的各种优势也就根本无法得到体现。网格技术通过将资源充分共享，最大限度发挥资源的使用价值，可以将原来闲置和浪费的资源收集起来供网格用户使用，而且可以避免以前由于地理位置闲置所带来的各种额外开销，显然网格对使用者存在着很大的见地开销的潜力。

这些要求都是网格需要去解决的问题，也是网格技术发挥作用的地方。网格作为一种新型而重要的基础设施，不是一夜之间就能够奇迹般地突然出现的，需要各个方面联合起来，共同努力才可以实现。

2.4 网格的特点

(1) 分布性：分布性是网格的一个最主要的特点。网格上的各类资源通常类型复杂、规模较大、跨越地理范围较广，在分布式计算环境下，需要解决资源与任务的分配和调度问题，安全传输与通信问题，实时性保障问题，人与系统以及人与人之间的交互问题等。

(2) 异构性。网格可以包含多种异构资源，包括跨越地理分布的多个管理域。构成网格计算系统的超级计算机有多种类型，不同类型的超级计算机在体系结构、操作系统及应用软件等多个层次上可能具有不同结构。

(3) 可扩展性：网格可以从最初包含少数的资源发展到具有成千上万资源的大网格。由此可能带来的一个问题是随着网格资源的增加而引起的性能下降以及网格延迟，网格必须能适应规模的变化。

(4) 共享性：网格的根本特征是资源共享而不是它的规模。尽管网格资源是分布的，但是它们却是可以充分共享的。分布是网格硬件在物理上的特征，而共享是网格软件支持下实现的逻辑上的特征。

(5) 可适应性：在网格中，具有很多资源，资源发生故障的概率很高。网格的资源管理或应用必须能动态适应这些情况，调用网格中可用的资源和服务来取得最好的性能。与一般的局域网系统和单机的结构不同，网格系统由于地域分布和系统的复杂使其整体结构经常发生变化，网格系统的应用必须能适应这种不可预测的结构。

(6) 结构的不可预测性：动态和不可预测的系统行为。在传统的高性能计算系统中，计算资源是独占的，因此系统的行为是可预测的。而在网格系统中，由资源的共享造成系统行为和系统性能经常变化。

(7) 多级管理域：由于构成网格系统的超级计算机等资源通常属于不同的机构或组织并且使用不同的安全机制，因此需要各个机构或组织共同参与解决多级管理域的问题。

2.5 网格的关键技术

任务的协同是网格计算的核心。要解决任务的协同，必须要解决好资源管理和任务管理。资源管理和任务管理的前提是通信和安全，所以网格计算系统除了需要仔细研究其体系结构之外，还要特别注意研究通信技术、安全机制、用户界面等。因此网格的关键技术包括了网格的体系结构、资源管理，任务管理、安全机制和通信技术等。

2.5.1 网格的体系结构

网格体系结构是关于如何建造网格的技术，包括对网格基本组成部分和各部分功能的定义和描述，网格各部分相互关系与集成方法的规定，网格有效运行机制的刻画。显然，网格体系结构是网格的骨架和灵魂，是网格最核心的技术，只有建立合理的网格体系结构，才能够设计和建造好网格，才能够使网格有效地发挥作用。目前网格体系结构的设计已有了一定的研究，提出的模型有：五层沙漏模型、组件模型、开放网格体系结构(OGSA) 模型、计算池模型、CPU 模型、神经网络模型、节点模型等。其中五层沙漏[25]是经典的模型(图 2—2)。这五层分别是：

(1) 构造层(Fabric) 。它的功能是向上提供网格中可供共享的资源，它们是物理或逻辑实体。

(2) 连接层(Connectivity) 。它是网格中网络事务处理通信与授权控制的核心协议。构造层提交的各种资源间的数据交换都在这一层的控制下实现，各资源间的授权验证、安全控制也在这里实现。

(3) 资源层(Resource) 。它的作用是对单个资源实施控制，与可用资源进行安全握手、对资源进行初始化、监测资源运行状况、统计与付费有关的资源使用数据。

(4) 汇集层(Collective) 。这层的作用是将资源层提交的受控资源汇集在一起，供虚拟组织的应用程序共享、调用。为了对来自应用的共享进行管理和控制，汇集层提供目录服务、资源分配、日程安排、资源代理、资源监测诊断、网格启动、负荷控制、账户管理等多种功能。

(5) 应用层(Applications) 。它是网格上用户的应用程序。应用程序通过各层 API 调用相应的服务，再通过服务调用网格上的资源来完成任务。

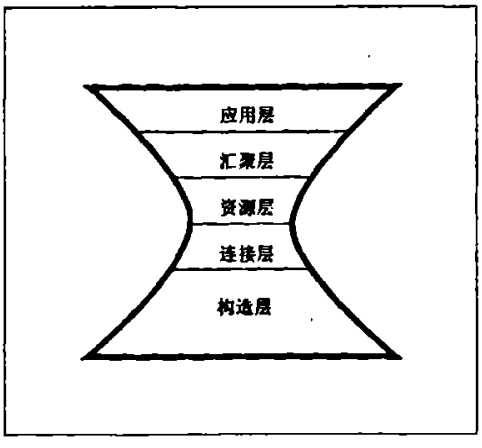


图 2—2 五层沙漏结构的协议分层

五层沙漏结构是一个抽象层次结构，它的一个重要特点就是构成一个“沙漏”形状，如图2－2所示。在沙漏结构中，资源层和连接层共同组成沙漏的瓶颈部分，为网格计算提供底层的通信、安全以及局部的资源管理。不同的高层(沙漏的顶部) 行为映射到它们的上面，它们自身也能被映射到不同的基本技术之上(沙漏的底部) ，显然瓶颈部分的核心协议的数量较少。较少的核心协议有利于移植，也比较容易地实现和得到支持。

2.5.2 资源管理和任务管理

由于网格资源纷繁复杂种类多、信息量大，如何有效地管理好网格中的各种资源就显得尤为重要。资源管理包括了资源的发现、描述、定位、组织(注册)、分配、监测、更新和信息发布等。要实现高性能计算和共享异构网格资源，必须提供统一的资源管理机制。统一资源管理模型负责用户与网格计算环境的交互，提供与网格计算系统的统一出入口。要组织网格计算的资源，通常使用单一映象文件系统来实现。单一映象文件系统将地理上分散的异构资源映象成一个单一入口的虚拟机器。目前，构造单一映象文件系统一般使用虚拟目录服务技术，将各种分散的地理资源映射到逻辑的文件系统[26]。目录服务是指一个存储着用于访问、管理或配置网络资源信息的特殊数据库，它把网络环境中的各种资源信息(计算资源：IP地址、可使用软件、系统管理者、连续的网络、操作系统名称和版本号、存储系统信息、系统负荷、进程信息、内存信息、任务队列等；网络资源信息：网络带宽、网络协议、网络延迟、网络的逻辑拓扑结构等；基础设施信息、主机信息、资源管理者等)都作为目录信息。在目录树结构中分层存储，对这些信息可以存储、访问、管理并使用，它采用动态可扩展的框架来管理网格计算环境中各种资源的静态和动态信息(资源信息、状态信息、优化信息)，保证了灵活性和动态性。

任务管理是网格计算研究必须解决的另一个关键问题。网格计算的目标是分解一个应用为几个任务(或子任务)并为每个任务匹配一个最适合执行的机器[26]。由于应用程序分解的任务之间往往包含优先约束关系，对这样的任务进行调度是必须重点考虑的问题之一。任务管理完成任务提交、查询、为任务指定所需资源、删除任务并监测任务的运行状态。任务调度的作用是根据当前系统的负载情况，对系统内的任务进行动态调度提高系统的运行效率，即按照用户提交的任务类到所需资源、可用资源等情况安排运行日程和策略。

本文所讨论研究的任务调度问题正是从资源管理和任务管理两个方面同时入手，将分布在网格中的资源进行协同，在保证任务调度的基础上，使得整个网格系统能够保持较高的系统资源利用率和系统负载平衡。

2.5.3 安全机制和通信技术

除了上述所说的网格的关键技术外，安全机制和通信技术也是网格计算中必须要解决的问题。所谓网格的安全机制是指两个方面的问题，一是身份验证的问题，另一个是网格计算必须要受到有序的控制和管理。而通信技术是实现网格计算系统安全可靠地进行资源动态整合、任务分布协同的保证。这两个技术不是本文所讨论的关键问题，因此在此就不在详述。我们所讨论的网格环境下的任务调度是假设网格系统安全和忽略通信延迟的。

2.6 本章小结

网格是一种关系科研、教育、经济、社会和国防的重要的国家网络基础设施，被称为下一代Web技术。网格计算系统将使成千上万的用户在大范围的网络上共享各种资源，其运作方式犹如输电网把电力送到消费者手中一样。本章介绍了网格计算的概念、特点、功能、起源、研究现状及其关键技术等问题。旨在使读者对网格计算有一个总体的概念，做初步了解。

第三章 P2P 技术概述

3.1 引言

网络技术的飞速发展与迅速普及使其成为数据通信的重要手段，网络规模越来越大，连入网络中的计算设备的数量和种类也越来越多，然而这些资源并没有得到充分利用，如果能将这些计算单元的处理器计算能力、磁盘存储能力、网络带宽资源等进行充分利用，将会有效缓解目前互联网所面临的一些问题。P2P(Peer-to-Peer对等网络)，就是在这种背景下提出的一种网络技术。

在这一章之所以要介绍P2P技术，是为本文所讨论建立的P2P_Grid模型提供理论基础。在本章中，将分别从P2P产生的技术背景、P2P的概念、技术特点和关键技术等几个方面来展开阐述。

3.2 技术背景

P2P(Peer-to-Peer 即对等网络)起源于最初的联网通信方式，它并不是一种新的技术，如果回顾一下，我们就不难发现在WWW出现伊始，P2P就是互联网的本质特征之一。现在互联网是以S(Server)/B(Browser)或S/C(Client)结构的应用模式为主的。首开P2P之风的最有名的计划是由柏克莱大学展开的寻找外星生命的SETI@home研究计划。1999年，SETI@home开始用P2P来分析行星的无线电信号，寻找宇宙中可能存在的其他外星文明的证据。P2P技术串联所有参与研究计划者闲置的电脑来执行庞大复杂的运算，然后再把结果传到SETI@home总部。目前，全球共有240多万人为SETI@home贡献出了其闲置的电脑处理能力，这些电脑每天平均发挥的效能超过了全球造价最高的，运算速度最快的超级电脑。

追根溯源，P2P技术计算方法来源于局域网文件共享，该技术在20世纪70年代中期就已经很流行了。许多用户游戏就是在P2P模式下运作的。美国Intel公司在十年前已经开始使用P2P技术了。图3－1给出了美国Intel公司的NetBatch系统中P2P技术的应用。从某种意义上讲，P2P体现了Internet的本质。在P2P技术的推动下，因特网的存储模式将由现在的“内容位于中心”模式转变为“内容位于边缘”模式。从这个角度看，P2P带来了几个改变：

首先，客户不再需要将文件上载到服务器，而只需要使用P2P技术将共享信息发布出去；

其次，运行P2P的个人电脑不再需要固定IP地址和永久的因特网连接。用户可以随时随地地加入到P2P网络中，其活动不再受固定IP地址的限制。用户的身份也和IP地址

没有任何关系。这使得那些通过动态IP地址连接因特网的用户(如拨号上网的用户)也可以享受P2P带来的变革，而这部分用户在因特网用户总数中占有极大的比重；

最后，P2P完全改变过去控制因特网的客户机/服务器(Client/Server)模式，消除客户机和服务器二者之间的差别。P2P网络中的任何一个对等点既是客户机又是服务器。

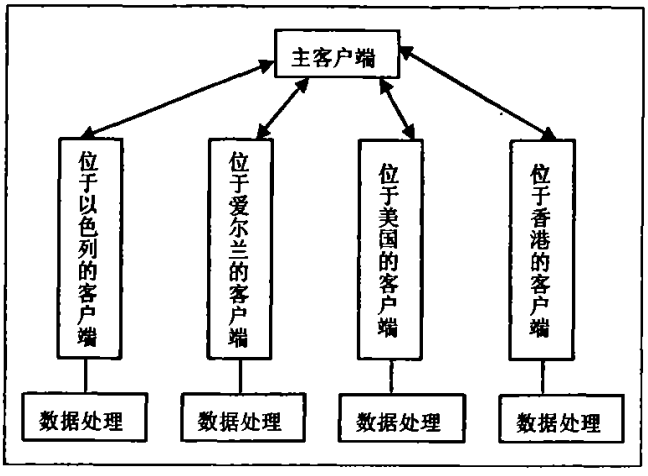


图 3－1 Intel 公司 NetBatch 系统中 P2P 技术的应用

3.3 P2P 的概念和技术特性

3.3.1 P2P 的概念

目前对P2P还没有一个统一的标准，P2P是Peer-to-Peer 的缩写，Peer 在英语里有“(地位、能力等)同等者”、“同事”和“伙伴”等意义。这样一来，P2P 也就可以理解为“伙伴对伙伴”的意思，或称为对等联网。这种对等网络模型中，所有节点都是对等的，各节点具有相同的责任和能力，并协同完成任务。对等点之间直接互连，共享信息资源、处理器资源、存储资源甚至高速缓存资源，无须依赖集中式服务器或资源就可完成。它的节点具有很高的自治性和随意性。它是为区别客户器/服务器模型而提出的。在P2P 模型中，每个节点既可以是客户机也可以是服务器。节点之间可以直接进行信息的共享和交换。如图3－2、图3－3 所示。其中，图3－2 为客户服务器模型，图3－3 是P2P 模型。

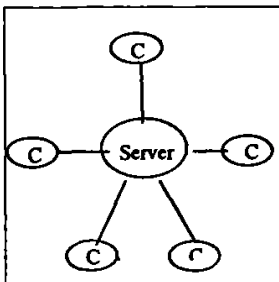


图 3－2 C/S 模型
C 代表 Client

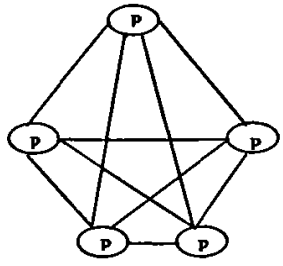


图 3－3 P2P 模型
P 代表 Peer

与传统的C/S模型相比，P2P在网络资源利用率、消除服务器瓶颈等多方面有明显的优势[27]。在P2P模型中，每一个对等点具有相同的地位。既可以请求服务也可以提供服务。同时扮演着C/S 模式中的服务器和客户端两个角色。还可以具有路由器和高速缓冲存储器的功能，从而弱化了服务器的功能，甚至取消了服务器。而且P2P技术可以使得非互连网络用户很容易地加入到系统中。每一个对等体可以充分利用网络上其他对等体的信息资源“处理器周期”高速缓存和磁盘空间；对等体通常没有固定的IP地址，并且可常常从网络上断开，信息的存储及发布具有随意性，缺乏集中管理。P2P 是基于内容的寻址方式[28]。内容包括信息、空闲机时、存储空间等。P2P网络中用户直接输入要索取的信息的内容，而不是信息的地址。P2P软件将会把用户的请求翻译成包含此信息的节点的实际地址，这个地址对用户来说是透明的。

但是P2P也有不足之处。首先，P2P不易于管理，而对C/S网络只需在中心点进行管理。随之而来的是P2P网络中数据的安全性难于保证。因此，在安全策略、备份策略等方面P2P的实现要复杂一些。 另外，由于对等点可以随意地加入或退出网络，会造成网络带宽和信息存在的不稳定。P2P技术与C/S技术性能比较如表3.1

表3. 1 P2P技术与C/S技术性能比较

性能比较	数据发布	数据接收	数据互动性	数据及时性	数据安全性	数据更新	数据质量	覆盖率数据	成本控制	管理方便性
P2P	好	中	好	好	差	好	中	差	好	差
C/S	差	好	差	差	好	差	好	好	差	好

3.3.2 P2P 的技术特性

(1) 既是S(Server)又是C(Client)，如何表现取决于用户的要求，网络应用由使用者自由驱动；
(2) 信息在网络设备间直接流动，高速及时，降低中转服务的成本；
(3) 构成网络设备互动的基础和应用；
(4) 在使网络信息分散化的同时，相同特性的P2P设备可以构成存在于互联网这张大网中的子网，使信息按新方式又一次集中。

3.4 P2P 的关键技术和研究现状

3.4.1 P2P 的关键技术

这里我们主要讨论的为P2P的软件技术。
(1) 对于互联网上众多计算机，P2P 应用应更多考虑那些低端PC的互联，它们不具备服务器那样强的联网能力，且现在的硬件环境已更为复杂， 这样P2P必须提供在现有硬件逻辑和底层通信协议上的端到端定位(寻址)和握手技术，建立稳定的连接。涉及的技术有IP地址解析、NAT路由及防火墙。
(2) 在应用层面上，如果对等点已通过互联网建立连接，那么一方的信息就必须为另一方所识别。所以当前互联网上关于数据描述和交换的协议，如XML、SOAP、UDDI等都是一个完善的P2P软件所要考虑的；
(3) 如何设置中心目录服务器、控制网络规模和保证网络的鲁棒性等；为保障信息的安全，必须要考虑加密技术；
(4) P2P网络的强动态性决定了难以实现对等点间的相互快速准确定位，即发现对方。目前的 主要发现机制有：基于本地缓存的发现、通过集合点或不通过集合点的本地发现、穿越防火墙的发现、基于集中点的传布方式发现等；
(5) 高效的搜索策略。一是应尽量减少搜索时经过的节点数，以直接减少请求消息和处理时间；二是数据应倾向流向对其感兴趣的区域，即数据活跃区。 快速搜索感兴趣的资料，与其他的对等点共享资源和服务。

3.4.2 P2P 的研究现状

由于P2P模式所具有的技术特点，很多计算机公司、研究部门都认为该技术蕴含着巨大的商业和技术潜在价值，并从不同的角度应用和研究该技术。目前主要的研究角度有：文件交换、对等计算、协同工作、即时通讯、搜索引擎、网络游戏、基于Internet的文件存储系统、基于Internet的操作系统等。另外，还有对P2P开发平台的研究和P2P

安全框架的构建等。

文件交换是P2P最初的应用和基本功能之一，可以说文件交换的需求直接引发了P2P技术热潮。Napster抓住人希望通过互联网共享MP3音乐文件的需求，以P2P模式实现了自由的文件交换体系，从而引发了网络的P2P技术的革命。通过P2P来搜索和下载与传统的方式最大的区别就是你不是从其它站点的服务器搜索与下载资源，而是从任何一个在线网友的机器里直接下载，当然其它网站的服务器也可以看作是一个对等点，这样真正实现了与服务器平起平坐。

P2P用于对等计算的优势在于每个对等点不再只是单纯的接收计算任务，它还可以根据自己的情况(如分到的任务太多)再搜索其他空闲节点把受到的任务分发下去。然后中间结果层层上传，最后到达任务分发节点。对等点之间还可以直接交换中间结果，协作计算。按照这种方式进行，可以合力整合闲散的计算能力和资源，使得总体计算能力得到大规模提升，获得非常可观的计算性能/价格比。

协同工作是指多个用户之间利用网络中的协同计算平台互相协同来共同完成计算任务，共享各种各样的信息资源等。协同工作使得在不同地点的参与者都可以在仪器工作。而P2P技术使得互联网上任意两台PC都可建立直接的通讯联系，不再需要中心服务器，降低了对服务器存储以及性能的要求，也降低了对网络吞吐量和快速反应的要求，从而大大节约了成本，使低成本的协同工作成为可能，最终帮助企业和关键客户，以及合作伙伴之间建立起一种安全的网上工作联系方式。因此基于P2P技术的协同工作目前受到了极大的重视。

及时通讯(如OICQ、ICQ等被称为在线聊天的软件)应用将超过文件共享应用，称为P2P的第一大应用。

搜索引擎是目前人们在网络中搜索信息的重要工具，目前的搜索引擎如：Google、天网等都是集中式的搜索引擎。P2P网络模式中节点之间的动态而又对等的互联使得搜索可以在对等点之间直接地、实时地进行，既可以保证搜索的实时性，又可以达到目前传统目录式搜索引擎无可比拟的深度。可以说，P2P为互联网的信息搜索提供了全新的解决之道。

以上我们列出了P2P技术的部分发现的现状，对于其他的P2P技术也同样得到了广泛和深入的研究，并出现了一些成果和产品，体现出巨大的商业和技术上的发展潜力，因此也吸引了更多大型公司、著名高校和研究机构的加入，P2P技术正在成为计算机领域的一个重要的热点课题。

3.5 P2P 与网格的互补性

前面我们介绍了P2P技术的产生背景、概念、关键技术和研究现状等一系列相关问题，这一节，将引入本课题所研究的一个重点问题：P2P与网格的互补性讨论。这个讨论对我们所研究的课题极为重要，有了这样的理论基础，我们才能使课题的研究成为可能。因此，我们将从以下几个方面来比较P2P与网格的异同。

3.5.1 网格计算与 P2P 的比较

网格计算和P2P都是分布式计算的模型[29]，将计算机集成到网络中以进行分散协作，二者相比，有许多相同之处[30]，比如：

(1) 最终目标都是协调使用大规模分布式资源；

(2) 均研究在虚拟组织内资源的动态共享问题；

(3) 均采用层叠网络结构方法来解决问题；

但二者在具体的细节上还有明显的区别。网格目前主要面向复杂的服务和应用，协同处理复杂科学等方面；而P2P主要用在文件传输、即时通讯等方面，现在正逐步扩展到复杂应用系统的开发，同时在大规模自治系统的管理上有许多创新，如协作文件系统CFS[31]、大规模可靠的存储架构PAST[32]、全球范围内的数据存储基础架构OceanStore[33]等。

目前的网格为中等大小的团体提供服务，强调基本资源的集成是在一个一定限度可信的环境中来提供非平凡服务。而P2P是针对众多参加者，提供有限的专业服务，很少关注服务质量(QOS)，并且很少考虑信任关系。

3.5.1.1 目标

网格计算和P2P在满足用户要求的具体目标不完全相同。

网格计算的目标[34, 35]：协调处于非集中控制的共享资源，汇聚各类资源，提供非平凡服务，解决对于任何单个超级计算机来说很难解决的问题。网格是利用Internet边界和中间的共享存储与计算，实现自配置、自调节、自愈。

P2P的目标：资源共享、汇聚节点上各类资源，将代价或成本分摊到参与的所有节点共同承担，具有可扩展性、自治性、动态性等，支持匿名，甚至支持Ad Hoc通讯等。

3.5.1.2 资源发现与管理

网格环境中的资源发现主要是基于集中或分层模型进行的。例如：在Globus 工具包中，通过查询运行于特定结点上的服务器应用程序，或者查询运行于特定结点上的信息检索与发布程序，用户或应用程序能够直接获取有关一个结点的资源信息。因为构建这些网格系统是用来满足某一组织的固定需求的，这些组织中的网格无需支持更动态、范围更大的分布式环境。在那些动态而大范围的分布式环境中，查询数量的庞大会使客

户机/ 服务器方法很快失效。资源发现包括发现网格中的空闲结点，当然也包括发现一些管理问题，目前在网格中还没有定义资源发现的全局性机制。

而在P2P系统中，资源管理协议比较成熟，它采用的是主动报告策略，每一个结点周期性地向网络中的其它结点报告它的资源现状，同时发现其邻居结点的相应信息。随着网格的发展，应该采用P2P系统中的分散资源发现模型，以提高网格模型的开放性。

3.5.1.3 连通性

网格中一般都包括了具有很强计算能力的机器，它们通过高可靠性的高性能网络静态地连接在一起，同时，网格中可访问的结点数一般都较少，因为对于网格资源的访问是与严格的帐户机制结合在一起的，这类似于客户机/服务器计算模式。

P2P系统主要是由桌面计算机所组成，它们是间断性地连入计算机网络，连入网络时，可为其它计算机所共享，但随时可能会断开连接，因此其可靠性不稳定。由于P2P网络中连接的结点数远大于网格中的结点数，就新结点的接入、用户访问及账户机制而言，网格中的连接方法太繁琐、显得呆板，因此，网格计算可以从P2P更灵活的连接模型中受益。

3.5.1.4 访问服务

在建设和使用网格时，对远程资源进行访问是其主要的创新。对于在远程机器上提交批量任务或进行交互性的应用而言，网格工具提供了相应的安全服务，这些机制为在结点间有效地共享和移动数据提供了保障。

当前实现的P2P系统虽然不支持在远程机器中进行复杂的运算和数据存储机制，但是它在结点间提供了高效的数据共享和交换协议，研究将P2P任务提交模型、P2P任务调度方法应用到网格的任务调度和管理中来是一个非常有意义的事情。

3.5.1.5 安全性

安全无疑是非常重要的问题。网格计算起源于科学领域，面向的是专业团体。因此，网格计算具有更强的安全要求，如认证、授权、数据的完整性和机密性等等。网格计算不允许匿名用户或资源。在网格平台中，为了对用户进行认证、授权，并考虑到信息的完整性和保密性，已经在整合相应的机制方面做了一些努力。

P2P系统则不同，它具有开放性，用户可以共享范围更广泛的资源(比如：从整个Internet 中检索音乐)，而不仅是在具体、确定的目标中进行工作(如：参加高能物理模拟)。因此，在大多数广泛分布的P2P系统中，安全机制中一般都不进行认证和内容确认，相反地却提供确保匿名和应对信息审查的协议，使系统更具有开放性。因此，很多用户对于P2P系统的安全性很担心，安全性是P2P中的一个重要因素，P2P和网格这两个模型在处理安全性问题上的不同，使得研究如何整合它们中已有的方法来创建一个安全的P2P网格模型是很有意义的工作。

通过以上分析可以看出，P2P与网格都是分布式计算模型，它们都是分布式计算的子集。而且可以发现网格从本质上来说是一个P2P系统，它们在广域网中实现异构机器间的合作、动态监视和管理网格资源并适应资源规模的动态变化等方面都有着很多相似

之处。P2P与网格两者都具有相同的目标，都是为高性能计算、数据密集型应用等提供远程计算资源的访问能力。尽管网格中的许多方面是分层服务的，这种服务模式在以后的研究与实现中应该会得到修改或者是被舍弃掉。随着复杂应用中的网格结点数从数十到数千的增长，对于结点的功能应该进行分散以避免性能颈瓶的出现，这就为将P2P模型中的一些技术引入到网格中来创造了条件，这些技术融合到网格系统中以后，有助于确保网格的可扩展性，因为能够使用P2P的思想和技术来实现非分层的分散网格系统。

P2P系统与网格计算均是分布式计算的子集，作为新型的分布式系统，网格与P2P模型有不少相同的特征，充分认识这些共性，有利于这两种技术的进步和互补。

3.5.2 P2P_Grid 模型

经过以上分析，提出本课题所研究的调度模型－P2P_Grid模型，之所以建立这样的一个模型，是因为：

(1) P2P和网格都是一种新型的分布式系统，它们都可以视为分布式系统的子集；

(2) 网格本质来说就是一个P2P系统，它们在广域网中实现异构机器间的合作、动态监视和管理网格资源并适应资源规模的动态变化等方面都有着很多相似之处；

(3) P2P与网格两者都具有相同的目标，都是为高性能计算、数据密集型应用等提供远程计算资源的访问能力；

(4) 尽管网格中的许多方面是分层服务的，这种服务模式在以后的研究与实现中应该会得到修改或者是被舍弃掉。随着复杂应用中的网格结点数从数十到数千的增长，对于结点的功能应该进行分散以避免性能颈瓶的出现，这就为将P2P模型中的一些技术引入到网格中来创造了条件，这些技术融合到网格系统中以后，有助于确保网格的可扩展性，因为能够使用P2P的思想和技术来实现非分层的分散网格系统。

以上的四点，为我们建立新的资源管理模型提供了可靠的理论基础，利用P2P和网格的许多相似点，采用P2P协议和模型来处理网格计算，比如像可扩展性、连通性以及资源发现等，利用P2P与网格技术之间的协同和互补，能够构建高性能的分布式系统。

3.6 本章小结

P2P与网格计算都是新型的分布式计算模型，它们的总体目标相似。本章首先介绍了P2P的概念和技术特点，对网格计算和P2P中的目标、资源发现与管理、连通性、安全性、访问服务等几个重要问题进行分析。网格计算和P2P的这些异同点为将P2P模型中的技术应用到网格中来创造了条件，同时也为课题的研究奠定了一定的理论基础。

第四章 基于网格环境下资源调度模型的建立

4.1 引言

网格环境是异构环境，是新型的分布式系统。研究网格环境下的任务调度有着重大的意义。目前，在这一领域里已经存在一些静态的启发式算法，比如经典的 Min_min 算法[36]，遗传算法[37]等；而对于动态的算法，几个成熟的技术已经被报告[38]。在网格计算环境中，由于各处理器运行在不同的速度而带来的负载平衡的困难，并且难以设计和实现，在异构平台上，任务调度策略已经遇到了很多问题。

4.2 网格任务调度的特点和主要目标

4.2.1 网格任务调度的特点

网格计算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。由于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、数据库或其他设备。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) 任务调度是大规模的、非集中式的。由于网格系统是一个大到整个Internet的分布式巨系统。要实现一种全局的统一集中的任务调度管理是根本不可能的。因此，网格的任务调度必须以分布、并行方式进行任务的管理与调度。

(3) 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。

(4) 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。

(5) 任务调度能够动态自适应。网格中的资源不但是异构的而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。总之网格的动态性是明显的，所以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。

4.2.2 网格任务调度的主要目标

简单地说，网格计算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。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，负载平衡是一个关键问题。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。

4.3 几种经典的任务调度算法

本节中将对那些在异构的计算系统中存在的启发式的调度算法进行回顾并做出比较。在异构的计算系统中已经存在了大量的启发式的静态调度算法，它们分别从各自的侧重角度完成任务的调度，但也存在许多缺陷，本节将对如下列举的 12 种基本的启发式调度算法给出详细分析和比较。

User-Directed Assignment (UDA)：该算法在对任务进行调度时，总是把任务指派给执行该任务时预期执行时间(expected execution time)最小的主机[39]。

Opportunistic Load Balancing (OLB)：该算法是将每个任务以随机的顺序指派到下一个可用的主机[39]。

Genetic algorithm(GA)：遗传算法是适合于全局解空间的搜索。一个标准的遗传算法有三个基本成分：①种群。也称为个体群。一个个体代表问题解空间的一个元素，遗传算法就是要在给定的时间里搜索到或进化出一个可以接受的解。②适应度函数。用适应度函数来评价问题解的好坏，它不受连续可微的约束，且定义域可以是任何函数。③遗传操作。包括选择、交叉和变异。

Simulated Annealing(SA)：模拟退火算法使用的是迭代的技术，对某个时刻所调度的任务仅考虑一种可能的解决方案。SA 使用这样的程序，允许解决方案试图获取一种基于系统温度的对解决方案空间的更优搜索[39]。

Genetic Simulated Annealing (GSA)：遗传模拟退火算法是混合了 GA 和 SA 两种技术[40]。

Ant Algorithms(AA)：基于蚂蚁算法的任务调度算法。蚂蚁算法是一个新的启发算法，它是基于蚂蚁的行为。蚂蚁算法固有的并发性和可扩充性，使它非常适合用于网格计算的任务调度。在同一时间内，所有影响资源状态的因素都能由一个事情，即信息素描述。那么调度程序就能非常简单、快速地获得预测结果。

Fast Greedy ：该算法在对任务进行调度时，总是把任务指派给执行该任务时完成时间最小(minimum completion time)的主机[39]。

Min-min ：在该算法中，将计算每个待调度的任务在所有主机的最小完成时间，从这个集合中再选择完成时间最小的任务，将其分配给相应的主机。将这个任务从等待队列中移除，重复这个过程直到所有的任务调度完成。

Max-min ：Max-min 算法和 Min-min 算法很类似，区别在于在 Max-min 算法中从所计算的最小完成时间集合中选择完成时间最大的任务，将其分配个相应的主机。

Greedy ：贪心算法是将 Min-min 和 Max-min 算法混合在一起从而得到更好的解决方案[39]。

Tabu ：该算法是对确切的解空间的区域进行搜索，这样就可以避免重复的搜索已知确切区域的临近区域[41]。

A* ：该算法是从通常为空解的根节点开始搜索，随着树的成长，中间节点给出了局部解，而叶子节点给出最终解。每个节点都有自己的开销函数，那些拥有最小开销函数的节点会被它的孩子节点所取代。任何时刻，当需要添加新的节点时，必须要删掉开销最大的节点。这个过程会反复执行直到所有的叶子节点都得到调度[42]。

试验结果表明，OLB，UDA，Max-min，SA，GSA 这几种调度算法并不能产生很好的调度结果，而 Min-min，GA 及 A*则表现出较好的性能。这些算法的调度时间(Makespan)的差异在 10%左右。GA 的性能总是比 Min-min 算法要好一些，而 A*算法在不同的情况下与 GA 和 Min-min 算法相比较会呈现时好时坏的不稳定的状态。比较 GA， Min-min， A*三种算法，Min-min 的执行时间最快，GA 次之，A*最慢。对于 512 个任务，16 台主机这样的条件下，Min-min 的运行时间大约为 1s，GA 约为 30s，而 A*约为 1200s。

通过以上分析可以看出，每一种任务调度算法都有其针对性，适用于不同的实际问题，但是同时我们也发现，尽管是有针对性的提出这样或那样的调度算法，它们仍然是有缺陷的，都是有一定的改进空间的。比如 Min_min 算法的思想是每次待调度的任务都是完成时间最小的，这样的话，自然可以保证小任务的及时完成，但是对于大型的调度任务却总是搁浅，延长了任务调度的完成时间；而对于 Max_min 算法来说，它总是

从完成时间最小的任务集合中选择最大的，这样的调度方法不仅不能保证小任务的完成，同样不能保证大型任务的完成，也同样会造成整个任务调度时间的延长。而对于待调度任务的用户来说，他们总是期望自己提交的任务能够在最短的时间内得到调度；对于系统来说，我们总是期望任务调度的总时间最小，同时系统的吞吐量尽量大，资源负载可以保持平衡等。总之，对于网格环境下的任务调度算法，不论是哪一种调度算法，我们总是期望用户(或者说待调度任务)和系统之间取得一个相对较好的平衡，本文所讨论的调度算法同样是希望在保证任务调度的基础上，对于系统资源利用率和系统负载平衡能够取得一个较好的效果。

通过以上分析，可以看出目前网格任务调度算法所面临的问题是：

(1) 设计最优化(也就是最小执行时间)调度算法是一个完全NP问题。例如：调度带有单位通信延迟的单位长度的任务到不定的资源的核心问题是一个完全NP问题。最优化的方法依靠拇指原则，诸如评价关键路径的问题。

(2) 准确评价任务执行时间和通信延迟是困难的，例如在编译期估计通信延是不可行的，因为运行期网络争用延迟。

(3) 静态方法不能解决处理器速度的多样性，由于它们的负载可变。

4.4 资源调度模型

对于分布式系统下的任务调度算法来说，研究已经证明它是一个 NP 完全问题，甚至是一个 NP 难解问题，也就是说在现有的条件下，不可能找到一种最优的解决方案来完成任务的调度；然而幸运的是，可以找到次优解来解决一些实际的问题。本文所讨论的任务调度算法是基于分布式系统的，只是我们在现有的分布式系统，即网格系统中加入了 P2P 技术，产生了一种新的资源管理模型－P2P_Grid 模型。在这个新的分布式的模型的基础上来讨论本文所研究的问题。

4.4.1 现有的资源调度模式

在有众多资源和众多参与者及协调者的情况下，资源调度是一个极其繁琐复杂的问题。资源调度的基本原则是保障资源为完成尽可能的任务服务，不能出现死锁是基本的要求，还要考虑资源在时间和空间上的搭配；目的是为了完成用户提交的任务和满足用户提出的要求，把网格中所有可用资源、计算资源、存储资源和网络资源进行匹配，找到最好最合理的资源分配方式和资源调度策略；目标是各个主机能够得到适合自己的任务，并且几乎能够同时完成任务。

现有的调度策略有两种，集中式调度和分布式调度：

(1)集中式调度：在网格中只有一个调度中心，负责调度网格中的所有资源。

其优点是：调度系统知道网格中的所有资源，对于一个应用可以高效地产生

资源调度方案；

其缺点是：当网络比较大时，调度系统很难掌握所有的资源。调度系统会成为瓶颈。比如由于错误使得调度系统出现故障就会影响整个网格系统；

(2)分布式调度：在网格中有多个调度中心，各个中心是平等的。

其优点是：健壮性、可靠性和可用性比较高；

其缺点是：调度中心之间的通信量比较大。由于不能掌握网格中的所有资源，所以很难找到全局最优的资源分配方式。

从上面的分析可以看出，调度中心在整个调度策略中的地位非常重要，它负责发送任务数据给计算主机并接收结果；收集各个主机的负载信息并据此调度任务，保持整个系统的负载平衡。 所以当网格规模较小时，集中式调度效果较好；而当网格规模很大时，调度中心的负载很重，比较适合分布式调度。

4.4.2 新的资源组织管理模型

4.4.2.1 P2P_Grid 模型的研究背景

Internet 环境下的各种资源分布极不均衡，网吧、公司、政府、教育部门、科研机构所拥有的资源一般是通过局域网接入 Internet，它们彼此间的距离比较近，单位面积范围内的资源密度比较大，彼此之间的通信速度比较快。居民区里的计算机分布密度远远不如上述的那些部门。不同的方式接入互联网，网络速度也不一样，局域网内部的通信速度明显大于通过互联网的通信速度。

目前构造网格系统所用到的主要资源管理体系结构模型有三种：层次模型、抽象所有者模型和计算经济模型。各个城市之间若使用分层模型或抽象所有者模型构造网格系统，由于通信速度较低，加上长距离的影响，必将对资源的查找和任务的提交所需花费的时间产生很大影响。

针对以上情况，提出了 P2P_Grid 模型。每个超级 Peer 相当于一个控制中心，里面存有很多的资源，实际上是构造 P2P_Grid 系统的一个局部网格系统。超级 Peer 也应该根据 P2P_Grid 环境下的资源分布情况，尽可能地在资源密集的地点建立超级 Peer 这样用户提交的任务绝大多数情况下是提交给其所在超级 Peer 处理，从而大大缩短通信距离，进一步缩小通信延迟。

4.4.2.2 P2P_Grid 模型的理论依据

这一节将介绍本文所采用的模型，即 P2P_Grid 模型。通过前面的比较分析，可以得到这样的结论：在网格与 P2P 的发展中，网格是一个终极目标，它融合了分布式系统的许多技术，力图实现对目前网络上所有能共享资源的访问，而 P2P 则是为了使得网络上的所有计算通信实体间能实现平等互利的通信而构建的，结点间可以相互访问和共享。因此，P2P 计算克服了集中式计算中对功能强大的网络计算机的需求以及昂贵的带

宽开销的缺陷，从而提高了系统的效率。

网格的特点及其最终面向最普通的计算机用户的目标，决定了网格系统不宜采用传统的集中式资源管理模式。普通计算机位于互联网的边缘，且 P2P 技术处理这类资源已经比较成熟。为把这类数量庞大的资源整合到网格系统中，结合 P2P 和 Grid 的互补性，可采用 P2P_Grid 模型组织网络环境下的资源。该 P2P_Grid 模型依据银行系统的运作模式，采用“分而治之”的思想，把“单一网格系统”分割成若干处于对等地位的“小规模网格”系统，本文称为超级 Peer。每个超级 Peer 可使用不同网格技术管理所在域的资源，就如同 P2P 系统中计算机的操作系统可以互不相同。

P2P 系统与网格计算均是分布式计算的子集，作为新型的分布式系统，网格与 P2P 模型有不少相同的特征，这样的话，我们就能够充分利用 P2P 与网格网络之间的许多相似点，采用 P2P 协议和模型来处理网格计算。在网格中融入 P2P 技术，即在网格中加入若干的超级 Peer，使得每个中心超级 Peer 构成一个局部网格系统，负责局部任务的处理。而各个局部网格系统通过 P2P 技术互联。在这个模型中，对于一组客户机而言，每个超级 Peer 作为一台服务器来使用，而超级 Peer 之间是对等的。这种拓扑在集中式搜索与匿名性的效率之间实现了平衡，并保持了系统的负载平衡和分布式搜索的鲁棒性。在基于超级 Peer 模型的网格信息服务中，每一个参与的组织都会配置一个或多个结点，并作为超级 Peer 来操作。在每一个组织内的结点将会与中心超级节点交换监视和资源发现信息，而对于不同组织中的超级节点将会以 P2P 的方式来交换信息。同时 P2P_Grid 模型还具有不依赖集中控制、分布性、可扩充性、能适应资源状态变化等特征。

4.4.3 P2P_Grid 内部资源调度模型

前面已经说明 P2P_Grid 模型就是让每个超级 Peer 掌管一个网格子系统，负责处理其 IP 地址范围内的事务。超级 Peer 根据 Internet 环境下的资源分布情况，尽可能地在资源密集的地点建立超级 Peer，这样用户提交的任务绝大多数情况下是提交给其所在超级 Peer 处理，从而大大缩短通信距离，进一步缩小通信延迟。这样整个任务的调度时间也会大大降低。图 4－1 给出了一个超级 Peer 的局部资源管理模型。

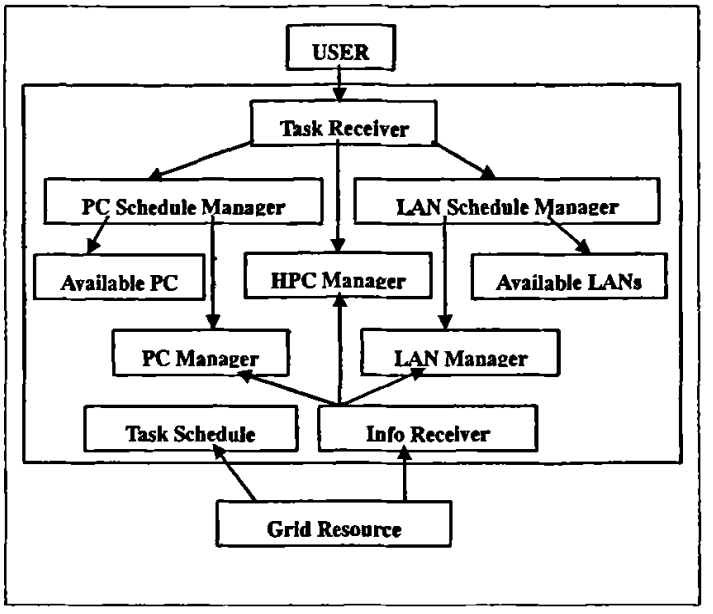


图 4—1 超级 Peer 的局部资源管理模型

图 4—1 中的网格资源(Grid Resource)通过信息接收器(Info Receiver)提交资源属性信息请求对资源进行注册。信息接收器将所接收的资源进行分类分别把信息提交给 PC 管理器(PC Manager)，HPC 管理器(High Performance Computer)和 LAN 管理器(LAN Manager)。用户向任务接收器(Task Receiver)提交任务，任务接收器接到任务后响应该任务的资源请求，并向资源管理器提交查找资源的请求。资源管理器通过一定的查找算法搜索注册的可用逻辑资源，并把这些可用资源的属性信息按一定的策略和级别高低顺序记录在可用的资源管理器中。同时选择最佳的资源或资源集合分配给所提交的任务。

资源接收器还要及时掌握系统中的负载情况，一旦发现所在的局部系统的负载过重，就马上将任务转交给其它空闲的超级 Peer 处理，这样就克服了传统算法中负载不平衡的缺陷。

网格任务调度实际上可分两个部分，即资源调度和任务调度。待调度的任务总是在其所需的资源满足的条件下，才予以调度；当现有资源不能满足待调度任务时，调度任务进入等待队列。而对于分布式系统下的资源自然也是分布的，这就需要我们不得不考虑这样一个问题，即分布系统下的资源协同问题，最充分的利用系统中的资源是任务调度始终非常关注的问题。但是传统的调度算法都是将两种调度分开进行讨论，这样势必会忽略掉某些实际存在的参数。例如在任务调度算法中会忽略调度某些资源的时间，而资源调度中会忽略任务的某些属性，例如任务所需的 QOS 的请求等。

本课题所提出的 P2P_Grid 模型，正是从资源协同或者说资源调度的角度出发来考虑问题的。该模型提出了新的对资源组织管理的模型。由于 P2P 技术的引入，我们使用超级 Peer 将整个网格系统划分为小规模的子网格，每个超级 Peer 是其所在的子网格的

中心，负责内部的任务调度；而对于加入的每个超级 Peer，它们是对等的，可以进行资源的协同和任务协同，当某一个超级 Peer 不能完成它所负责的任务调度时，可以转交给其他的超级 Peer，共同完成任务的调度。那么每个超级 Peer 对于任务的调度可以分为下述三种情况：

(1) 若任务管理器中某类任务的队列存储器中没有其他的任务正在执行或等待调度，当任务管理器接收到用户提交的第一个该类任务后，向资源管理器查找能够满足这一要求的资源。资源管理器根据所接收到的资源的请求查找相应的资源组中是否有可用的资源队列。若资源队列为空，则检索资源组目录树的子树或叶子节点，然后向任务管理器及资源监控器返回所收集到的信息。任务接收器接收到信息后，根据资源的 IP 地址范围对资源按 IP 地址的范围分组组合，并保存所有这些资源组合的信息。资源监控器接收到这些资源的信息后，搜索数据库中这些资源的最新状态信息并返回给任务管理器。任务管理器根据前后两次接收到的资源状态信息，对资源的调度作最后的部署，根据组合优化原理选出一组整体性能最好的资源分配给该任务。

(2) 若用户提交的任务所属队列的存储器中还有其他的任务或有这类任务正在执行，而任务管理器中相应的任务队列为空，并且资源管理器中相应资源组中可用资源队列不为空。由于第一次接受这类任务时，资源管理器已经保存了比较好的 QOS 完成该任务的资源集，即还有已被调度的等待分配的资源队列，则没有必要再向资源管理器提交查找资源的请求。直接在已被调度的资源队列中选取还没有被分配的资源组合，并把这些资源的属性信息传输给资源监控器。资源监控器根据 IP 访问资源，并立即对资源进行实时定位，之后同样要返回资源的最新状态信息。资源管理器分析这些信息后，根据一定的组合优化策略，选出一组资源分配给该任务。

(3) 若用户提交的任务所属队列的存储器中有任务在等待且所有的这类资源已经被占用，而该任务又需要在一定的时间内完成，则任务管理器可以加入等待队列，或向其他超级 Peer 提交任务，超级 Peer 接受到任务后，依照上面的(1)或(2)处理，处理后的结果直接返回给用户。

基于 P2P_Grid 模型的任务调度，把整个任务调度过程分为了两个部分，即资源调度和任务调度。这种新的资源组织管理方式，把满足某类任务的资源也划分为相应的类别，并将该类资源的信息保存在资源管理器中，同时放入资源队列中。当有满足条件的任务调度时，若该类任务的资源队列不为空，即有资源可以满足待调度的任务，那么我们可以直接将资源分配给该任务，完成任务的调度。

这样的话，在 P2P_Grid 模型下，我们可以对任务按照其 QOS 要求对所有待调度任务进行分类。若是第一次调度该类任务，并且资源队列不为空，即有资源可以满足，那么将资源分配给该任务，同时在资源管理器中保留该类任务的资源信息；若不是初次调度该类任务，即以前曾经调度过该类任务，并且现有的资源队列不为空，可以满足该任务，则直接将资源队列中的资源分配给该任务，完成任务调度；若该类任务的资源队列

为空，即已经没有资源满足该任务，则该任务或着加入其所在子网格系统的任务等待队列；或着直接转移给其他空闲的并可以满足其所需资源的的超级 Peer 来处理该任务。

以上分析可以看出，基于 P2P_Grid 模型的任务调度即是集中式的同时也是分布式的。对于每个超级 Peer 负责处理其所在子网格系统的任务调度和资源调度，这可以看作是集中式的工作模式；而对于分布于不同子网格系统的超级 Peer 来说，它们之间是对等的，采用 P2P 协议，当某类任务在本地子网格系统不能被调度时，完全可以在其他的子网格系统内完成对其的调度，这时超级 Peer 之间会进行资源和任务的协同，共同完成对于任务的调度。这可以看作是分布式的工作模式。这样我们就弥补了现有的资源调度模式的缺陷，把集中式调度和分布式调度融合在一起，由超级 Peer 来完成整个调度。

由于 P2P_Grid 这种既集中又分布的特点，使得整个网格系统的资源利用率得到了提高；而由于采用 P2P 协议来处理分布于不同子网格系统的任务和资源，使得整个网格系统的负载可以取得一个较好的平衡。前面已经证明过，对于网格任务调度问题是一个 NP 完全问题，并不存在最优解，只能找到一个次优解。而本文所讨论研究的课题，其结果也只是在任务调度完成时间(Makespan)和负载平衡这两个性能指标上有了进一步的提高，我们也仍然不能找到一个最优的解决方案。仅仅是为以后的讨论研究提供了一点理论依据。希望可以为分布式系统下任务调度问题的研究开辟一个新的空间。

4.4.4 P2P_Grid 模拟环境的建立

这一节将详细论述本课题所研究的算法思想。如前所述，本课题所讨论的分布式系统下的任务调度是以网格为基本环境的，并在此基础上我们建立了新的环境模型 P2P_Grid 模型。之所以建立这样一个模型，是受到了 P2P 和 Grid(网格)存在许多相似点的启发。它们都是新型的分布式系统，而网格的最终目标就是 P2P。有了这样的理论依据，使得模型的建立成为了可能。现在的关键问题有两个：一是 P2P_Grid 模型的模拟建立；二是基于该模型下的算法思想。接下来，将分别来论述这两个问题的解决方案。

4.4.4.1 P2P_Grid模型的模拟环境的建立

(一)GridSim 模拟工具

基于试验环境的限制，建立一个真正的环境是有相当困难的，因此退而求次，我们选择现有的网格模拟器来建立试验所需模拟环境。目前网格调度模拟工具主要有：Bricks[43]，MicroGrid[44]，SimGrid[45]，GridSim[46]。

(1) Bricks 实现了在全球计算系统下不同调度算法的性能评估。它的基础模拟环境是在高性能计算机上提供科学计算函数包的以 C/S 模式运行的全球计算系统。它能够模拟资源的发现和查询、资源行为的预测、任务的虚拟处理、网络拓扑结构等。Bricks 允许开发者使用 Bricks 脚本语言来配置全球计算环境。但是这个工具现在无法下载到。

(2) MicroGrid 是为了研究网格资源管理而提供一个虚拟的网格基础设施。它是建立在 Globus 平台上，因此它的基础仿真环境是利用 Globus 工具包建立起来的网格环境。很好地仿真了资源的行为、网络、任务的处理、资源发现等。仿真结果接近实际，但需要用 Globus 构造实际的网格任务，因此适用于实际系统开发过程中对调度算法的仿真评测。

(3) SimGrid 为了模拟异构的分布式环境下的分布应用而提供一系列的核心函数。它的基础模拟环境是分布式计算平台，从简单的工作站到复杂的计算网格。能够模拟网络拓扑结构、资源、任务的处理等。SimGrid 提供了相当多的函数，使得模拟具有很大的灵活性，但整个过程显得比较繁琐和复杂，需要用户有一定的经验。

(4) GridSim 为网格有效资源分配技术的研究提供一个模拟环境。GridSim 的基础模拟环境是网格计算平台，主要是市场经济的网格计算平台。它能够模拟异构且分布全球的资源、简单的网络、资源的查找、任务的虚拟处理等。GridSim 采用面向对象技术构建，整个框架比较清晰，开发者能够很容易地理解，且开发应用也较简单。

在这几种模拟工具中，GridSim 模拟工具是更具优势的。与 SimGrid 相比，GridSim 主要针对网格计算，提供了网格的各种基本功能部件，并且模拟了各个功能部件之间的基本行为，使得开发者在这个模拟工具上很容易地实现调度模拟。与 MicroGrid 相比，GridSim 模拟采用了虚拟时间，不受主机性能的影响，而且不需要开发实际的任务。因此 GridSim 工具为网格资源调度算法的研究提供了良好的基础。

(二)体系结构

GridSim 采用分层的方法将复杂的网格调度模拟任务分解，每层专注解决一方面的问题。整个 GridSim 体系结构分为运行环境层、GridSim 工具和开发应用层，如图 4—2 所示[46]。

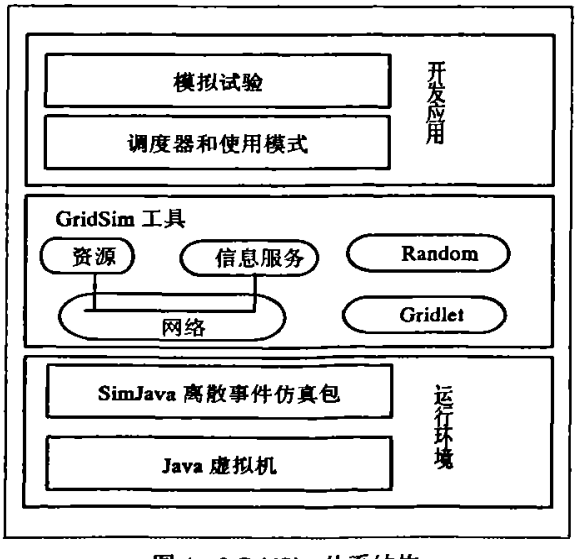


图 4—2 GridSim 体系结构

GridSim 工具包含多个实体，对应网格中的资源、网络、信息服务等，体现了它们的特性，模拟了它们在实际网格系统中交互行为。

资源实体主要由 GridResource 和一系列的辅助类实现的，有着众多的参数来描述资源的分布和异构特征。异构方面的参数主要有处理器的数目、处理成本、处理速度、内部调度策略、本地负载参数、机器数目等。

网络实体主要是通过两个实体 Input 和 Output 来实现的，主要模拟了数据在网络中的传输及由此产生的延迟。假设网络的拓扑结构是完全相连。这样的简化模型没有完全反映网络在实际网格中作用，但对于调度来说，是没有太多的影响。

另外 GridSim 工具还提供了其他实体，如 Gridlet、GridSimStandardPE、GridStatistics、GridSimShutdown。这些实体是为了方便使用者进行开发或者为了方便记录和控制模拟过程。

(三)基于 GridSim 的网格调度模拟

为了分析调度算法的性能，还需要在 GridSim 上进行开发。开发任务主要包括调度器模型的建立、网格使用模式模型的建立、调度模拟的管理。

调度器模型的建立是整个模拟的核心，要实现将任务分配到合适的资源上，并决定所有任务的执行顺序。因此调度器模型不仅包括调度算法的模拟，而且包括合适资源的发现、任务的提交和接收等。在开发过程中，为了便于处理，往往将调度器模型建成一个独立的类。一般来说，调度器的行为主要包含资源查询、调度决策、任务提交、任务结果接收等，GridSim 工具对这些行为提供了多方面的支持。资源查询行为可以调用 GridSim 类中的 GetGridResourceList( )方法来实现。调度决策过程中如果需要同资源进行交互来了解资源的意愿，可以调用 GridSim 工具中 GridSim 类的 GetResourceWish( ) 方法。任务提交行为可以调用 GridSim 类中的 gridletSubmit( )方法来建立。任务接收行为可以调用 GridSim 类中的 gridletReceive( )方法来建立。

网格使用模式模型指网格的高层功能模型，其开发内容涉及用户任务在网格中的表示、用户任务在网格中的提交方式、调度行为的管理。其中调度行为的管理可能包括调度器的启动和结束、调度器的创建等。

调度模拟的管理主要是网格用户模型和调度模拟实验的建立。网格用户模型主要包括与调度相关的用户的各种参数和行为，包括任务的要求、任务的创建、结果的处理等；调度模拟实验的建立主要是管理整个调度模拟行为，包括模拟的初始化、资源的建立、用户的建立、模拟的控制等。

另外，实体之间的通信是涉及比较多的行为，除了上面提到的各种方法，GridSim 工具还提供了比较基础的方法。发送消息可以调用 Send()方法或 sim_schedule()方法。接收消息可以调用 ReceiveEventObject()方法或 sim_get_next()方法。

4.4.4.2 算法思想的产生

本课题所讨论研究的任务调度算法是在一种新的模型的基础上，而并非产生了一种全新的网格任务调度算法。在本章的最开始的部分，我们列举了已经存在的几种经典的

启发式的异构环境下的任务调度算法，在此选取了Min_min算法作为本文所讨论的算法的基本思想。由于P2P_Grid模型是对资源组织管理的一种新的方式，在这种模型下，首先要对所有待调度的任务按照其所需资源的要求进行分类，进而对于资源相应的分类。也就是说在本文所讨论的任务调度算法要加入任务的QOS请求。新算法中将任务的QOS请求嵌入其中，这样可以更好的满足不同水平的QOS的请求。一般来说，QOS具体要求是与任务实际申请的资源密切相关的[9]，例如，对于网络来说，应用程序的QOS请求可能意味着需要满意的带宽；而对于CPU来说，应用程序的QOS请求可能意味着需要更高的处理速度或者CPU的利用率等。目前的大多网格调度算法中，很少有去考虑任务的QOS请求。这样的话，一个没有QOS请求的任务既可以获得高QOS资源也可以获得低QOS资源。而一个高QOS请求的任务就只能在系统有空闲的高QOS资源时才能执行。在不考虑QOS的调度算法中，可能会出现这样的情况，即低QOS的任务占用了高QOS资源，而高QOS的任务只能等待，低QOS的资源空闲。从而导致系统资源利用率不高，负载不均衡[47]。

4.5 本章小结

本章从任务调度和资源调度两个方面讨论了目前网格环境下的任务调度和资源调度的特点和存在的问题。对几种经典的网格任务调度算法进行了比较分析，给出了它们的缺陷和不足。在次基础上提出了本课题所研究的模型－P2P_Grid 模型。新模型中的超级 Peer 既是子网格系统的控制中心，同时也是它们之间也是对等的。这样对于任务的调度既可以是集中式也可以是分布式，可以与其他超级 Peer 进行协同工作。对于资源调度，由于对于用户所提交的任务总是先在子网格系统内部进行调度，资源同样也是进行内部进行调度的，这样就避免了由于长距离的资源调度而造成的通信时间的延长，降低了整个系统任务调度的平均完成时间，提高了系统资源的利用率。

第五章　基于 P2P_Grid 模型的任务调度算法的研究与改进

5.1 引言

Min_min 算法是经典的网格任务调度算法，它是一种在异构环境下的启发式调度算法。本课题所研究的任务调度算法以 Min_min 算法作为基本思想，并在次基础上加以改进。使得整个网格系统的 Makespan 和负载平衡得到提高。

5.2 Min_min 算法在 GridSim 模拟器上的研究

5.2.1 Min_min 算法的思想

首先来详细介绍一下 Min_min 算法的主要思想。Min_min 算法仍然是目前网格调度算法的研究基础之一，该算法的主要思想如下：

当需要调度的任务集合非空时，反复执行如下操作直至集合为空：

(1) 对集合中每一个等待分配的任务 Ti ，分别计算出把该任务分配到 n 台机器上的最小完成时间。假设任务在第 k 台机器上的完成时间为最小，记为 Min_Time (i) =MCT(i， k)，可得到一个含有 m 个元素的一维数组 Min_Time；

(2) 设第 a 个元素是 Min_Time 数组中最小的，其对应的主机为 b，把任务 a 分配到机器 b 上；

(3) 从任务集合中把任务 a 删除，同时更新 MCT 矩阵。

考虑在一个理想的 ETC(表 5.1)矩阵上使用 Min_min 算法，得到的平均完成时间(Maskspan)如图 5－1 所示。

表 5.1 一个理想的 ETC 矩阵

	h_1	h_2	h_3
T_1	2	4	8
T_2	4	8	16
T_3	8	16	32

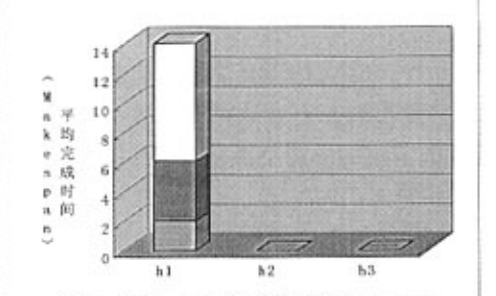


图 5－1 Min_min 调度算法的 Makespan

可以看出，如果用 Min_min 算法，会把所有的任务都调度到主机 h1 上，得到的 Makespan(任务完成的总时间)为 14，负载严重不均衡，同时对网格资源的利用率也非常低下。

5.2.2 传统 Min_min 算法的实现步骤

在给出具体的算法步骤之前，首先对算法中出现的变量做如下定义：

(1) ET_{ij}：待调度任务 T_i 在机器 M_j 上的预期执行时间；

(2) CT_{ij}：待调度任务 T_i 在机器 M_j 上的预期完成时间；

(3) START：任务 T_i 在机器 M_j 上的开始执行时间；

(4) AT：任务 T_i 的到达时间；

有了上述定义，则任务 T_i 在机器 M_j 上的预期完成时间 CT_{ij}=START+ ET_{ij}；则具体的算法步骤如下：

(1) while 有任务等待调度
(2) for 每一个待调度的任务 T_i
(3) for 每一个机器 M_j
(4) 计算 CT_{ij}=CT(T_i，M_j)；
(5) end for
(6) 计算所有任务中 CT_{ij} 最小的一个；
(7) end for
(8) 寻找最佳匹配，即将待调度任务 T_i 分配给机器 M_j；
(9) 计算任务完成的时间；
(10) 执行待调度任务 T_i；
(11) end while

5.2.3 Min_min 算法在 GridSim 模拟器上的实现

本课题所讨论研究的任务调度算法是以 GridSim 作为模拟平台的。本节，首先使用 GridSim 模拟器模拟 Min_min 调度算法。

Min_min 算法的思想不再详述，利用 GridSim 工具建立了 Min_min 调度算法的模拟平台。我们建立了调度器、网格的使用模式和模拟管理等多个模块，建立完整的网格系统模型，能够实现各种不同的应用场景来估价调度算法的性能。Min_min 调度算法模拟平台主要结构如图 5－2 所示，其中主要包括了 SimManager、User、GridInterface、GridBroker 和 UserTask 类。

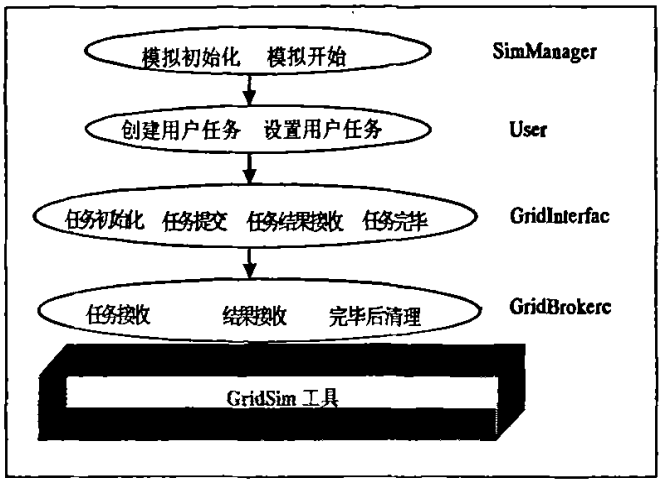


图 5—2 Min_min 调度算法模拟平台结构

GridBroker 类实现了调度模拟器模型。任务的完成时间的计算是通过与资源交互完成的。GridBroker 在收到 GridInterface 对象提交的用户任务后，它开始执行各项功能。在 GridBroker 收到任务处理完毕消息后，开始清理环境，并结束自身。

GridInterface 类主要实现了网络的使用模式模型。在这个模型中还有 UserTask 类，这个类主要保存用户任务的模型。

User 类主要实现了网格用户模型，保存了用户的各种参数，并提供了用户任务的创建、结果的接收等方法。

SimManager 类主要是建立各种模拟场景，包含创建各个对象、模拟的初始化、模拟的启动等功能。

对于 Min_min 调度算法模拟平台可以研究算法的多方面的性能，比如任务的执行性能、资源的利用率、资源负载的均衡性等。模拟仿真过程中，我们给定的任务数量为 150 个进行了 150 次仿真试验，分别对任务的 Makespan 和系统资源利用率进行了数据记录。如表 5.2 和 图 5—2 所示。

表 5.2 Min_min 算法的任务平均完成时间和系统资源利用率

高QOS请求所占比例	Makespan	系统资源利用率
20%	177	68%
50%	228.35	79.3%
80%	308.93	67.9%

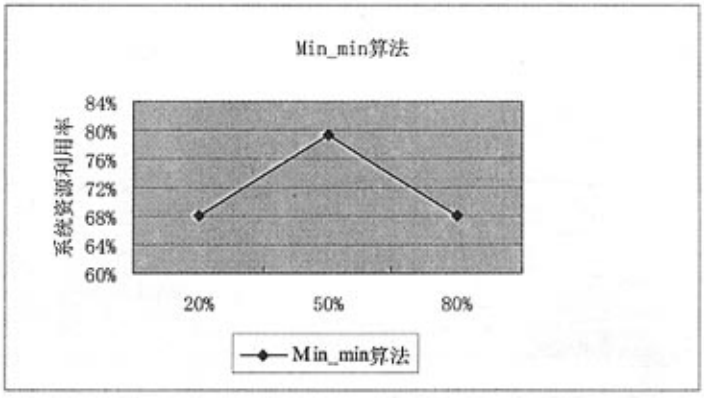


图 5－2 Min_min 算法的系统资源利用率

在仿真初始，我们将任务按照其 QOS 请求进行分类，3 次仿真我们分别使其高 QOS 请求的比例设定为 20%，50%和 80%。从图 5－2 可以看出，尽管将任务按其 QOS 请求进行分类，但是对于 Min_min 调度算法来说，系统资源利用率并没有因此而有所提高，系统平均负载性能也不理想。

5.3 QOS-Min_min 算法在 GridSim 模拟器上的实现

5.3.1 QOS-Min_min 算法

在第四章中已经讨论过，任务的 QOS 请求对于网格环境下的任务调度有着很重要的影响。一个没有 QOS 请求的任务既可以获得高 QOS 资源也可以获得低 QOS 资源。而一个高 QOS 请求的任务就只能在系统有空闲的高 QOS 资源时才能执行。在不考虑 QOS 的调度算法中，可能会出现这样的情况，即低 QOS 的任务占用了高 QOS 资源，而高 QOS 的任务只能等待，低 QOS 的资源空闲。从而导致系统资源利用率不高，负载不均衡。

本课题所研究的任务调度是将任务的 QOS 请求嵌入其中，这样可以更好的满足不同水平的 QOS 的请求。所谓 QOS-Min_min 算法就是将待调度的任务首先按照其 QOS 请求进行分类，分别将其放到不同的 QOS 请求任务调度队列中。根据空闲资源的情况决定调度哪个队列中的任务。为了简化讨论，我们将待调度任务分成两个集合，高 QOS 任务集合和低 QOS 任务集合。在本试验中，我们划分 QOS 请求高低的标准是根据待调度任务所需带宽的要求。规定带宽需求大于 10Gigabit/s 的任务为高 QOS 请求；带宽需求不足 10Gigabit/s 的任务为低 QOS 请求。并设定三次仿真中，高 QOS 请求的任务分别所占的比例为 20%，50%和 80%。

5.3.2 QOS-Min_min 算法的思想和步骤

QOS-Min_min 算法是对 Min_min 算法的改进，其思想仍然是基于 Min_min 算法的，只是在对待调度任务进行初始化时，首先将其按照 QOS 请求分为两个集合，即高 QOS 任务集合(我们用 Task_H 表示)，和低 QOS 任务集合(我们用 Task_L 表示)。然后分别按照 Min_min 算法的思想处理这两个任务集合。算法的具体步骤如下：

(1)for 所有待调度任务T_i

(2) for 所有空闲的机器M_j

(3) 计算 $CT_{ij}=ET_{ij}+b_j$

(4)do Task_H集合中的所有任务

(5) 对每一个高QOS请求的任务，寻找所有符合其QOS请求的所有机器；

(6) 计算任务T_k在所有满足条件的机器上的最小完成时间CT_{ij}；

(7) 将任务T_k分配给机器M_l；

(8) 从Task_H中删除任务T_k；

(9) 更新b_l；

(10) 更新CT_{il}

(11)end do

(12)do Task_L集合中的所有任务

(13) 对每一个低QOS请求的任务，寻找所有符合其QOS请求的所有机器；

(14) 计算任务T_k在所有满足条件的机器上的最小完成时间CT_{ij}；

(15) 将任务T_k分配给机器M_l；

(16) 从Task_L中删除任务T_k；

(17) 更新b_l；

(18) 更新CT_{il}

(19)end do

该算法共有三个循环，第一个循环是对所有待调度的任务计算其在每一台机器上的预期完成时间，并将其放到 MCT(i，j)集合中；第二个循环是处理高 QOS 请求的任务，首先从 MCT(i，j)集合中找到可以符合任务 T_kQOS 请求的所有机器，然后再次计算任务 T_k 在所有满足该 QOS 请求的机器上的最小完成时间。最后将任务 T_k 分配给预期完成时间最小的机器 M_l；更新 Task_H 集合，更新任务开始时间 b_l，更新预期完成时间 CT_{il}。第三个循环是处理低 QOS 请求的所有任务，方法与处理高 QOS 请求的任务是完全相同的。

5.3.3 仿真试验

仿真试验仍然在 GridSim 模拟器进行。首先将所有待调度任务按照网络带宽的需求分成了两个集合。对于任务带宽要求大于 10Gigabit/s 为高 QOS 请求，不足 10Gigabit/s 为低 QOS 请求。我们进行了三次仿真，其中高 QOS 请求的比例分别为 20%，50%和 80%。为了便于比较算法的性能，仿真的任务数量仍然是 150 个。Min_min 算法和 QOS-Min_min 算法的性能比较，如表 5.3、5.4 和图 5—3、5—4 所示。

表 5.3 两算法 Makespan (s) 的比较

高QOS 请求所占比例	Min_min	QOS-Min_min	Makespan 的比较
20%	177	163.25	7.77%
50%	228.35	202.28	11.42%
80%	308.93	303.93	1.62%

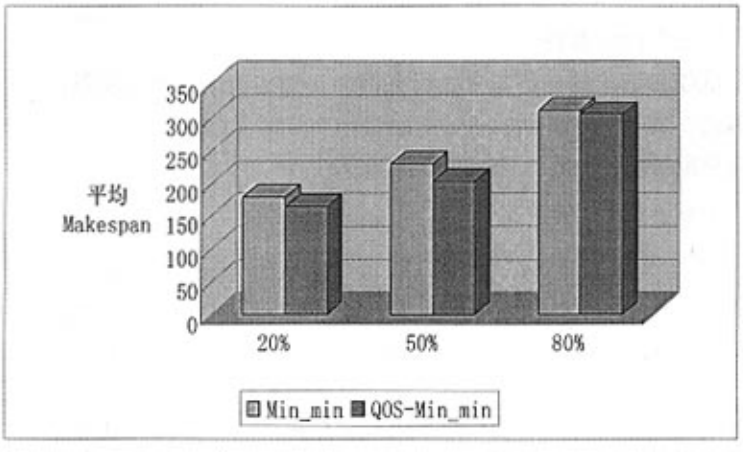


图 5—3 两算法 Makespan 的比较

表 5.4 两算法系统资源利用率的比较

高QOS请求所占比例	Min_min	QOS-Min_min	与Min_min比较
20%	68%	81.8%	13.8%
50%	79.3%	88.4%	9.1%
80%	67.9%	85.2%	17.3%

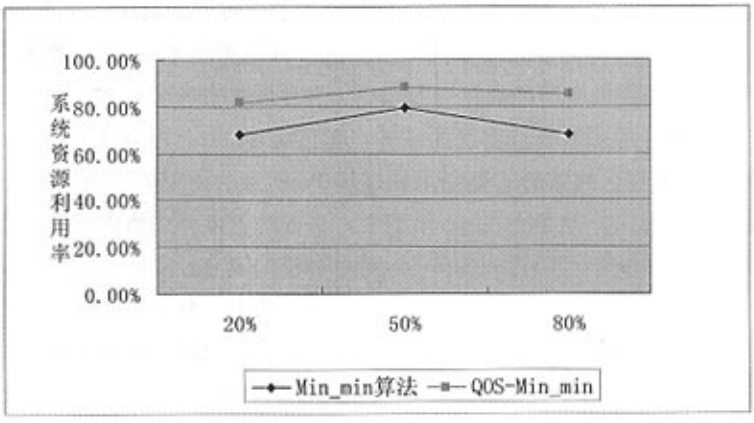


图 5—4 两算法系统资源利用率的比较

从以上的数据分析中可以看出，嵌入了 QOS 的 Min_min 算法在任务的平均完成时间(Makespan)和系统资源利用率上都有一定的提高。由此不难发现，QOS 请求在网格任务调度中所起到的重要作用。本课题所研究的任务调度算法不仅是在一个新的资源组织管理模型上进行的，同时，我们也考虑到待调度任务的 QOS 请求。

5.4 基于 P2P_Grid 模型的任务调度算法

5.4.1 P_G_Min 算法的思想

我们将新的算法命名为 P_G_Min 算法。该算法是在一种新的资源组织管理模型 P2P_Grid 模型研究讨论的。由于超级 Peer 的加入，使得整个网格系统的资源可以更好的协同。而好的资源调度算法同样对整个任务调度起着关键的作用。因此，该算法一个很重要的特点就是比传统的算法更加注重资源的协同。我们把资源调度作为任务调度的一部分，超级 Peer 把整个网格系统划分为若干个子网格，每个超级 Peer 作为其所在子网格的中心节点，负责处理该子网格内任务调度。当用户提交任务的时候，总是将任务提交给其所属的子网格系统的超级 Peer，而超级 Peer 总是先在子网格内部搜索待调度

任务所需的资源，完成任务的调度；只有当所需资源不能满足待调度的任务时，超级Peer才会考虑向相邻的其他超级Peer转移任务，与其他超级Peer协同工作，共同完成任务的调度。

在P2P_Grid模型中，我们使用资源管理器来记录每个超级Peer所负责的子网格系统的资源使用情况。当有某一类任务进行调度的时候，资源管理器会查找是否曾调度过该类任务，若调度过该类任务，则资源管理器中会记录了最优资源选择信息，若此时子网格系统中所需资源可以满足，则直接将资源分配给待调度任务，完成任务调度；若资源管理器中没有记录过该类任务的最优资源组合信息，则此时资源管理器会将可用资源的属性信息按一定的策略和级别高低顺序记录在可用的资源管理器中。同时选择最佳的资源或资源集合分配给所待调度的任务。可以看出，每个超级Peer都会有各自的资源管理器，用于记录某类任务所需的最佳资源组合，并将资源分配给任务。而对于那些本子网格系统不能满足任务所需资源需求的任务，则超级Peer会根据任务的优先级和紧急情况，考虑将待调度任务转移给其他空闲的超级Peer，来协同完成这一次的任务调度。

从以上分析可以看出，在P2P_Grid模型中，资源的调度是由资源管理器完成的，它不仅记录了子网格系统中可用资源的信息，同时会根据提交任务所需资源的要求对资源进行分类，放入资源队列中，以便进行资源的分配。而且，资源管理器还要及时掌握系统负载情况，一旦发现所在的局部系统的负载过重，就马上将任务转交给其它空闲的超级Peer处理，这样就克服了传统算法中负载不平衡的缺陷。

有了这样的资源调度策略，超级Peer就可以根据资源管理器里所记录的可用资源的信息完成任务的调度。下面给出具体的任务调度的步骤。

5.4.2 P_G_Min算法的步骤

P_G_Min算法中，我们仍考虑任务的QOS请求。与QOS-Min_min算法相同，将待调度的任务根据网络带宽的需求分为高QOS任务(Task_H)和低QOS任务(Task_L)两个集合。由于P2P_Grid模型对于资源调度的局部性特点，使得我们还要考虑的一个重要参数就是资源调度时间，用STi表示。假设待调度任务的预期执行时间为ETij，其开始执行时间为START，则任务的预期完成时间 $CT_{ij}=START+ET_{ij}$，其中任务开始执行时间 $START=AT^{i}+ST_{i}$，这里 AT^{i} 为任务到达时间。这样的话，$CT_{ij}=AT_{i}+ST_{i}+ET_{ij}$。新算法中资源调度是由超级Peer在其所属子网格系统内完成的，除非局部资源不能满足任务的需求，否则将在所属子网格内部进行资源调度，这样的话所有任务的平均STi必然大大降低，从而 CT_{ij} 的值也会随着大大降低。

在P2P_Grid模型中，每个超级Peer本身可以构成一个局部的网格系统，这样在用户提交任务时，除了考虑任务的QOS外，还要考虑其所属的超级Peer，用户总是将任务提交给他所属的超级Peer。一旦任务提交后，任务的调度由超级Peer完成。

P_G_Min算法中，对于属于相同超级Peer的同类任务仍然使用Min_Min算法的思

想来选取待调度的任务，这样可以保证任务的预期执行时间ETij最小。

算法实现步骤：

(1) 对所提交的所有任务按照任务的QOS请求及其所属的超级Peer进行分类，分别放在Task_H及Task_L两个集合中

(2) 对于Task_H集合中的每个任务Ti将其提交给所属的超级Peer；超级Peer按照上面所分析的三种情况处理该任务：

```
for  i=1 to m   // m 是Task_H集合中待调度任务总数
 1)if (task_queue= =null and resource_queue= =null)//即第一次调度该类任务
 {
    超级Peer搜索任务所需资源；
    返回资源最新状态信息；
    调度该任务；

 }
 2)if (task_queue= =null and resource_queue!=null) //有其它任务或该类任务正在执
                                                    行
 {
    超级Peer将任务加入等待队列中；
    把所需资源分配给该任务，等待调度；
     返回资源最新状态信息；
 }
 3)if (task_queue!=null and resource_queue= =null)//若任务队列不为空且资源队列
                                                    为空
    根据任务的紧急情况将其加入等待队列或转交其他超级Peer调度；
```

(3) 直到处理完Task_H集合中的所有任务

(4) 对于每个超级Peer，同样处理Task_L集合中的每个任务。

5.6 仿真结果

对于任务的QOS请求，在试验中我们仍然仅考虑其对带宽的要求，为了便于与Min_min算法和QOS-Min_min算法比较，仍然把高QOS请求的任务的所占百分比分别设置为20%，50%及80%。对每一种情况我们都对150个随机任务进行150次的的仿真测试，计算这150次仿真测试的Makespan(任务执行时间)的平均值。表5.5和表5.6，图5—6和图5—7分别给出了三种算法在Makespan 和系统资源利用率的仿真比较结果。

表 5.5 三种算法的 Makespan (s)的比较

高QOS请求所占比例	Min_min	QOS-Min_min	P_G_Min	与Min_min比较	与QOS-Min_min比较
20%	177	163.25	136.28	23%	16.52%
50%	228.35	202.28	172.25	24.57%	14.85%
80%	308.93	303.93	245.14	20.65%	19.34%

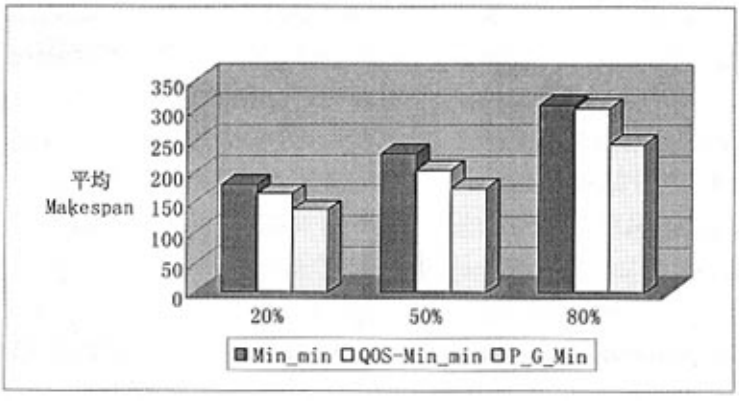


图 5—5 三种算法的 Makespan 的比较

表 5.6 三种算法的系统资源利用率的比较

高QOS请求所占比例	Min_min	QOS-Min_min	P_G_Min	与Min_min比较	与QOS-Min_min比较
20%	68%	81.8%	89.3%	21.3%	7.5%
50%	79.3%	88.4%	92.1%	9.1%	3.7%
80%	67.9%	85.2%	98.8%	30.9%	13.6%

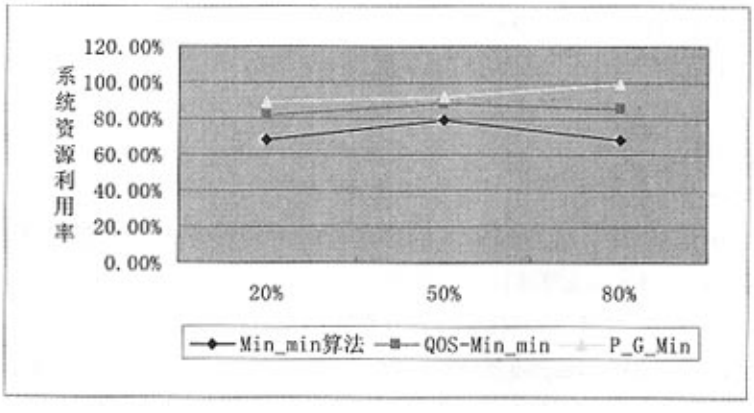


图 5—6 三种算法的系统资源利用率的比较

从仿真试验结果可以看出，对于不同比例的高QOS请求的任务来说，P_G_Min在Makespan和系统资源利用率这两项指标上总体要比Min_Min算法和QOS-Min算法都要好。当高QOS所占的比例为50%时，P_G_Min算法的Makespan要比传统的Min_Min算法和QOS-Min分别降低24.57%和14.85%；而在系统资源利用率的比较中我们可以看到新算法的系统利用率在80%时差异是最大的，分别提高了30.9%和13.6%。

5.7 本章小结

P2P_Grid模型是一种新的资源组织管理模型，研究在该模型基础上的任务调度算法是本章的目的。Min_min算法是传统的异构环境下的启发式算法，以该算法为基本思想，我们分别讨论了嵌入QOS请求和基于P2P_Grid模型的P_G_Min算法。通过仿真试验发现，QOS-Min_min算法和P_G_Min算法在任务调度的完成时间(Makespan)和系统资源利用率这两个参数上比传统的Min_min算法有了较大的提高。这说明，考虑待调度任务的QOS请求在任务调度中起着很关键的作用。对待调度任务根据其QOS请求进行分类调度，就避免了资源空闲和任务等待，从而可以提高系统资源利用率，同时缩短了整个系统的任务调度时间。

对于P_G_Min算法不仅考虑了任务的QOS请求，同时利用P2P技术很好的解决了资源和任务的协同问题。说明了资源调度在整个任务调度的过程中的关键作用。P2P_Grid模型利用超级Peer既可以集中调度又可以分布协同的特点，首先在局部子网格系统内完成任务的调度，这体现了集中调度的特点；当局部子网格资源不能满足任务的请求时，才考虑将任务转移给其他空闲的超级Peer，协同完成任务的调度，这体现了分布调度的特点。P_G_Min任务调度算法是与资源调度紧密结合起来的，把大型的网格系统局部化，尽可能在小范围内完成任务和资源的调度是本课题所提出的模型的目标所在。通过试验可以看出，P2P_Grid这中兼具了集中和分布特点的模型，在任务的平均调度时间和系统资源利用率上比Min_min算法和QOS-Min_min算法都有了很大的改善，达到了课题研究的目的。

第六章 结论和未来的工作

任务调度和资源调度是分布式系统设计的一个很重要的资源模块。由于这种优化组合问题是NP完全性的，除了在一些极少数特殊的任务图模型中可以找到最优解外，一般很难在多项式时间内找到最优解。本课题所讨论的分布式系统下的任务调度算法是以网格系统作为研究平台，并利用P2P对等技术的特点建立了一种新的资源组织管理模型P2P_Grid模型。

本文首先分别讨论了网格计算和对等计算的概念，技术特点等问题，说明网格计算和P2P都是一种新型的分布式系统，它们都是分布式系统的子集。而且，网格的最终目标就是P2P，这两种技术在目标、资源发现等很多方面都存在着相似点，这些都为新的模型的建立提供了可靠的理论基础。

P2P_Grid模型是资源组织管理模型，通过超级Peer将大型的网格系统划分为小规模的子网格，每个超级Peer作为子网格系统的中心，负责该网格内部的任务调度；而对于不同子网格之间的超级Peer是对等的，它们利用P2P协议协同工作。也就是说，P2P_Grid模型中的超级Peer既是其所属子网格系统的中心，对所提交的任务进行集中式调度；同时它们之间是对等的，可以与其他超级Peer协同工作，完成所提交任务的调度，这种调度模式又是分布式的。因此，该模型兼具了集中和分布的特点。

P2P_Grid模型对资源的调度是通过资源管理器完成的，在资源管理器中记录了子网格系统的所有可用资源的属性，通过任务的QOS请求将任务进行分类后，每类任务调度时所需的资源也作了相应的分类，放入资源队列中。同时资源管理器还会及时掌握系统负载的情况，当某个超级Peer的负载过重时，资源管理器就会通知子网格系统，以便将待调度任务转移给其他空闲的超级Peer。这样不仅保证了子网格系统的资源的利用率，同时也保持了较好的系统负载均衡性。

在详细讨论了P2P_Grid模型后，使用GridSim模拟器建立了网格模拟环境。并以传统的Min_min算法作为讨论的基础算法，首先在模拟器上实现了对于Min_min算法的模拟，给出了仿真试验数据。随后，将任务的QOS请求嵌入到待调度任务中，提出了改进后的QOS-Min_min算法，该算法是将任务根据其QOS请求进行分类，这样可以有的放矢的分配系统资源，减少了资源空闲，缩短了整个任务的调度完成时间。通过仿真实现，与Min_min算法比较，验证了算法的有效性。最后，提出了基于P2P_Grid模型的任务调度算法P_G_Min算法。期望可以在任务调度的平均完成时间(Makespan)和系统资源利用率两个性能上可以比Min_min算法和QOS-Min_min算法都有所改善。仿真试验证明，基于P2P_Grid模型的P_G_Min任务调度算法达到了预期的效果，对比三个算法，发现QOS-Min_min算法的性能优于Min_min算法，而P_G_Min算法在Makespan和系统资源利用率两个性能指标上都优于Min_min算法和QOS-Min_min算法。

前面已经讨论过，分布式系统下的任务调度算法并不存在最优解，本课题所研究的

任务调度算法也是一种启发式的算法，同样不是最优方案。本课题的所讨论任务调度算法的目的是希望通过P2P_Grid模型的提出，为网格任务调度算法提供一个新的研究空间。

另外由于自己的研究水平和研究时间所限，本课题还有许多有待完善和继续探讨的问题留在未来的工作中予以解决：比如，我们所提出的算法中对于QOS只考虑了对带宽的要求，今后可以考虑对QOS请求的细化，在多维空间中讨论QOS请求。而对于超级Peer也存在许多的研究空间，例如在网格系统中加入多少超级Peer的最为合适，以及当将任务提交给其它的超级Peer处理时，如何建立一个合理的搜索机制等。这些问题都可以成为我们以后研究工作的重点。

网格计算和对等计算都处于飞速发展的阶段，从底层到高层应用都以取得很多成果，但还是有很多问题需要解决。比如网格领域中的体系结构问题、安全问题、跨域资源管理与联合调度问题等，而P2P中也存在很多问题，比如资源查找、资源管理、安全与版权、互操作问题等。目前网格与P2P还没有很好的结合，而本文所提出的P2P_Grid模型也是从理论上进行研究的，在GridSim模拟器中，并没有真实的加入超级Peer，也仅仅是对于所建立模型的一个模拟而已。

总之，未来的发展方向必然是将P2P与网格合到一起[48]，目标是建立一个可扩展的可靠的网络空间资源共享系统，最终提供一种基于Internet的高效的分布式应用基础平台。

致谢

我的毕业论文至此已全部完成了，三年紧张而丰富多彩的研究生学习生活也将马上结束了。三年的学习生活使我受益良多，江南大学以及信息工程学院良好的学习氛围，使我开阔了眼界，专业水平也得到提高，这使我对今后的工作充满了信心。值此论文结束之际，谨向在研究生学习和生活期间为我倾注了大量心血的老师和提供帮助的同学表示深深的谢意。

首先，感谢我的导师张曦煌副教授，张老师不但教会了我从事科学研究的方法，还为我指明了以后工作和事业的方向。他谦逊的作风、严谨的治学、渊博的知识、敏锐的洞察力永远是我心中的楷模。

其次我要感谢王士同、须文波、顾耀林、冯斌和刘渊等老师，他们严谨的工作和治学态度，以及平时对我的科研和生活上的指导和关心都给我留下了深刻的印象。另外，我还要感谢徐黛岩老师在学习和生活上给予我的极大帮助。

同时，我要向在学习和生活上给予我极大帮助的杜精益、董占华、山艳、管芳景、张丽娜等同学，以及同一实验室的张伟宏、夏国武、高森、夏伏洋、韩忠海等同学表示深深的谢意。

最后，非常感谢我的父母和家人，没有他们的关心和支持，就没有我现在的成就，就没有我的将来。

仅以此文献给上述所有人！

赵巍

2007年1月于江南大学信息工程学院

参考文献

[1] 黄道颖，黄建华，庄雷等，基于主动网络的分布式P2P网络模型[J]. 软件学报，2004，15 (7)：1081-1089
[2] 都志辉，李三立，陈渝等. 网格计算及其原型实现研究[J]. 计算机科学，2002，29(8)
[3] 中国网格信息中转站http://grid.cs.tsinghua.edu.cn/
[4] Maheswaran M, Ali S, Siegel H J, etc, 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, 30-44.
[5] 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.
[6] 刘忠中. 网格计算及其技术需求分析[J]. 江西通信科技，2003，(2)：36-40.
[7] Ullman J. NP-Complete Scheduling Problems [J]. Journal of Computer and System Sciences, 1975, 10: 384-393.
[8] GGF6 Grid Scheduling Architecture [EB/OL]. http://ds.etechnik.uni-dortmund.de/～ yahya/ggfsched/WG/sched_arch/schedarch1.doc, 2005.
[9] He XS, Sun XH, Laszewski GV. QoS Guided Min-Min Heuristic for Grid Task Scheduling [J]. J Compute. SCI & Technology, 2003, 18 (4): 442-451.
[10] Min-You Wu, Wei Shu, Jun Gu. Efficient Local Search for DAG Scheduling [J]. IEEE Transactions and Parallel and Distributed Systems, 2001, 12(6): 617-628.
[11] Foster I, Kesselman C, Nick J, etc. The Physiology of the Grid:An Open Grid Services Architecture for Distributed Systems Integration[EB/OL]. http://www.globus.org/research/papers/ogsa.pdf, 2002, 210-217.
[12] Chap in SJ, KatramatosD, Karpovich J, etc.Resource Management in Legion [J]. Future Generation Computer Systems, 1999, 15(526):583-594.
[13] W Allcock, A Chervenak, I Foster, etc. The Data Grid: Towards an Architecture for the Distributed Management and Analysis of Large Scientific Datasets [J]. Journal of Network and Computer Applications, 2001, 23: 187-200.
[14] Spring N, Wolski R. Application Level Scheduling of Gene Sequence Comparison on Metacomputers [C]. Melbourne: the 12th ACM Int' 1Conf. on Super computing, 1998, 141-148.
[15] Buyya R, Chap in S, Dinucci D. ArchitecturalModels for Resource Management in the Grid [A]. The 1st IEEE /ACM International Workshop on Grid Computing[C]. London:Springer-Verlag, 2000, 18-35.
[16] Czajkowski K, Foster I, Karonis N, etc. A Resource Management Architecture for Meta-computing Systems [A]. Proc. of OPPS/SPDP' 98 Workshop on Job Scheduling Strategies for Parallel Processing[C]. Berlin: Springer-Veralg, 1998:62-82.
[17] Steve J C, Katramatos D, etc. Resource Management in Legion[A]. Workshop on Job Scheduling Strategies for Parallel Processing[C]. http://legion.virginia.edu /papers/legionrm.pdf, 1999, 1-18.
[18] Tannenbaum T, Foster I, Livny M, etc. 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, etc. Grid Resource Discovery Based on Routing-transferring Model[C]. The 3rd ACM / IEEE International Workshop 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] Noriyuki F, Kenichi H. A Comparison among Grid Scheduling Algorithms for Independent Coarse-Grained Tasks[C]. Proceedings of the International Symposium on Applications and the Internet Workshops(SA INTW' 04), 2004, 674-680.
[22] 都志挥，陈渝，刘鹏. 网格计算[M]. 北京:清华大学出版社，2002, 1
[23] 徐志伟，因特网之后是什么?网格技术探讨
http://www.it.simbnet.com/scyj/xjlt/0110-2.htm, 2006.
[24] I.Foster and C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, San Fransisco, CA. http://mkp.com/grids, http://www.gridforum.org/, http://www.ccgrid.org/, 1999.
[25] Ian Foster The grid: a new infrastructure for 21st century science [EB/OL]. http://www. aip.org/pt/vol-55/iss-2/p42.html.February, 2002
[26] 桂小林，基于hrternel的元计算系统的关键技术研究[D]博士学位论文.西安:西安交通大学，2001
[27] The Jxta Solution to P2P suns new network computing platform establishes a base infrastructure for peer to peer application development.
http://www.javaworld.com/javaworld/jw-10-2001/jw-1019-jxta_p.html.
[28] Manoj Parameswearan, Anjana Susarla, Andrew B Whinston. P2P networking: An information Sharing Alternative [J]. Computer, 2001, 34(7):31-38
[29] Domenico Talia, Paolo Trunfio. Toward to synergy between P2P and grids [J]. IEEE Internet computing, July/August, 2003, 96: 94—95.
[30] Ian Foster, Adriana Iamnitchi. On death, taxes, and the convergence of Peer-to-Peer and grid computing[C]. Second International Workshop on Peer-to-Peer Systems (IPTPS' 03 Revised Papers). Lecture Notes in Computer Science, Volume 2735, Springer-Verlag Heidelberg, 2003, 118-128.
[31] Frank Dabek etc. Wide-area cooperative storage with CFS[C]. In: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP' 01), 2001, 202-215
[32]Druschel P, Rowstron A, PAST: A large-scale, persistent peer-to-peer storage utility[C]. The 8th Workshop on Hot Topics in Operating Systems(HotOS-VIII), Germany, IEEE Computer Society, May 2001, 75-80.
[33] Sean Rhea etc. Maintenance-free global data storage[J]. IEEE Internet Computing, September/October 2001, 5(5):40-49.
[34] Jonathan Ledlie. Scooped, again[C]. Second International Workshop on Peer-to-Peer Systems(IPTPS' 03 Revised Papers), Lecture Notes in Computer Science, Volume 2735, Springer-Verlag Heidelberg, 2003, 129-138.
[35] Ian Foster. What is the grid? a three point checklist[EB/OL], http://www-fp.mcs.anl.gov/~foster/Articles/WhatIsTheGrid.pdf
[36] THOMSON S, NARTEN T. IPv6 Stateless Address Auto configuration[EB/OL]. IETF RFC 2462, http://www.ietf.org/rfc/rfc2462.txt, 1998.

[37] Vincenzo Di Martinol Scheduling in a grid computing environment using genetic algorithmsl Marco Mililotti the 16th Int' l Parallel and Distributed Processing System(IPDPS2002), Florida, USA, 2002
[38] J Watts, S Taylor. A practical approach to dynamic load balancing[J]. IEEE Transantion on Parallel and Distributed System, 1998, 9(3): 235-248
[39] R. Armstrong, D. Hensgen, and T. Kidd. The relative performance of various mapping algorithms is independent of sizable variances in run-time predictions [J]. In 7th IEEE Heterogeneous Computing Workshop (HCW ' 98), Mar. 1998, 79-87.
[40] H. Chen, N. S. Flann, and D. W. Watson. Parallel genetic simulated annealing: a massively parallel SIMD approach[J]. IEEE Transactions on Parallel and Distributed Computing, Feb. 1998, 9(2):126-136.
[41] I. D. Falco, R. D. Balio, E. Tarantino, and R. Vaccaro. Improving search by in corporating evolution principles in parallel tabu search. In IEEE Conference on Evolutionary Computation, 1994, 823-828.
[42] K. Chow and B. Liu. On mapping signal processing algorithms to a heterogeneous multiprocessor system[J]. In ICASSP 91, May 1991, 585-1588.
[43]Aida K, Takefusa A, Nakada H, etc. Performance Evaluation Model for Scheduling in a Global Computing System[J]. The International Journal of High Performance Computing Applications, 2000, 14(3): 268-279
[44]Song H J, Liu X, Jakobsen D, etc. The MicroGrid: A Scientific Tool for Modeling Computational Grids[C]. Proceedings of the 2000 Conference on Supercomputing, 2000, 4-10
[45]Legrand A, Marchal L, Casanova H. Scheduling Distributed Applications: the SimGrid Simulation Framework[C]. Proceedings of the third IEEE International Symposium on Cluster Computing and the Grid, 2003, 138-145
[46] Buyya R, Murshed M. GridSim: a Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing [J]. Concurrency and Computation: Practice and Experience, 2002, 14(13-15):1175-122
[47] Cao Junwei, Daniel P Spooner, etc. Agent based Grid Load Balancing Using Performance driven Task Scheduling [C]. International Parallel and Distributed Processing Symposium, 2003, 49-58.
[48] Andrew A Chien. Are grid standards suitable for P2P[EB/OL]. Chicago USA, http://www-csag.ucsd.edu/P2P/-Grid/Chien.ppt, October 5, 2003.

攻读硕士阶段发表的论文

1、赵巍，张曦煌 《Research of Scheduling Algorithm Based on P2P Technology》已发表于DCABE2006(第五届国际电子、工程及科学领域的分布式计算和应用学术研讨会)。

2、张曦煌，赵巍 《无线传感器网络中基于 Robust Position 节点定位算法改进的研究》已发表于核心期刊《计算机工程与应用》。