清 华 大 学

综 合 论 文 训 练

题目：网格数据流应用中任务管理和资源分配的算法设计和优化

系　　别：自动化系

专　　业：自动化专业

姓　　名：沈 虎

指导教师：曹军威　研究员

2008 年 6 月 6 日

签　名：_____导师签名：_____日　期：_____

关于学位论文使用授权的说明

本人完全了解清华大学有关保留、使用学位论文的规定，即：学校有权保留学位论文的复印件，允许该论文被查阅和借阅；学校可以公布该论文的全部或部分内容，可以采用影印、缩印或其他复制手段保存该论文。

(涉密的学位论文在解密后应遵守此规定)

签　名：　导师签名：  　日　期：

中文摘要

上个世纪 90 年代早期网格计算的理念被提出，这这项技术旨在整合网络中的计算资源，使其能够达到电力网络中的电力“即插即用”的效果，它被认为是“下一代网络框架的基石”。数据流应用是网格计算框架中的一个重要的组成部分，特别是在处理海量数据的传输及实时处理时，需要综合考虑网络带宽资源、计算网络的存储资源及计算资源等的各种限制性条件，此时采用流式数据传输及实时处理是一种适宜的选择，但如今数据流应用目前来说尚不成熟，因而对其进行研究和探索是非常有意义的工作。

本文的工作主要包括下面二个方面：一是结合美国 Laser Interferometer Gravitational-wave Observatory (LIGO)这个实际工程项目背景构建一套基于网格计算网络的数据流应用框架和机制包括寻求令人满意的任务管理和资源分配的解决方案；二是进行模拟仿真和以 Globus、Condor 等现有的网格计算平台工具搭建现实网络，进行实际测试。

通过测试可以发现，引入按需传输数据和及时数据清理等机制可以满足应用的要求，使得计算网络在有限的存储及带宽的条件下尽量充分地利用其计算资源，在整个过程中海量数据不断地以数据流的方式进行传输。实验结果还表明我们的网络环境具有良好的健壮性和自适应性。

当然本文所做的工作还有许多不足，考虑多个数据处理应用间的数据分享问题，尝试利用除遗传算法之外的其他启发式算法进行优化，扩宽我们实验的计算网络等都是本文后续可以研究的方向。

关键词：网格计算 数据流应用 任务管理 资源分配

ABSTRACT

The term Grid computing originated in the early 1990s as a metaphor for making computer power as easy to access as an electric power grid and this technology is considered as “the Blueprint for a new computing infrastructure”. Data streaming application is one important part of Grid computing framework, particularly when involving in dealing with transferring and just-in-time processing massive amount of data, if taking account of various limitations including network bandwidth resource, storage resource of computing network and its computing resource and so on, it is a considerable choice to introduce mechanisms of Data streaming transfer and just-in-time process. But this technology is not well developed yet, so it’s a meaningful job to explore and reach in this area.

This paper introduces three phrases of our work: Part one designs a series of Data streaming application framework and mechanisms based on Grid computing network, jointly with American Laser Interferometer Gravitational-wave Observatory (LIGO) project including the work of seeking a satisfying admission control and resource scheduler solution; Part two simulates it with computer tools and builds a actual network with existing Grid computing framework tools such as Globus and Condor, then tests the solution.

Experimental results show that the data streaming environment can scale well regarding storage usage and adapt to dynamically changing application data processing requirements especially with constraints of limit storage and bandwidth. On demand data transfers and just-in-time data cleanups are proposed to meet application requirements. In order to utilize computational resources with limit storage and bandwidth, large amount of data have to be streamed for processing. And Experimental results also show good scalability and adaptability of our environment.

Ongoing work include the consideration of data sharing scenarios among multiple data processing applications. Also some heuristic scheduling algorithm except Genetic Algorithm is under development for refined performance optimization. We are trying to join the OSG with the Condor pool at Tsinghua University so that data streaming and application enabling in a larger scale could be carried out.

Keywords： Grid computing Data streaming application
Admission control Resource allocation

主要符号对照表

Grid computing	网格计算
Data streaming application	数据流应用
Adimission control	任务管理
Resource allocation	资源分配
The Laser Interferometer Gravitational-wave Observatory, LIGO	引力波观测激光干涉项目
Globus Toolkit	Globus 工具箱
Application Queue	任务队列
Processor	处理终端
Storage	存储资源、存储器
Computing pool	计算资源池
Scheduler	调度管理器

目　　录

第1章 序言

1.1 网格计算的起源及发展[1]

网格计算是一门新兴的网络技术，它旨在将一个松散的计算机网络整合为一个集成的、内部有着密切联系的动态虚拟合作组织，以便可以分享散布于整个世界各地的各种资源：为用户特别是科学领域的研究人员提供前所未有的计算能力、服务和信息。

1.1.1 什么是网格计算

网格计算的理念可以描述为如下几层意思：

1.由分布式计算资源网络环境提供的在线计算或存储服务，比如说效用计算（utility computing）、按需计算（on-demand computing）或云计算（cloud computing）。

2.将同一个组织中的稀疏计算资源构建为一个“虚拟的超级计算机”。

3.将同一网络中散布于不同地理区域的计算机组织成一个“虚拟的超级　计算机”。志愿者计算（volunteer computing）是其中此项技术中最常用的一项应用，主要用于解决科学上的、数值性的问题。

网格计算产生于研究及商业上的需求，它充分利用了广泛的稀疏计算能力和能够将这些资源联系在一起的无所不在的英特网，而且网格计算的优势还在于它不会破坏网络的动态连接性。如今，计算机是根据其可能的最大负荷单个独立被构建、定制的。而它们的最大负荷可能一个月、一个季度甚至一年才发生一次，使得许多计算资源在大多数时候是处于低效利用状态。这正是网格计算致力于解决的难题。

网格组织的动态调整是是网格计算的另一项主要的优点，这项功能得益于网格计算网络可以由一些小的、标准的、内部可以改变的组织形成，所以计算网络可以先从小的、简单的组织开始，然后不断地通过增加更多的此类组织来扩大网络，这意味着网格组织可以得到急剧地增长。

1.1.2 网格计算的起源

网格计算如同英特网是在学术研究团体中诞生的，它的初衷是为了满足合作和分享计算资源的需要。早期的网格计算使得研究组织之间可以更有效率地分享海量数据和计算资源，这不仅使得他们可以大大地缩短计算时间，同时也提升了相互之间的合作质量。

网格的理念最早可以追溯到上个世纪八十年代晚期和九十年代初期的成批调度程序的思想中。九十年代见证了一系列的分布式技术如点对点技术的兴起同时也对计算能力的分享、付费及其统一标准提出了进一步的需求：人们采用统一标准架构体系可以高效地分布地使用电力资源（可以将其称为电力网格），那么计算资源为什么不能被这样使用呢？

同比于英特网使得计算机之间能够实现相互间的通信，网格计算能够使得计算机在一起协同工作。网格计算是英特网技术后的另一次革新，它与分布式计算、点对点计算、虚拟合作技术和 Web 服务等一系列下一代网络技术，为商务及科研应用提供了一条更新的、更强大的方式。

1.1.3 驱动网格计算发展的经济动力

在过去的十几年里，企业和机构对 IT 领域给予了大量投资来支持日益增长的客户需求。这些投资形成了一个高度分散的、不能被轻易管理的网络框架。

据估计有相当大一部分的企业计算资源只利用了其实际能力的 20％至 40％，造成这种浪费现象的因素主要有下面三个：

1. 在每种个操作平台上只运行一个应用；
2. 系统是根据预想的最大工作负载需求定制的，这往往比实际运行负载要大得多；
3. 负载顶峰持续时间往往非常短。

在引入一些简单的技术、更大的带宽、自动化方式和更简便的配置网络后，网格计算提供了一种更高效的方式，使得人们能够非常有效地提供和管理 IT 服务。

网格计算带来的深远影响将是非常令人兴奋的，它以相当小的代价使得计算机在其整个运行时间内保持在 100％的计算状态。企业和研究机构这些服务提供者将从中获取巨大的利益。

同时，网格计算能够合理地将有限的各种资源分配给散布于各地的企业以保障他们的商务和研究应用，这项技术为企业和研究机构提供了灵活性，无论是通

过重新整合资源来来适应新的机遇还是使得应用能够更好地为现有客户提供服务。

1.1.4 当今网格计算的主要应用

UK e-Science Programme

该项目为许多致力于网格计算的跨不同学科领域的工程项目提供资助，包括尝试构建虚拟天文台的 AstroGrid 计划，数据管理和分享领域的 Axiope 计划，实现模生物分子模拟仿真的 BioSimGrid 计划等等。

Large Hadron Collider Computing Grid Project

位于瑞士日内瓦的 the European Organization for Nuclear Research（CERN）实验室拥有世界上最大也是最强大的粒子加速器 the Large Hadron Collider (LHC)，所以他们对计算能力有着极大的需求，正计划通过部署全球范围了的计算网格服务即构建一个虚拟的计算组织将涵盖欧洲、美洲和亚洲的科学计算中心的计算资源整合到一起。

My Grid

这是一个旨在支持将生物实验网格化而开发的开放源计划。

DOE Science Grid

这个项目由美国能源部开发和部署，构建了一个基础架构来服务于高级科学应用和问题解决流程。

Globus Alliance

它是一项将网格的理念引入到科学和工程计算领域的研发项目。这个组织致力于开发各种网格应用、为网格计算方面的研究提供技术及方向性的知道意见、开发各种支持网格计算的程序（比如说 Globus Tookit）以及为未来网格发展提供标准建议。

1.1.5 网格计算的未来

全球范围内有共计数亿台个人计算机，但是计算能力是高度分散且远远没有达到被充分利用的水平。因特网上最知名的分布式计算的应用 SETI@Home 就为利用这些潜在计算资源提供了一个很好的样例，它的初衷是收集闲置的计算资源来为外太空领域的研究工作，现在它已经运行在分布在 226 个国家、超过 250 万台个人计算机上了。

组织虚拟化可以推动网格计算利用这些闲置的计算资源，Web 服务可以作为网格计算中动态传送机制的基本技术使得企业间能够动态、经济地传送软件，而且可以根据需要随时调整虚拟组织的大小。第三代网格计算平台（由 Optena 开发的 GridSpaces）就可以实现上述分布式计算的设想。

随着技术、网络和商业模式的成熟，网格将被广泛地应用到企业和研究机构当中，通过链接无以计数的海量资源为人们之间的通信，数据访问和计算提供服务。同时也可以期待网格计算将以 Web 页面的简便形式成为因特网上传输平台的常用服务。

1.2 问题提出

依据网格计算概念搭建起来的网络整合得到的难以想象的计算能力为一些特殊领域的工作如科学研究带来革命性的影响。与我们实验室（清华信息研究院未来信息技术研究中心 LIGO 工作组）合作密切的美国引力波观测激光干涉项目（Laser Interferometer Gravitational-Wave Observatory, LIGO）就是其中最为典型的一个例子。

1.2.1 LIGO 项目

开始于 1992 年的 LIGO 是 MIT 和加州理工的合作项目，由美国国家科学基金会（National Science Foundation，NSF）赞助。它的主要工作是寻找引力波（见图 1.1）——由黑洞碰撞和恒星爆炸产生的微弱的时空涟漪，探测到这些涟漪将为爱因斯坦的相对论提供新的检验，同时也成为探寻宇宙中不可见部分的一种方法[3][4]。

LIGO 拥有两套干涉仪，在路易斯安娜州李文斯顿的干涉仪（见图 1.2）有一对菲比在 1.2 米直径的真空管中的 4 公里长的臂，而在华盛顿州汉福较小，有一对 2 公里长的臂。

这两套 LIGO 干涉仪在一起工作构成一个观测所，这是因为激光强度的微小变化、微弱地震和其它干扰都可能看起来像引力波信号，如果是此类干扰信号，其记录将只出现在一台干涉仪中，而真正的引力波信号则会被两台干涉仪记录。所以，科学家可以对两个地点所记录的数据进行比较得知哪个信号是噪声[5]。

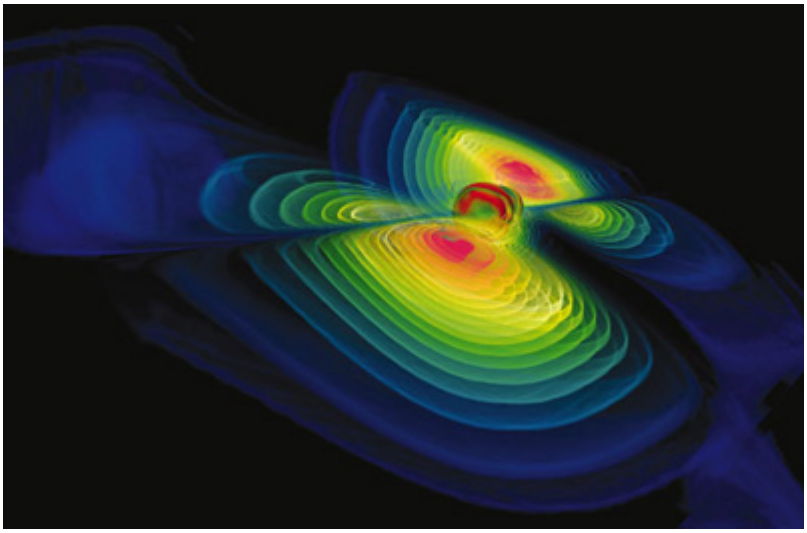

图1.1　黑洞与中子星相碰撞释放出引力波模拟图

图1.2　路易斯安那州的LIGO观测所

1.2.2 数据流需求

LIGO 观测仪中的数千个传感器无时无刻地产生实验数据，每天有 1TB（2^{40} B）的海量数据需要传送到散布于世界各地的实验室（见图 1.3 和图 1.4）进行数据分析处理，这不仅对数据处理能力、网络传输能力和数据存储能力提出了极高的要求，还需要数据产生端和处理端协同工作。目前参与 LIGO 计划的各个大学中超级计算机集群和一个名为 Einstein@Home 的分布式计算网络搭建开发科学网格提供的计算能力满足 LIGO 项目的计算需求，但是由于单个计算机或集群所拥有的网络带宽资源和存储能力都是非常不足的，这就带来了数据与计算能力分离的现象，数据流的思想提供了一个很好的解决方向[6]。

图1.3　汉福控制实验室

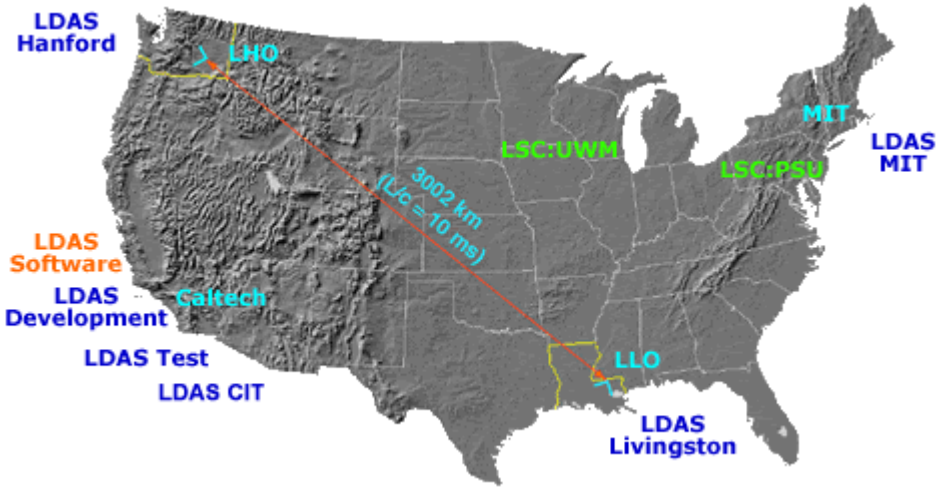

图1.4　LIGO干涉仪和主要实验室地理分布图

数据从产生端持续不断地传送到数据处理端，当计算机或集群的存储器中存储的数据达到了一定下限时，数据处理端就可以对存储的数据进行处理，处理完后保留少量结果数据同时及时清除存储器中处理过的数据，这样的流式的数据传输和处理方式能够大大降低对网络带宽资源和存储能力的要求。但是数据流式应用同时也对网格计算资源池中的任务管理和资源分配等方面提出了新的要求：我们建立任务管理队列，将不同任务合理地分配到计算池中适宜的计算机上，实现计算资源被最大化利用，进一步的要求是能够实现各个处理终端间的相互通信，

防止出现冗余处理并实现结果共享；我们也要考虑如何最优化地利用从属于整个计算集群网络带宽资源和存储资源并进行多方面地协调。

我们理想的数据流应用具有以下特征：

1. 数据传输和处理满足各种限制性条件，特别是不能出现集群存储器数据溢出，同时单位时间传输数据总容量小于网络总带宽等等；

2. 任务队列的管理必须做到队列平均等待时间最短同时又要避免出现某个任务被无限挂起的饥饿问题；

3. 计算集群或计算资源池应该做到负载均衡，同时应尽量使得执行任务的计算机空闲时间最短既实现数据传输和处理的同步；

4. 整个传输和处理系统要实现良好的鲁棒性：网络传输速率不能有剧烈且频繁的波动，集群存储器中的数据存储容量尽量保持在一个合适的区间；

5. 系统应具有可拓展能力，主要指的是当网络带宽资源、存储资源或计算资源的升级时，整个方案能够自动调整并适应，这要求我们的解决方案具有良好的动态调整性能和自适应性能。

1.3 本文的工作和文章结构

1.3.1 本文的工作

针对前面提出的问题和要求，本文的主要工作概括如下：

第一步：结合美国 LIGO 这个实际工程项目背景构建一套基于网格计算网络的数据流应用框架和机制包括寻求令人满意的任务管理和资源分配的解决方案；

第二步：进行计算机模拟仿真并以 Globus、Condor 等现有的网格计算平台工具搭建现实网络，进行实际测试；

第三步：得出结论，并对以后的工作进行展望。

1.3.2 文章的结构

本文其它章节安排如下：

在第 2 章，我们针对数据流应用的要求并结合 LIGO 工程项目背景，提出了一套基于网格计算网络的数据流应用框架和机制，在此基础上，我们运用遗传算法这种启发式的算法寻求令人满意的任务管理和资源分配的解决方案。

第 3 章是在 Matlab 模拟仿真实现我们提出的数据流应用框架和机制、任务管理和资源分配的解决方案，在动态的自启发算法的优化下得到仿真情况下各个时

刻的集群进行任务管理和资源分配的控制参数。另外我们以 Globus、Condor 等现有的网格计算平台工具搭建现实网络，并部署我们的数据流应用框架和机制、任务管理和资源分配的解决方案，同样在动态的自启发算法的优化下得到实际系统中各个时刻的集群进行任务管理和资源分配的控制参数。

第 4 章我们给出结论，分析所做工作的优缺点，并叙述未来可能的工作方向。

第2章 数据流应用框架和机制

2.1 系统框架

我们在 Globus 工具箱的基础上设计了一个数据流应用系统如下图 2.1 所示。

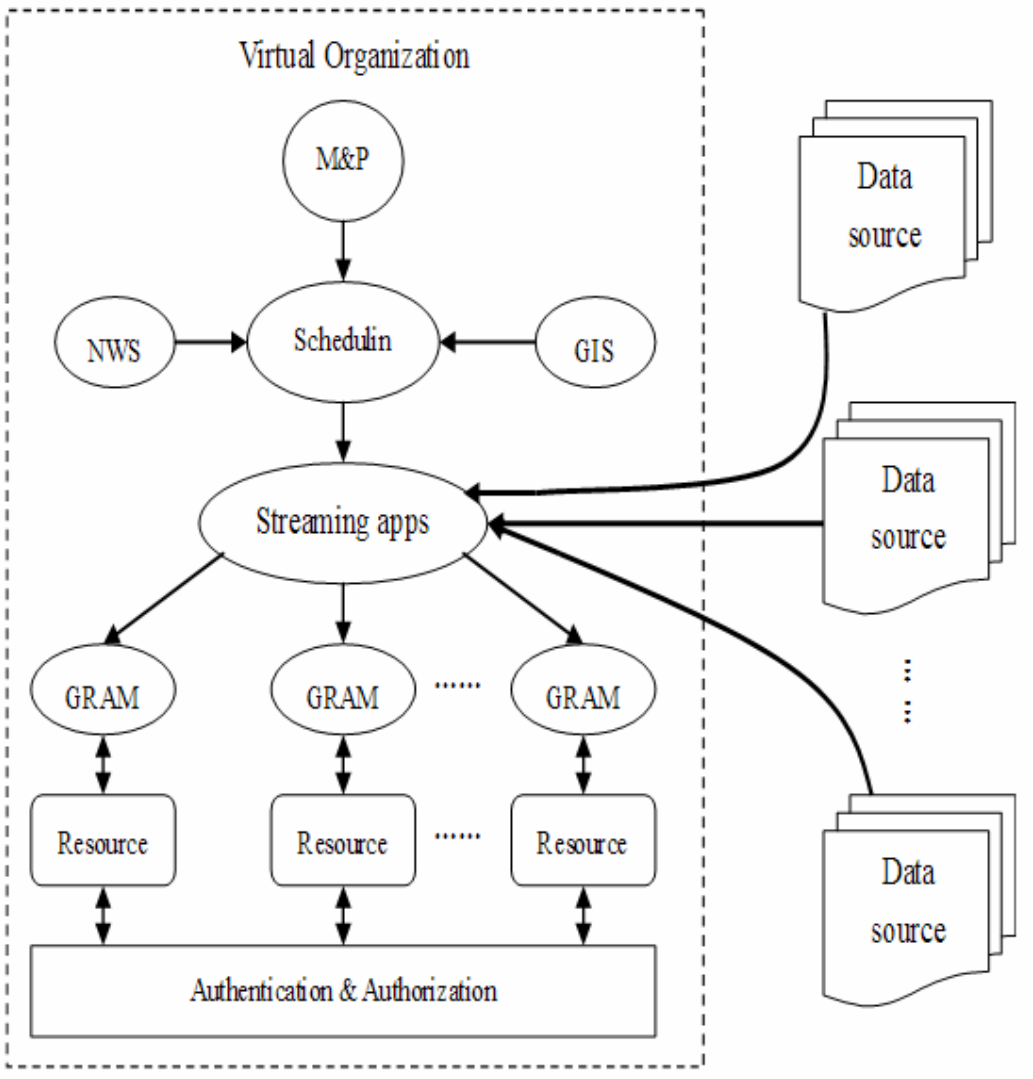

图2.1　系统框架总图

通过 Grid Security Infrastructure（GSI）的授权及认证，计算资源、存储资源和网络资源等网格资源形成了一个虚拟网格，彼此可以分享各自的资源并且可以合作为数据流应用提供支持。远端的数据源将数据以数据流的形式持续地传送过来并进行及时清除。图中的网格信息服务（GIS）和网络境况服务（NWS）提供网格资源的动态信息，包括硬件配置、CPU 负载和实时的网络带宽信息等，起到一个监测的作用。此外网格资源分配和管理器（GRAM）负责将任务提交到各个网格站点及执行资源的分配。最后 M&P 是为优化过程提供测量和预测功能[7]。

图 2.1 中许多功能模块都是可以由 Globus 平台实现的，本文工作的重点侧重在负责动态优化和任务、资源调度管理的 Scheduling 和 Streaming apps 模块，这部分模块与其它模块之间的关系又可描述如图 2.2 所示。

1. 客户工具（Client Tool）

这个工具是用户以 XML 标准格式提交他们各自应用程序的接口，包括应用程序诸如可执行性、对处理器的要求、对最小传输带宽和存储的要求和数据源等附加信息。同时它也可以监视已提交应用执行情况和整个网络资源利用状态。现今，这种工具是基于命令行方式的，将来可以运用用户图像接口（GUI）来实现。

2.管理器（Management ）

管理器接收用户提交的应用程序并把他们加入到任务队列中，等待调度器（Scheduler）的响应。除了为网格网络提供任务流之外，它还负责协调、监督和管理网络的安全性、资源检测、网格目录服务和网格中间件服务等。实现管理器的 Globus 工具箱部件包括 GRAM、GSI、Globus 二级缓存接口（GASS）、NWS 和 GIS 等。

3.调度器（Scheduler）

这是整个系统框架的核心部件，它的具体算法会在下面的第 3 章和第 4 章中详细讨论。它负责优化任务管理（包括任务选择、建立任务与处理端的映射等）、网络带宽和存储的分配等，得出适宜方案。具体执行交由下面的 Dispatcher 模块。

4. Dispatcher

它负责发送当前可执行程序的描述文件到指定到的处理端另外它时刻保持着与远端的 Wrapper 模块的通信。

5. Wrapper

这个模块将会解析 Dispatcher 发送的描述文件，初始化可执行程序以及依据优化得到的存储和带宽分配方案开始数据传送。同时，它也会将结果有 Dispatcher 模块反馈上去。

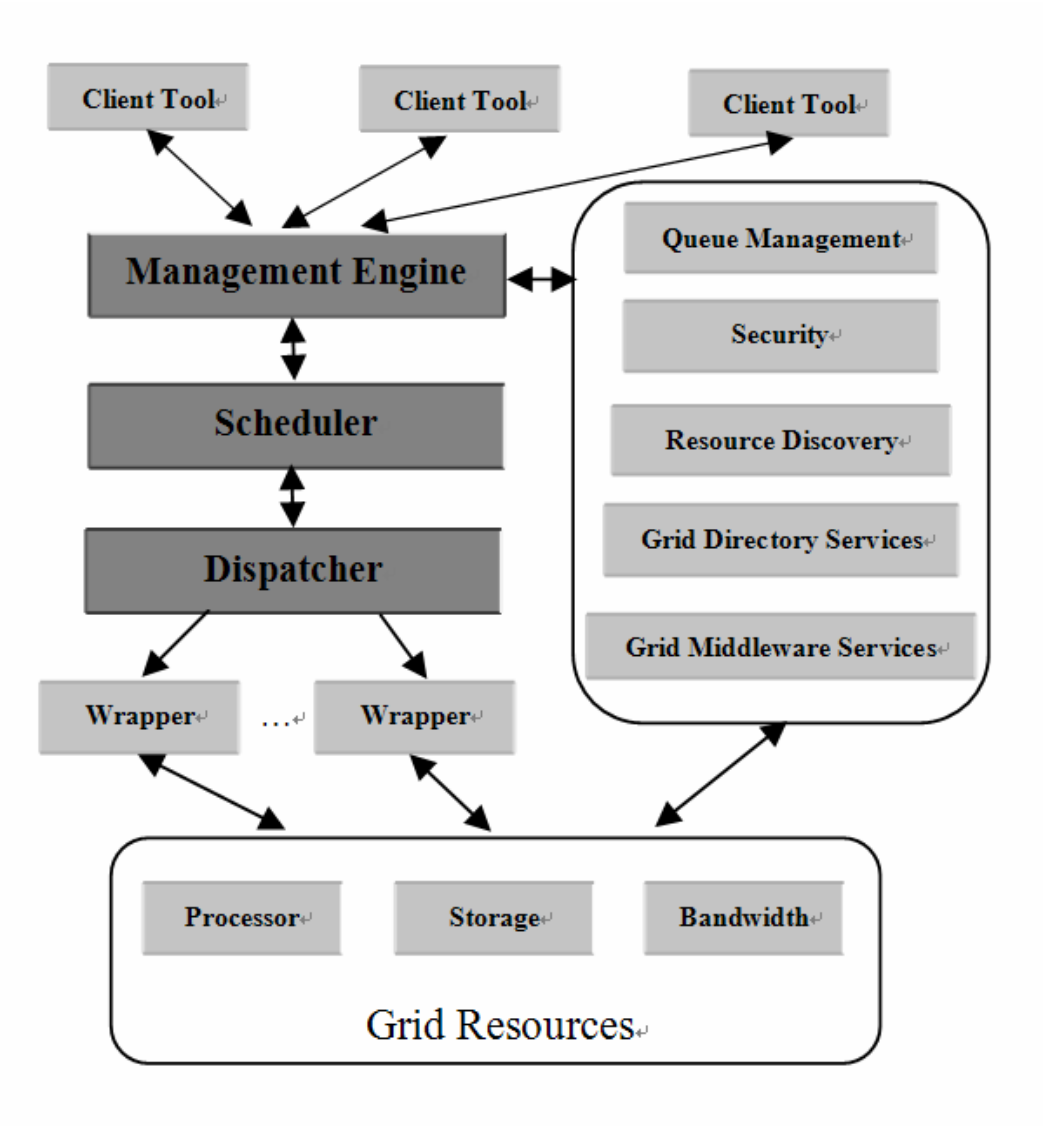

图2.2　任务、资源调度管理模块关系图

管理器的任务队列管理部分和调度器的资源优化环节这两部分可以构成一个闭环的控制回路，如图 2.3 所示。

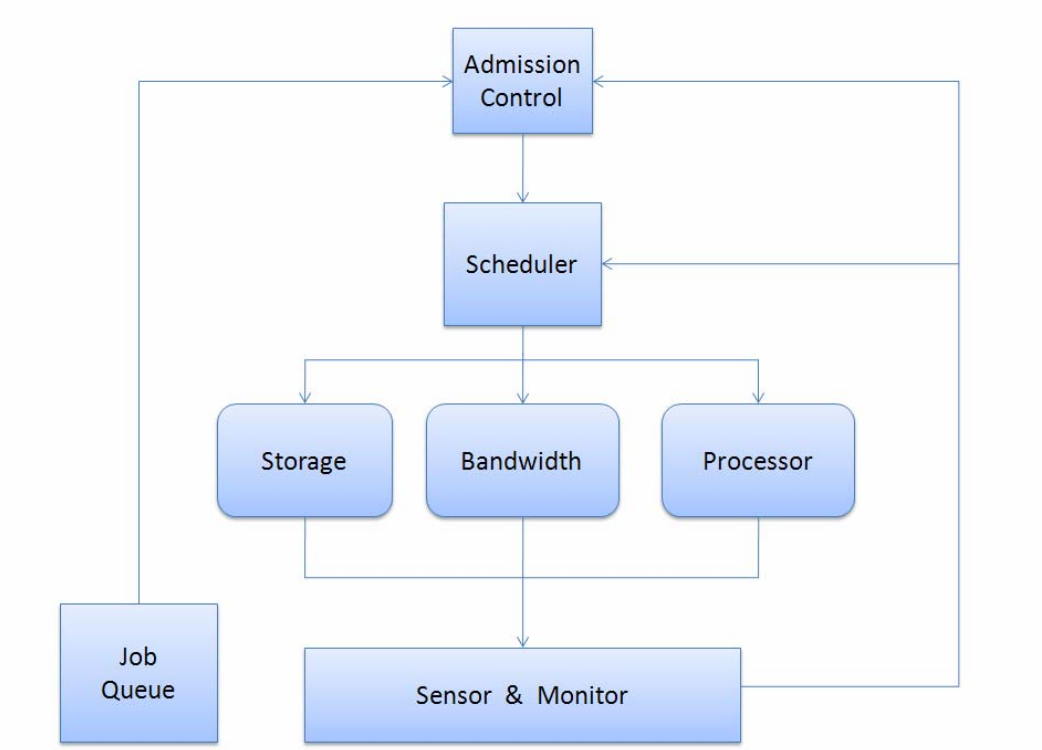

图2.3　任务管理和资源调度控制回路图

任务队列中的每个数据流应用被执行时都需要一定的资源，包括一个或更多的空闲处理终端（processor），一定量的带宽（bandwidth）和存储资源（storage）。在我们考虑的情况下，每个应用在其运行期间独占一个或多个处理终端。同时若给某个正在执行的任务分配了过少的传输带宽会导致处理终端大部分时间由于缺乏足够的数据供应而挂起，其处理能力没有得到充分利用；反之，过多的带宽分配会导致出现数据积压甚至造成存储溢出的情况。存储器保存从数据源传送过来待处理的数据，起到缓存的作用，同样太小的存储容量可能不能满足要求（因为极容易出现数据溢出），而太大的存储容量虽然有助于提高系统的鲁棒性但成本高而且无助于提高系统的传输和处理效率。

因此每个数据流应用对各种资源都有着自己不同的最小需求，而且由于资源池中可用的处理终端、存储容量和网络带宽等资源都是有限的，所以需要进行任务管理，决定是否接收一个应用的申请，或者是说怎样从候选的任务队列中启动合适的任务。我们会根据资源的匮乏度运用一些规则来拒绝应用的申请。而当计算池中有可用的空闲资源时，我们会运用一个启发式的选择算法从候选队列中挑出最佳选择，将其加入到正在执行的任务中去。

调度器负责为正在执行的任务分配存储和带宽，这是一个动态的进化过程，基本的原则是尽可能地充分利用带宽和存储同时留下一部分资源为新来的任务做准备。由于环境的动态变化，任何一个事先定好的调度方案都不可能是长时间适宜的，因此我们必须定期性的或者事件触发式的进行优化，得出适宜当前的情况的调度方案。

2.2 任务管理

2.2.1 任务管理工作流程

任务管理是基于触发模式工作的，当出现如下情形时，它将被启动并进行管理调度工作：

1. 数据流应用系统初始化时，任务管理器要从最初待处理的任务队列中优选出一组任务组合并将该组合与计算资源池中的处理终端建立起映射关系；

2. 计时器达到预定的工作周期，任务管理器也被启动。这主要是考虑到在一段时间内，可能会出现新的任务加入、正在执行的任务工作终了或外界环境的改变如计算资源池中处理终端的添减等因素，这样我们就有必要重新进行新的任务管理调度，才能保证系统始终处于一种高效运行的状态。

任务管理的核心工作是挑选任务、建立任务与处理终端的映射，与资源调度器一起协同工作，力争在满足该网络限制性条件下通过优化配置网络中的各种资源，使得所有任务能在最短时间内被执行完毕，任务管理主要配置的计算资源池中的计算资源。被任务管理器挑选出的任务加入到执行任务队列中，其它的任务仍置于等待任务队列中，为了避免出现任务永远得不到调用的情形（由于该任务权限低），我们可以根据任务的等待时间定期调整等待任务的权限。此外任务管理器进行建立任务与处理终端的映射决策时，决策装置应该得到网络资源的使用信息，这些信息可以由网络资源检测器（Network Resource Monitor）提供。任务管理工作流程图图 2.4。

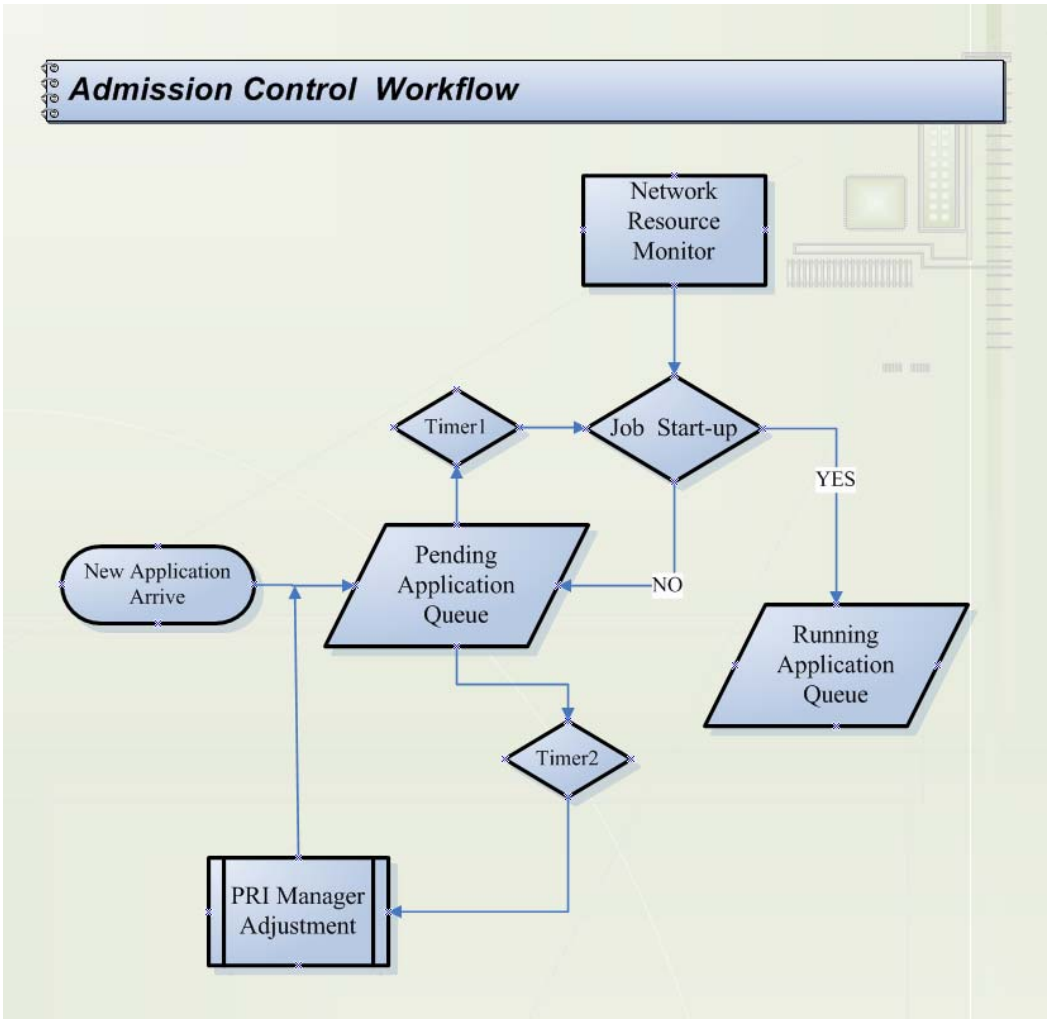

图2.4　任务管理工作流程图

2.2.2 任务管理的优化机制和算法

如图 2.4 所示，任务管理优化环节有两个：一个是任务决策模块（Job Start-up）根据当前网络资源使用信息从候选任务队列（Pending Application Queue）中挑选任务并将其与数据处理终端建立映射；二是任务优先级调整模块（PRI Manager Adjustment）根据候选任务队列中任务等待时间调整它们的优先级。

若在某次任务管理优化过程中，候选任务队列中的任务数目是 m，计算资源池中空闲的处理终端数目是 n。为简便考虑，我们设定处理终端与任务之间的映射关系是一对一的而且映射建立后会固定下来直到任务终结。那么该次优化空间（映射关系集合）的大小将是

$$\sum_{p=0}^{n} A_m^p \,(m \geq n) \qquad or \qquad \sum_{p=0}^{m} A_n^p \,(n > m) \tag{2-1}$$

优化算法的搜索空间将随着 m 、n 的增大而急剧增大，如 $m=30, n=10$，则优化空间的数量级为 10^{14}。因此不可能进行进行全局穷尽，必须制定一些优化机制来简化搜索算法。我们再考虑其它一些优化机制：

先来先服务机制（FCFS）——一旦有空闲的资源就启动候选队列中最早到达的队列：这项原则目的不在于寻求全局最优解，我们所需要的是能够在一个短时间找到满意解即可，所以先来先服务是一种简便而有效的方式。它能够快速地将任务推到计算池中，最大程度地避免计算资源空闲。这种配置存在的不足是没有考虑到处理终端与任务之间的相互适应关系，若任务被安置于不适当的终端上进行处理时，计算资源的效能得不到很好地发挥。

优先级机制（Priority）——网络资源出现空闲时，将候选队列中优先级最高的任务推入计算池：这项原则能够很好地简化搜索，而且可以兼顾任务的重要性，但同样没有考虑到处理终端与任务之间的相互适应关系。

由上可知，我们很难找到一种简单的优化机制能够完全符合数据流应用系统的要求，因此我们考虑基于一些简单的机制设计出一套综合的优化机制，其中我们主要引入了先来先服务、优先级、尝试策略和动态调整等原则，具体过程如下：启动任务管理器后，权衡先来先服务和优先级，得出若干候选的任务优化组合（具体实现可以依据先来先服务原则，根据候选队列中各任务的等待时间来调整优先级，由队列中优先级较高的任务进行组合排列得出若干优化组合）。将候选的任务优化组合以此部署到空闲的计算资源上运行一段时间，得出一个网络适应值，最后选择最大网络适应值对应的任务优化组合作为任务管理实施方案。动态调整体现在任务管理是一个动态过程，而且候选队列的任务优先级也是动态调整的。这套优化机制极大地缩小了优化空间的大小，兼顾任务的重要性和队列等待时间，同时能够考虑到处理终端与任务之间的相互适应关系（通过适应值可以反映）。这套机制的主要缺点是需要调配若干权重参量，这个工作往往是比较复杂的，但这是一次性的工作，故在整个优化过程中所占用的时间开销比例不会很大。

2.3 资源调度

2.3.1 资源调度工作流程

任务管理器建立起任务与处理终端的映射之后，任务开始在处理终端上执行，并且任务所需要的数据由数据产生端流水式地传送到计算池的存储器

上，而处理终端从存储器成批次地提取数据进行分析处理同时将及时地清除存储器上的冗余数据。行之有效地为正在运行的任务队列安排网络带宽和存储容量等网络资源使得我们的数据流应用系统处于一种高效能的工作状态将成为资源调度的优化目标，具体的衡量标准是让正在执行任务的处理终端由于数据供应不足造成的计算资源空闲之和最小。

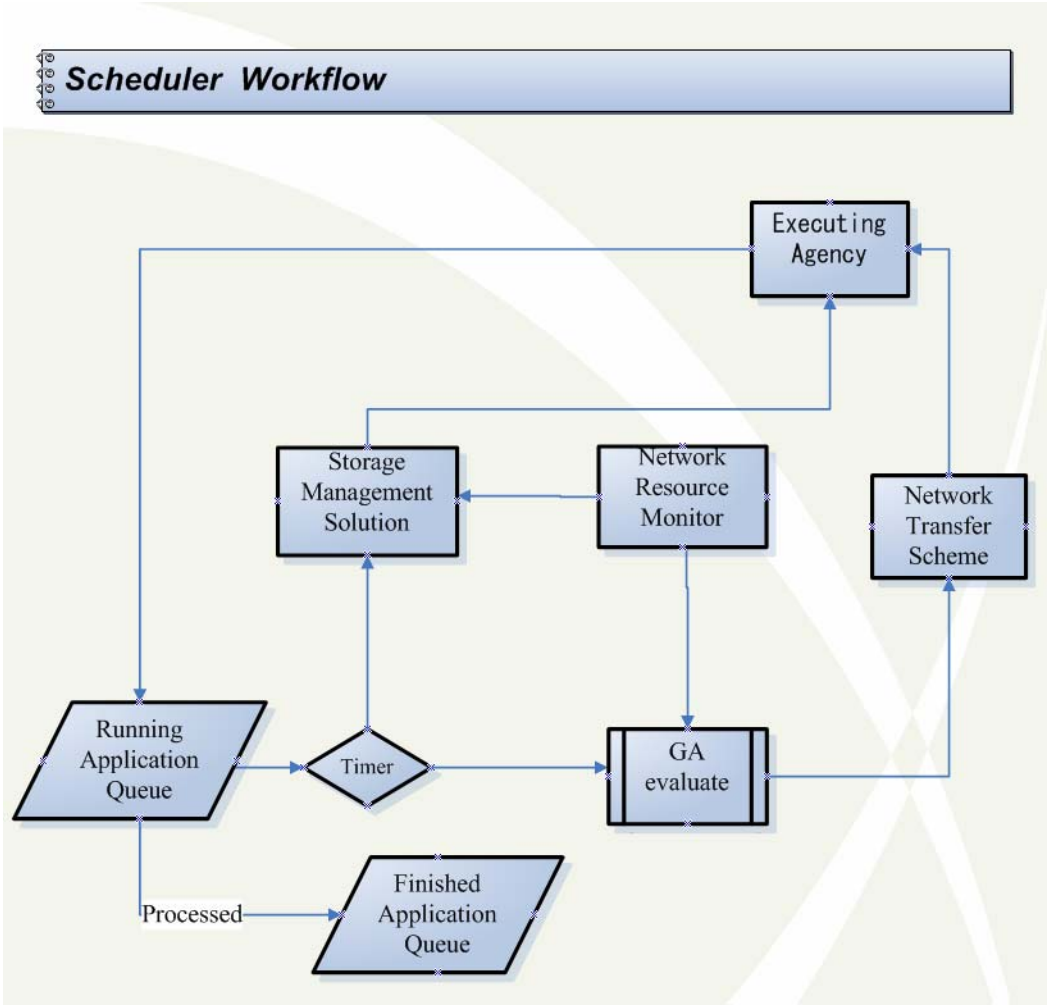

图2.5　资源调度工作流程图

资源调度是被定时触发的，这样也可以保证资源调度方案能够得到及时更新。任务管理器根据网络资源检测器收集的网络资源信息，得出存储管理方案（Storage Management Solution）和网络传输方案（Network Transfer Scheme），其中后者是由遗传算法优化出来的，前者则是通过反馈调节，最后交由执行机构（Executing Agency）去实施。

任务执行完毕后推出执行任务队列进入已完结任务队列（Finished Application Queue）。

存储管理方案主要是通过网络资源检测器构成的闭环反馈来调节，我们的目标是保证任务所在的处理终端的计算资源尽量不出现空闲，因此我们必须尽量保证每个任务对应的存储时刻都存有数据以备处理终端提取，要做到这点单靠分配存储容量办不到的，存储管理必须协同网络传输一起工作。我们的资源调度其实就是尽量保证每个正在被执行的任务的网络传输速度与处理速度保持一致，最理想的情况就是数据用多少就传多少。但是由于处理终端的处理速度是时刻再变化的而我们又无法提前预知（我们可以根据其历史记录来作预期，但还是与实际情况是有出入的），而且由于网络总带宽的限制无法保障处理终端工作负荷顶峰的需求，所以我们引入存储器起到数据缓存的作用。综上可知，存储管理和网络传输两者是相辅相成的，两者都处于由网络资源检测器构成的闭环反馈回路中。

2.3.2 存储管理的优化机制和算法

最简单的存储管理机制是将存储容量平均分配给正在执行的各个任务，但这种机制是有很多缺陷的。比如说有些处理终端工作起来比较稳定，其处理速度变化不大，而另一些处理终端的处理速度变化较剧烈，这时平均分配就不是一种好的策略，另外一个刚被启动的任务与一个已经存储相当数据的任务分配同样的存储容量也是一件应该商榷的事情。诸如此类，在平均分配的前提下，我们希望加入一些其它的控制机制来实现更好的控制效果，包括最小存储、动态调整和保留裕量等。

因为处理终端从存储器提取数据是成批次以块状方式进行（这可以被理解，比如图像处理应用处理的对象是图片，那么我们不能一次只提出半张图片信息进行处理，我们只能以单张图像信息作为数据提取的基本单位），存储器为每个正在执行任务的处理终端分配的存储容量至少应当大于该任务应用单次提取数据块（一般将最小存储设为任务应用单次提取数据块大小的整数倍），同时也应当设定若存储器中存储的数据小于单次提取数据块的大小，则处理终端不能进行数据提取，处理终端闲置一段时间。

动态调整是反馈控制的实现手段，我们可以认为在受限的总存储容量限制下，存储器中存储的数据越多越有利于系统的健壮性，所以我们应该减少空置的存储器容量（这不同于下面提及的特意留出的存储裕量，下面将会详

细讨论），具体的办法是减少分配给那些闲置率（存储的数据与分配的存储容量的比值）高的存储容量，增多分配给那些闲置率低的存储容量。这样的化存储器整体的闲置率会降低，有利于保证系统的稳定性。

往后一个存储策略是在存储器中适度地保留存储裕量，这主要是考虑到为可能新加入的任务保留存储资源，否则若整个存储器都已经被当前运行的任务存储的数据占满，那么新的任务的数据传输就无法进行。我们预留了一定的存储裕量可以满足一两个新加入任务的存储需要，它们加入后再通过动态调整使得存储器的存储裕量又返回原来的水平，为下一个的任务加入做准备。

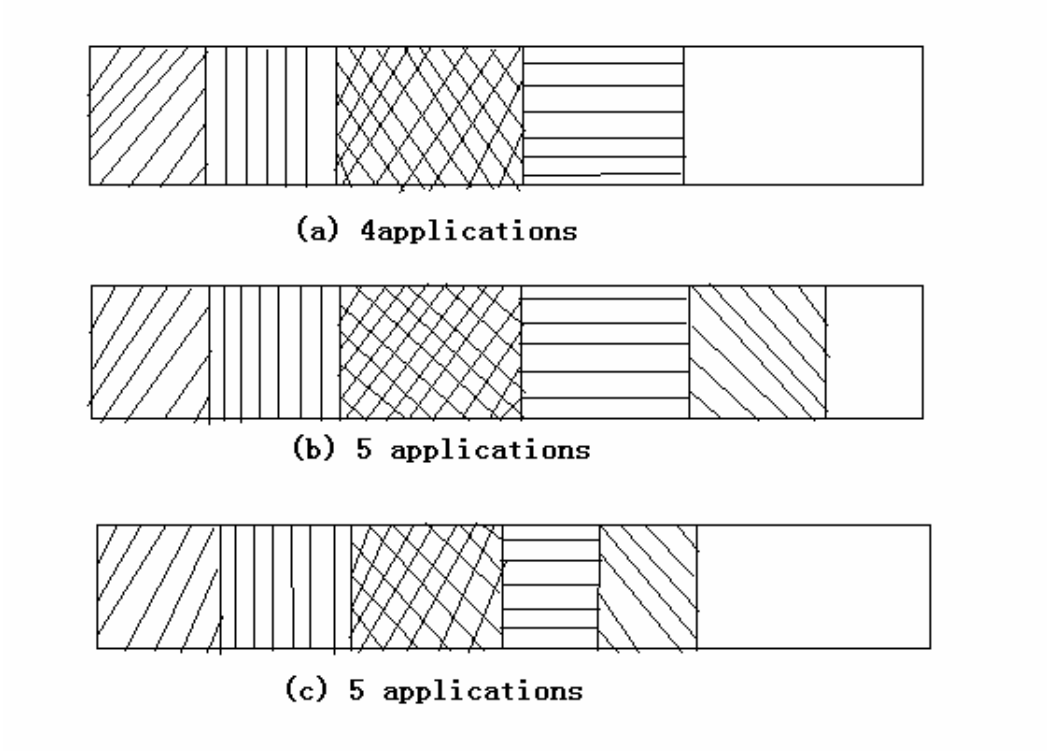

图2.6　存储管理示意图

图 2.6（a）显示存储器当当前运行的 4 个执行任务分配了存储容量同时整个存储器留有存储裕量（空白处）；图 2.6（b）显示新加入一个任务后存储裕量减少了一部分，分配给新加入的任务；图 2.6（c）显示经过一段时间各任务存储容量的调整后，存储器的存储裕量恢复到原有水平。

2.3.3 网络传输的优化机制和算法

网络传输控制在数据流应用系统中起着决定性的作用，带宽分配关系到能否时刻提供适量的数据以确保计算池中处理终端上的任务被连续地执行。数据传输过少会造成处理终端无法从存储器中提取数据（存储器中存储的数据不足）而处于空闲状态，相反数据传输过多会造成存储器数据溢出、数据丢失带来数据重传等一系列的难题。虽然由于一些无法预知的因素使得我们无法达到理想的“要多少数据就传多少数据”的状态，但我们仍然寄希望于通过增加存储器数据缓存这个环节达到近似于上述理想状态：确保存储器不出现数据溢出的问题同时尽量使得正在执行任务的处理终端的闲置时间最短。

我们从数据源通过网路连接单个计算资源池组成的简单网络传输系统（这样的话，接入计算池的是单条主干网线，如下图所示）来考虑。

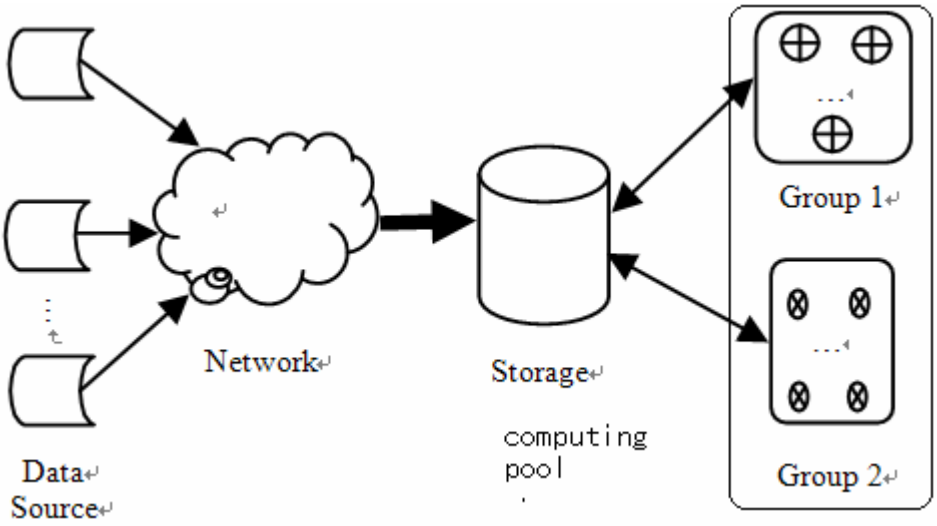

图2.7　网络传输系统示意图

我们记主干线总带宽为 I，它是受限的并且为各个数据流应用所共享。单个任务应用标记为 s，它从属于一个大的任务集合 S。每个任务都有一个分配的带宽 x_s，且 $x_s \in X_s$。当任务 s 的数据传输速率为 v_s 时，它有一个效用评价函数记为 $U_s(v_s)$。我们在兼管公平原则的前提下尽量使得所有任务效用评价值之和最大，这个问题可以描述如下：

$P.$

$$\max \sum_{s \in S} U_s(v_s) \qquad (2\text{-}2)$$

s.t.

$$\sum_{s\in S} v_s \le I \quad and \quad v_s \le x_s \tag{2-3}$$

$$x_s \in X_s \quad and \quad \sum_{s\in S} x_s \le I \tag{2-4}$$

由于存储容量的限制，可能有的任务由于前一段时间网路传输过快，存储的数据已达到一定界限，这时我们将其对应任务分配的网络带宽 x_s 置为 0（易知此时 v_s 也为 0），此时任务 s 隶属于 S 的一个子集 S_A，即

$$x_s = v_s = 0 \quad s \in S_A \subset S \tag{2-5}$$

公式

我们运用网络传输模型中的加性增，乘性减原则来进行拥塞控制，迭代公式如下所示：

$$x_s^{(k+1)} = 0 \quad \forall s \in S_A \tag{2-6}$$

$$x_s^{(k+1)} = \begin{cases} [x_s^k + \alpha\ U^{'}(v_s^{(k)})] & if \sum_{s\in S} v_s \le \rho I \\ \beta x_s^{(k)} & if \sum_{s\in S} v_s \ge \rho I \end{cases} \tag{2-7}$$

注：$$U^{'}(x_s^{(k)}) = \frac{\partial U(.)}{\partial x_s^{(k)}} \tag{2-8}$$

$$U(.) = \sum_{s\in S_B} U_s(v_s^{(k)}) \qquad S_B = S - S_A \tag{2-9}$$

在以上的传输模型当中，有三个参量 $\alpha, \beta, \rho \in (0,1)$ 需要确定，我们通过遗传进化算法进行优化得出其结果。

第3章 数据流应用系统测试和结果分析

3.1 计算机模拟仿真测试

我们在 Matlab2007 工作平台上模拟实现了第 2 章中的数据流应用系统，设计出了若干实验数据对其工作流程进行测试，我们将根据系统工作运行状况对系统的性能进行评估，主要评价指标包括系统吞吐量（单位时间计算资源池处理的数据信息量）、存储资源和网络带宽资源的有效利用率、任务队列的平均等待时间等。

3.1.1 数据流应用系统的模拟实现框架结构

我们将图 2.1 中描述的数据流应用系统框架简化成如下图所示的线性型结构，能够更好地符合程序语言的编写习惯，理顺各个功能模块间的关系，简化编程实现的复杂性并利于调试。

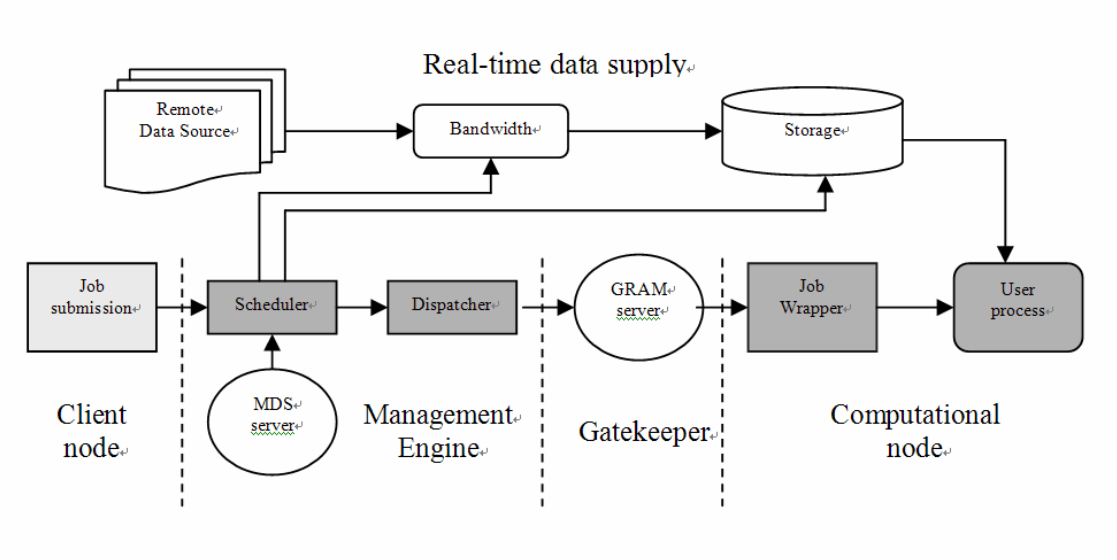

图3.1　模拟仿真系统框架图

第一阶段客户节点（Client node）和远端数据源（Remote Data Source）属于预先设定实验数据范畴，比如说任务提交（Job submission）中描述的

每个任务提交系统的时刻、任务的初始优先级、对应数据源中数据信息的大小等信息。

第二阶段调度管理器（Management Engine）是数据流应用系统的核心部分，调度器（Scheduler）根据网络检测器（MDS server）提供的当前网络资源分布状况和任务管理队列的情况进行优化进化，制定调度方案指导带宽资源（Bandwidth）和存储资源（Storage）的分配。

第三阶段网络关口（Gatekeeper）主要考虑网络接口的安全性，这在实际的网络环境中是必须加以考虑的（Globus 工具能够为网格平台提供安全认证功能），我们在计算机模拟仿真阶段不考虑这部分的实现。

第四阶段是计算资源节点，我们用均匀分布的随机数表示计算处理速度（能够反映处理速度变化的未可预知性），其中不同的平均值和分布区间表示不同计算处理终端的性能差异。

3.1.2 进行测评的实验数据

实验数据有若干组，每组数据包括一个任务的任务提交系统的时刻（t_s）、任务的初始优先级（w_{s0}）、对应数据源中数据信息（a_s）的大小、对应计算资源池中数据终端的类型（pg）及任务在存储器中的最小存储（st_s）：

表3.1　实验数据表

number	st_s	pg	w_{s0}	a_s	t_s
1	8	2	1	10100	0
2	9	3	4	10500	0
3	10	3	5	6000	0
4	10	1	5	5900	0
5	8	2	1	4400	0
6	5	2	2	10100	0
7	5	1	2	4700	0
8	6	2	5	2200	0
9	10	1	3	8900	0
10	6	3	5	4900	0
11	9	1	1	3500	79
12	6	2	2	5100	229

number	st_s	*pg*	w_{s0}	a_s	t_s
13	10	1	1	2000	261
14	7	2	1	2400	363
15	6	1	5	10500	470
16	6	2	3	10600	489
17	8	3	3	6800	645
18	7	3	1	1600	764
19	7	2	5	3400	769
20	9	1	4	4600	1243
21	8	1	2	9300	1247
22	8	3	3	1200	1496
23	10	1	3	1500	1556
24	6	3	1	2700	1636
25	9	2	2	7500	1694
26	9	3	1	8400	1846
27	7	1	1	7500	1917
28	8	2	2	5600	1948
29	5	1	3	6500	2540
30	5	3	1	4000	2911

由上表可知，在开始时刻候选任务队列中有 10 个等待任务，在此后的 3000s 时间内其它的任务以一个马尔可夫序列依次到达。任务的最小存储、初始优先级、对应数据源中数据信息的大小及对应计算资源池中数据终端的类型都是通过均匀分布的随机数产生的。

此外计算资源池中的数据处理终端分为 3 种类型（即 pg），每种类型的计算机数目分别是 5、4、7。

3.1.3 调度管理程序流程图

调度管理作为数据流应用系统的核心模块，依据反馈调节、遗传进化和迭代调整等方法试图寻找能够综合利用计算资源、存储资源和网络带宽资源等优化解决方案。程序实现模块的具体流程图如图 3.2。

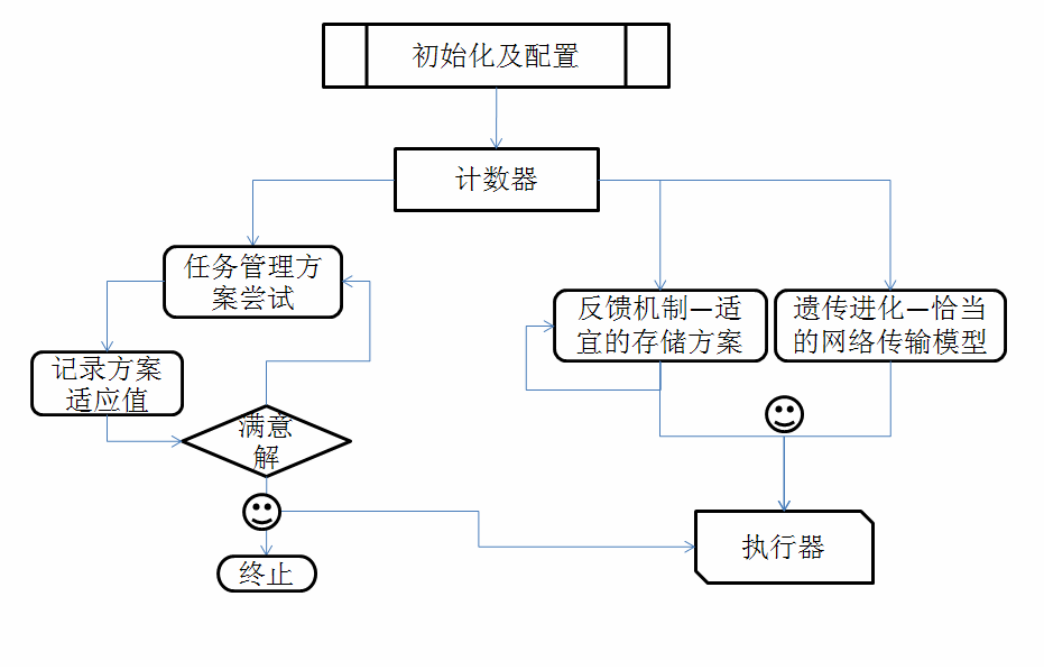

图3.2　调度管理模块程序流程图

本文工作中核心算法——遗传进化算法的各项参数设置如下。

设计遗传算子：

选择运算使用比例选择算子

交叉运算使用单点交叉算子

变异运算使用基本位变异算子。

确定遗传算法的运行参数

群体大小：$A=60$

终止代数：$S=100$

交叉概率：$p_c=0.2$

变异概率：$p_m=0.01$

3.1.4 测试结果及分析

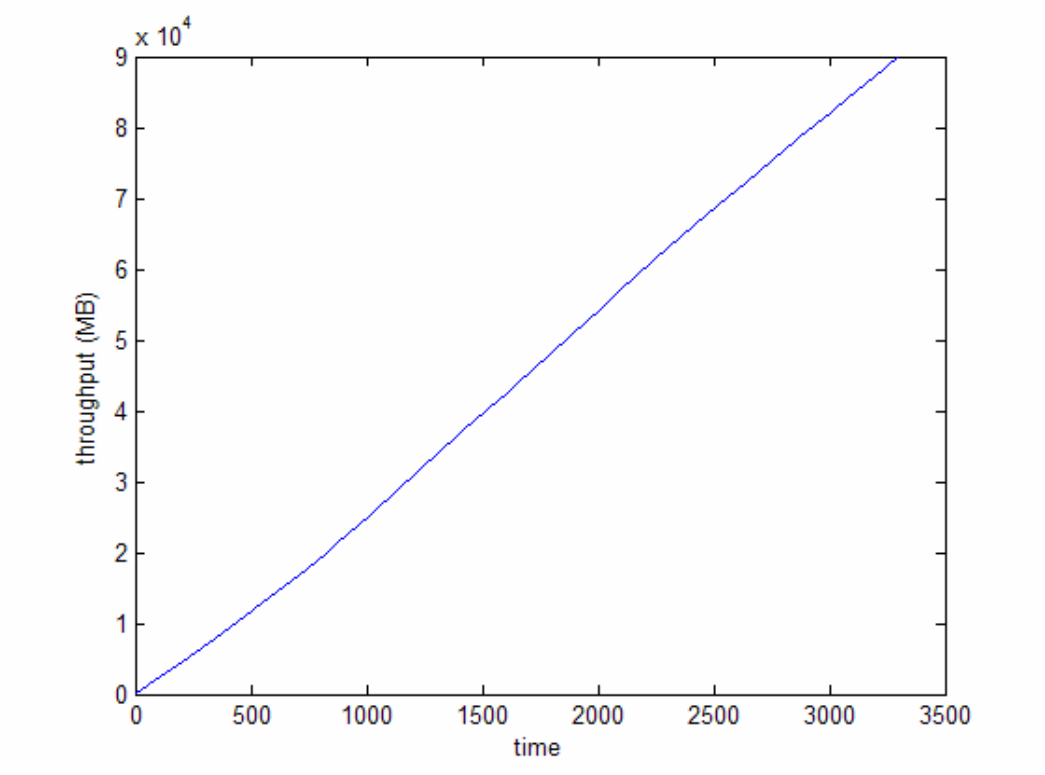

图3.3　数据流应用系统数据吞吐量示意图

首先我们观测系统的数据吞吐量变化情况（见上图图 3.3），我们可以发现在不断有任务加入的情况下吞吐量基本上以线性速度增长，表明系统基本上以恒速处理任务，具有良好的稳定性。

接着我们观测系统存储资源和网络带宽资源的利用情况（分别见下图图 3.4 和图 3.5）。

由图 3.4 可知，在观测时段内，存储没有出现存储溢出的现象（存储器的存储容量为 180Mb），同时可以看出存储器在整个阶段都处在一个相对较高的工作负荷状态而且存储裕量的控制也是比较合理和适宜：我们设定保留整个存储容量的 20％作为存储裕量，而存储器中实际存储的数据平均在 140Mb 左右，占存储容量的 78％，所以实际上存储闲置率只有 2％到 3％。

此外我们还可以观测某个具体任务在其执行阶段的数据存储的情况，如下图所示，第 7 个任务从初始时刻（零时刻）到达并执行到任务终结（3200s 左右）其数据存储保持在一个相对比较稳定的水平，而且绝大部分时间内数据存储量都在其要求的最小存储（5Mb）之上，这表明执行该任务的计算资

源不会由于数据传输不足而闲置，故我们的数据流应用系统的存储控制和传输控制的配合工作是非常协调有效的。

网络带宽方面的利用情况也是令人满意的，如图 3.6 所示整个观测期间系统的占用的总带宽没有超过带宽容量（30Mb），大部分时间内带宽的利用率同样也处在一个较高的水平上。

最后我们看一下整个任务队列的整体执行情况（如下图图 3.7 所示），观测任务队列的等待时间，可以发现除为数不多的几个任务的等待时间较长外（这是由于这些任务的初始优先级低造成的，比如第 12、14、21 和 27 号任务的初始优先级分别是 2、1、2、1 而最高优先级是 5），整个任务队列的平均等待时间还是令人满意的。

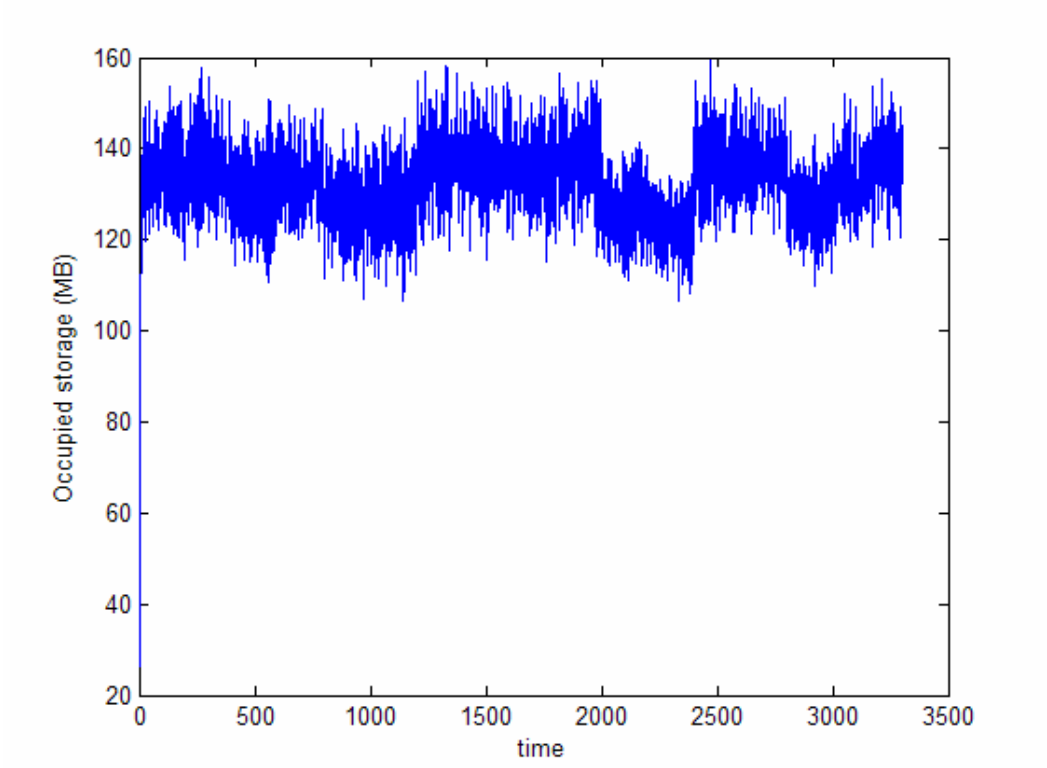

图3.4　数据流应用系统存储资源使用图

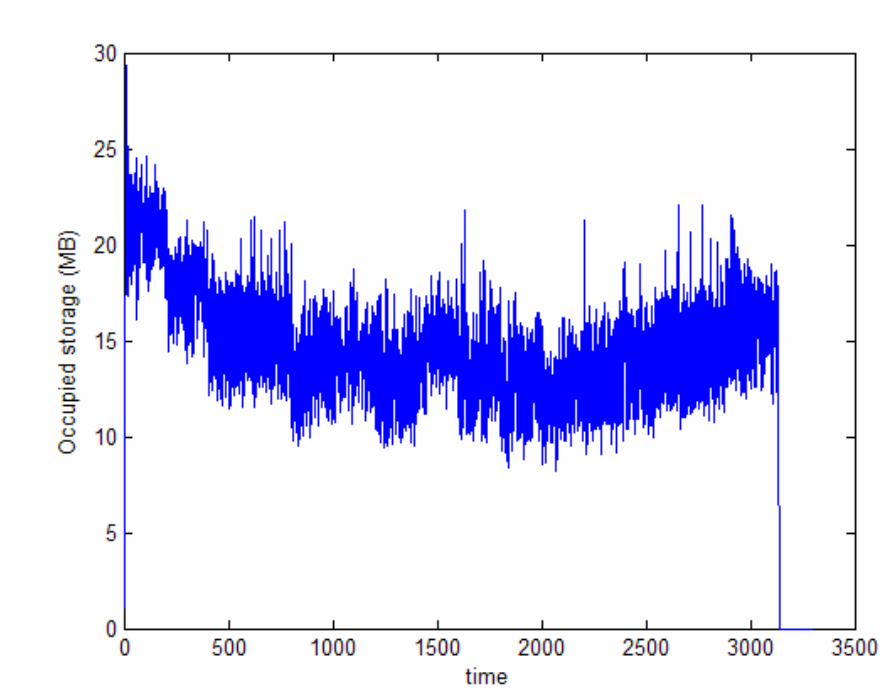

图3.5　第7个任务数据存储情况图

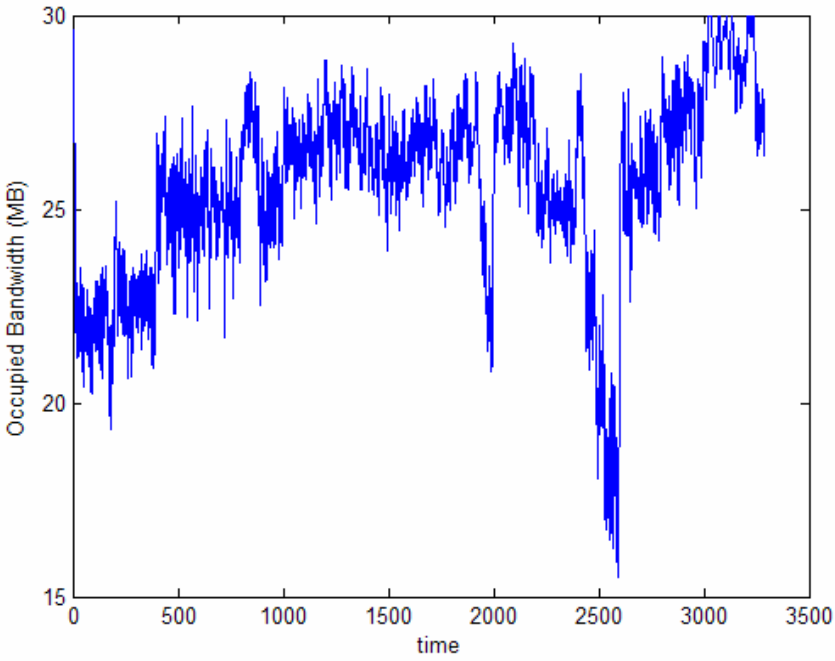

图3.6　数据流应用系统网络带宽使用图

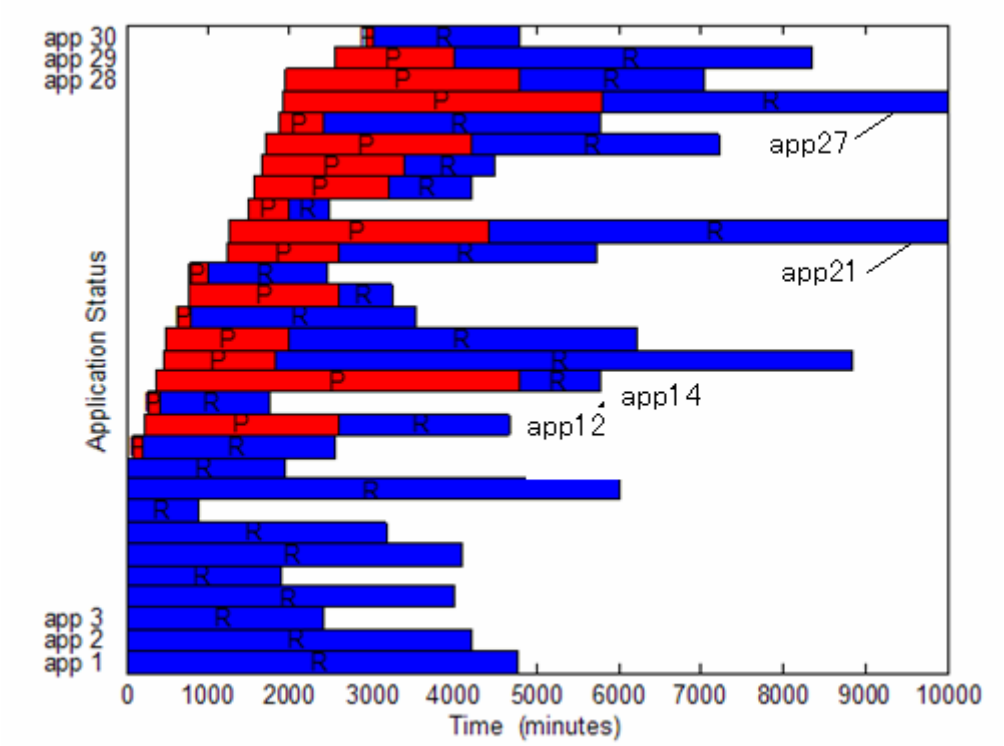

图3.7　整个任务队列的整体执行示意图

注：图 3.7 中红色标 P 字母的区域表示任务在此段时间处于候选队列中等待被执行，蓝色标 R 字母的区域表示任务正在执行。

3.2 搭建 Condor 计算网络测试

测试环境：清华大学信息研究院未来信息技术研究中心 100M 以太网络

测试平台：

硬件：Intel（R）Pentium（R）Dual E2140　1.60GHz　1GB 内存

Intel（R）Pentium（R）D　2.80GHz　2GB 内存

Intel（R）Core（TM）2 Duo CPU　2.10GHz　2GB 内存

软件：Linux Fedora Core7

Globus Toolkit4.0

Condor－6.8.6

软件下载地址：http://fedoraproject.org/

http://vdt.cs.wisc.edu/vdt_181_cache:Condor

http://vdt.cs.wisc.edu/vdt_181_cache:Globus

http://vdt.cs.wisc.edu/vdt_181_cache:GSIOpenSSH

3.2.1 Condor 和 Condor 环境[8]

Condor 是一个用来管理计算敏感作业的批处理队列系统，而且能够访问分布式资源来获得更多的计算能力。Condor 通过诸如 classAds 和 matchmaking 之类的创新技术提供了调度功能。与其他批处理队列系统一样，提交给 Condor 的作业应该作为无人参与的后台任务运行。

要从网格环境中获得最大的收益，网格应用程序应该可以被划分成很多独立的任务。与 Globus 有关的作业提交、监视和控制机制在上面都简要进行了介绍。下面让我们来介绍一下 Condor 中的特性。作业可以使用 condor_ submit 命令和对作业的描述以及提交描述文件中的相关需求提交给 Condor。一旦作业被提交之后，可以使用 condor_ q 命令进行监视。condor_ q 命令返回队列中所有作业的列表，以及作业的详细信息，例如作业提交的日期和时间、作业状态以及优先级。当作业完成时，用户会收到通知，并得到有关作业执行的统计信息，例如所占用的 CPU 时间或作业的输入和所生成的输出。

在 Condor 中有很多运行环境可以用来支持不同类型的作业。运行环境被称为 Condor 环境。Condor 环境是在上面提到的提交描述文件中指定的。最通用的作业环境是标准环境。然而，还有一些环境可以支持并行虚拟机（Parallel Virtual Machine）、消息传递接口（Message Passing Interface）、Java® 虚拟机、网格计算，还有一个用于特殊调度场景的环境。本文的重点是关于网格环境的。

在所有的计算资源都运行 Condor 时，Condor 可以提供一个相当广泛的网格计算环境。然而，如果网格是通过加入不同管理域中的几个现有集群来创建的，那么这种方法可能是不可行的。不同的集群可能有一些现有的资源管理器，它们可以对集群的资源进行优化。集群可能在不同的域中，运行具有不同本地调度策略的不同资源管理器。在对部门的集群进行集成来构造网格时通常会出现这种情况。网格通常是由很多不同类型的硬件和软件构成的。Globus Toolkit 提供了一个基础设施来在这些异构分布式资源之间进行资源的共享。下面让我们讨论一下 Condor 如何在这个 Globus 基础设施的基础上提供高级的服务。

Condor 的网格特性是通过使用 Condor 的网格环境来启用的。在使用 Condor 网格环境时，Condor 可以通过 Globus Toolkit 将作业提交到远程网格资源上。Globus Toolkit 提供了网格基础设施，包括分布式和多域环境中的安全性和资源访问。

结合使用 Condor 的特性与 Globus 控制的资源一起进行作业提交、监视和控制就称为 Condor-G。Condor-G 使用 Condor 的作业管理特性，以及 Globus Toolkit 的安全性和资源访问特性。Toolkit 提供了身份验证机制、与远程资源进行数据传输的能力以及远程执行环境。Condor 通过在提供监视和控制功能的同时提供通知、容错和凭证管理功能，从而对作业的提交和管理进行了简化。

Condor 可以帮助在用户作业执行时管理用户的凭证或代理，防止它们在作业运行时过期。Condor 可以在作业完成或失败时通知用户，这样用户就可以执行适当的操作了。Condor-G 还提供了一个容错环境。如果一台机器出现了故障，或者一个作业在某台机器上失败了，那么这个作业就会返回这个队列。Condor 会使用另外一个资源来匹配这个作业，并自动重新提交这个作业。

3.2.2 Condor 网络数据流实验

在上文中提到的 LIGO 项目中，数据处理中心的数据检测工具（the Data Monitoring Toolkit，DMT）将两个 LIGO 观测所传输过来的流式数据快速读取并作处理——计算两批数据中脉冲信号的相似性系数 r。如果在两批数据中同时出现了相似的信号，则极有可能是我们需要探测的引力波。数据流都是有数据文件构成的，每个数据文件具有相同的格式且都是探测 16s 内的信号。

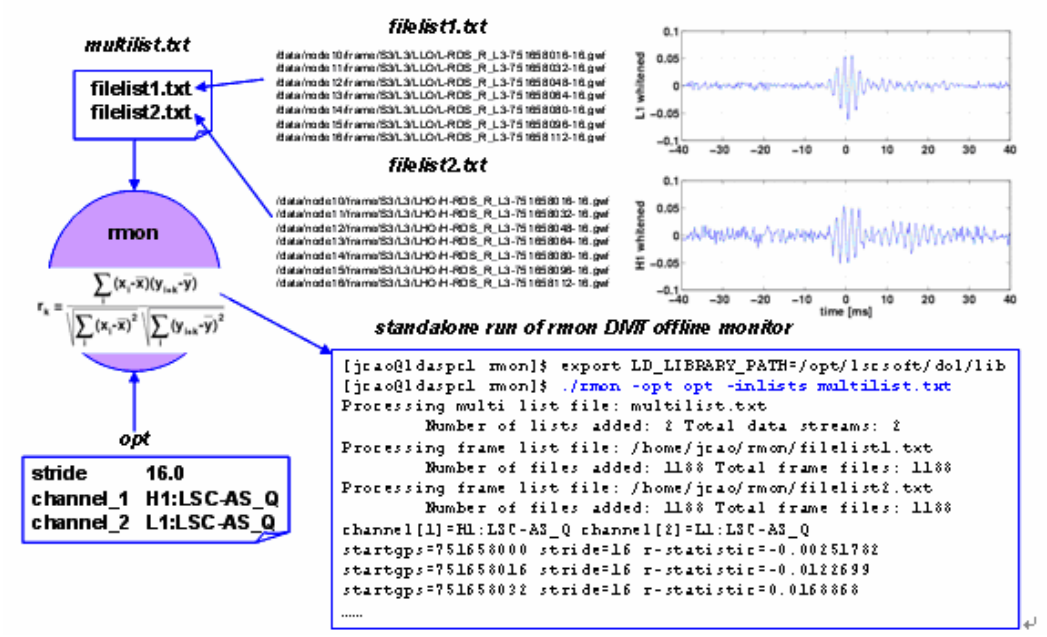

图3.8 LIGO中DMT的工作流程图

相似性系数 r 定义如下：

$$r=\frac{\sqrt{\sum_{j=0}^{n-1}(x_{1j}-\bar{x_1})(x_{2j}-\bar{x_2})}}{\sqrt{\sum_{j=0}^{n-1}(x_{1j}-\bar{x_1})^2}\sqrt{\sum_{j=0}^{n-1}(x_{2j}-\bar{x_2})^2}} \quad (3\text{-}1)$$

其中

$$\bar{x_i}=\frac{1}{n}\sum_{j=0}^{n-1}x_j,\quad i=1,2 \quad (3\text{-}2)$$

在我们的实验中，Condor 网络中的一台计算机硬盘中存储了共计 1188 对数据文件的原始数据集，而数据处理程序运行同一实验网络中的另一台计算机上。我们使用 GridFTP（参数设为－ f ）进行数据传输，同时采用我们设计的数据流应用系统进行调度管理。

3.2.3 测试结果及分析

我们进行实验时数据传输速度基本上是恒速（因为数据产生端和处理端都位于同一实验室的局域网内，网络基本没有拥塞），每次处理端从存储中提出固定大小的块状数据，我们观测存储器中数据存储变化如图 3.9 所示。

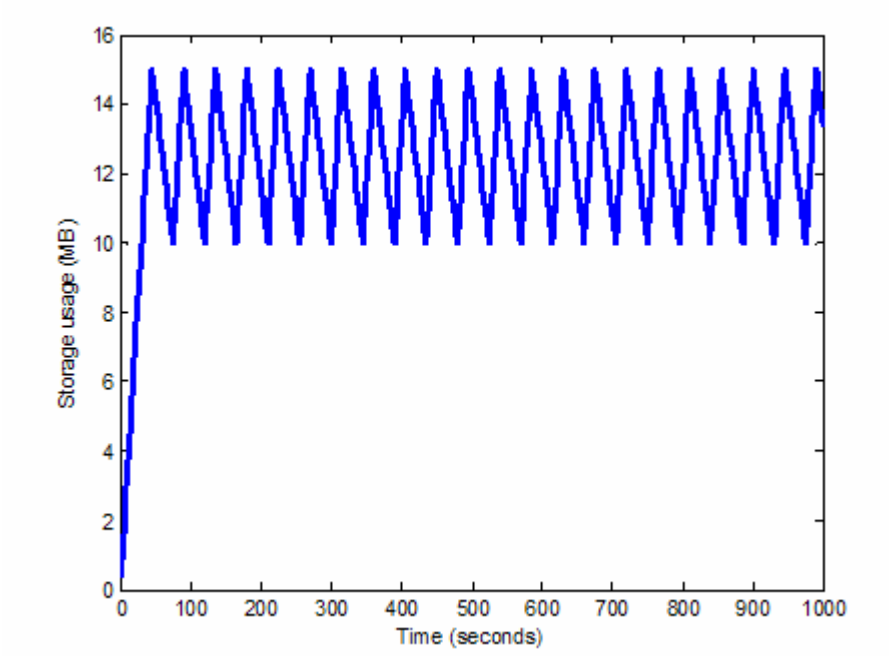

图3.9　处理端存储使用示意图

另外我们还可以从下图观察我们这个简单的实验系统数据传输、数据处理和数据存储的整体工作情况。

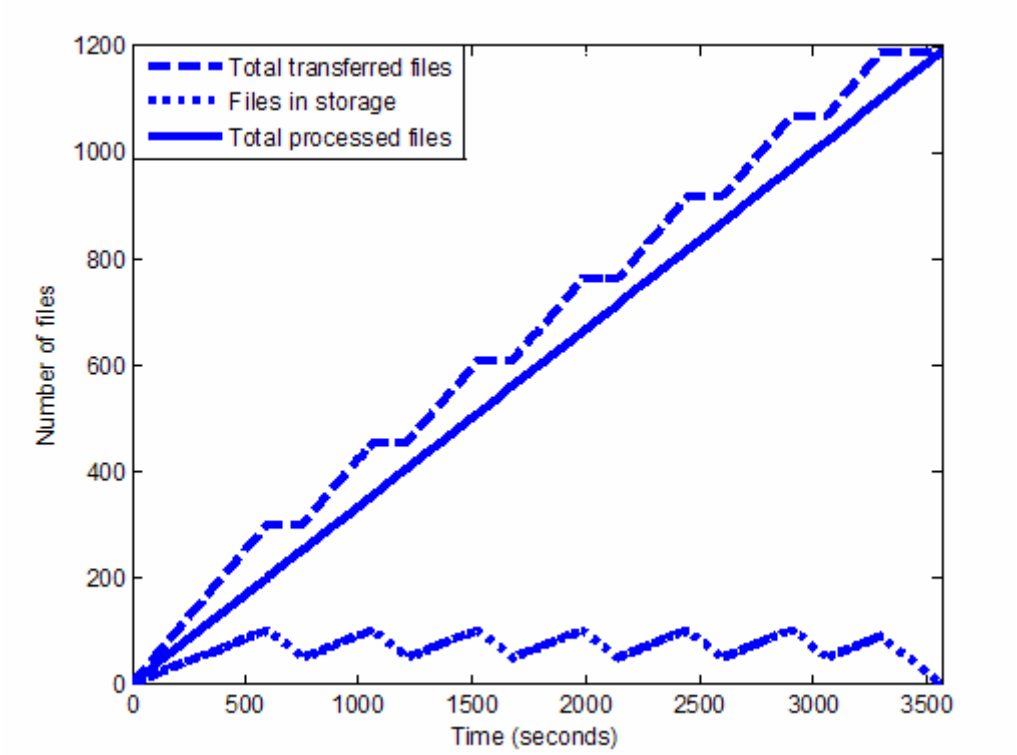

图3.10 实验系统整体工作情况图

第4章 结论及展望

我们根据网格计算中新兴发展的对数据流服务需求，结合与本实验室有合作关系的美国 LIGO 项目背景，设计了一套数据流应用系统，试图解决数据与处理能力相分离的问题，这其中涉及到了网络数据传输模型、存储资源和任务队列的调度管理等问题，而且需要建立一个同一的框架将各种问题统一考虑。我们通过计算机仿真测试及搭建实际网络测试的方法验证了我们设计的数据流应用系统的可行性，同时得到了许多令人欢欣鼓舞的性能评价。

我们已经描绘出了这套数据流应用系统的雏形，并取得了一些实质上的工作进展，但是这套系统还有许多不完善和值得改进的地方。

首先是应该将我们的数据流应用系统的应用范围推广到更加复杂的网络模型，比如接下来我们可以考虑多个数据产生源到多个数据处理端之间的数据传输模型，甚至还可以进一步考虑各个节点之间的通信，将调度管理推行到整个大系统。

再有我们应将系统应用到更多的实际应用中去，比如我们可以考虑如何改善我们的系统使之能符合当前流行的流式视频播放服务等具体的运用需求。

插图索引

表格索引

参考文献

[1] Surendra Reddy. “Grid Computing: Crossing the Chasm”. United States. 24 September 2004.http://sciencecareers.sciencemag.org/career_development/previous_issues/articles/3220/grid_computing_crossing_the_chasm.

[2] Wikipedia. Grid computing. http://en.wikipedia.org/wiki/Grid_computing.

[3] Geoff Brumfiel. "General relativity: To catch a wave". Nature 417, 482-484 (30 May 2002).

[4] 柯南[译]，探索宇宙涟漪，三思科学[J]. 第 6 期, 2002.

[5] 百度百科，引力波，http://baike.baidu.com/view/39440.htm.

[6] Einstein@Home 主题中文站，http://boinc.equn.com/einstein/.

[7] the Open Grid Forum, The Open Grid Services Architecture，Version 1.5(24 July 2006), . http://forge.gridforum.org/projects/ogsa-wg.

[8] Jeff Mausolf, 使用 Condor-G 和 Globus 进行作业提交、监视和控制，http://www.ibm.com/developerworks/cn/grid/gr-condorg1/index.html

[9] B. Agarwalla, N. Ahmed, D. Hilley, and U. Ramachandran, “Streamline: a Scheduling Heuristic for Streaming Applications on the Grid”, in *Proc. SPIE Multimedia Computing and Networking*, Vol. 6071, 2006.

[10] B. Agarwalla, N. Ahmed, D. Hilley, and U. Ramachandran, “Streamline: Scheduling Streaming Applications in a Wide Area Environment”, *Multimedia Systems*, Vol. 13, No. 1, pp. 69-85, 2007.

[11] B. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, and S. Tuecke, “Data Management and Transfer in High Performance Computational Grid Environments”, *Parallel Computing*, Vol. 28, No. 5, pp. 749-771, 2002.

[12] E. Deelman, C. Kesselman, G. Mehta, L. Meshkat, L. Pearlman, et. al., “GriPhyN and LIGO, Building a Virtual Data Grid for Gravitational Wave Scientists”, in *Proc. 11th IEEE Int. Symp. on High Performance Distributed Computing*, pp. 225-234, 2002.

[13] I. Foster and C. Kesselman, “Globus: A Metacomputing Infrastructure Toolkit”, *Int. J. Supercomputer Applications*, Vol. 11, No. 2, pp. 115-128, 1997.

[14] A. Ramakrishnan, G. Singh, H. Zhao, E. Deelman, R. Sakellariou, K. Vahi, K. Blackburn, D. Meyers, and M. Samidi, “Scheduling Data-intensive Workflow onto Storage-Constrained Distributed Resources”, in *Proc. 7th IEEE Int. Symp. on Cluster Computing and the Grid*, Rio de Janeiro, Brazil, pp. 401-409, 2007.

[15] Y. Zhu and D. Shasha, “Statstream: Statistical Monitoring of Thousands of Data Streams in Real Time”, *Technical Report TR2002-827, CS Dept, New York University*, 2002.

[16] S. Klasky, S. Ethier, Z. Lin, K. Martins, D. McCune and R. Samtaney, “Grid-Based Parallel Data Streaming Implemented for the Gyrokinetic Toroidal Code”, in *Proc. ACM/IEEE Supercomputing Conf.*, 2003.

致　　谢

曹军威老师是我 SRT 和毕业设计的指导老师，他为我指明了我研究工作的方向并为我提供了优良的实验室环境，同时他踏实严谨的工作作风和深厚精湛的学术造诣让我受益非浅，因此首先我向曹老师致以深深的谢意！

实验室的几位师兄给了我很多耐心的帮助，在此同样表示深深地感谢。其中特别值得指出的是张文师兄，作为我所在毕业设计项目的具体负责人，无论在技术支持上还是学术论文上都给了我很大的帮助。另外王震师兄负责实验室毕业设计实习生的进度管理，为我们付出了很多。

毕业实习阶段我在实验室度过了一段美好难忘的时光，感谢所有关系和支持我的人。

声　　明

本人郑重声明：所呈交的学位论文，是本人在导师指导下，独立进行研究工作所取得的成果。尽我所知，除文中已经注明引用的内容外，本学位论文的研究成果不包含任何他人享有著作权的内容。对本论文所涉及的研究工作做出贡献的其他个人和集体，均已在文中以明确方式标明。

签　名：_____　　日　期：_____

附录 A 外文资料的调研阅读报告

Brief Summary On Grid Computing

Grid computing was born out of the research and academic communities, originally it's a seamless solution for researchers in CERN who need high-performance computers with an extremely high processing speed, massive storage capacity, and an ability to transfer large amounts of data. It enables computes to work together , it's a emerging technology that transforms a computer infrastructure into an integrated, pervasive virtual environment for dynamic collaboration and shared resources anywhere in the world—providing users, especially in science, with unprecedented computing power, services and information and it's the next evolution of the Internet integrated with the next generation of distributed computing, peer-to-peer computing, virtual collaboration technologies and Web services, resulting in new, more powerful ways to conduct business and research.

But what is Grid computing?

Grid computing is based on three simple concepts:

(1) Virtualization, severing the hard-coded association of resources to systems.
(2) Resource allocation and management, dynamically allocating resources on demand, and managing them.
(3) Provisioning, configuring resources whenever and wherever needed.

Grid computing adapts to and dynamically aligns with research or business needs, it takes advantage of widespread spare computing capacity and the ubiquity of the Internet, which can tie these resources together but without dynamic alignment.

Dynamic scaling is another basic component of grid computing based on the idea that because the grid computer infrastructure can be built of small, standard, interchangeable components, computer users can start small and then simply add

more components as they scale. This means that grid computing can happen incrementally.

Aside in tradition network environment the utilization of many computers is quite low, as estimated, a sizeable portion of all enterprise computing has been estimated to run at only 20% to 40% of its total capacity. This underutilization is accepted as a tradeoff and forces people to choose between having enough scalability for peak loads(最大负荷) and buying too much so that they've invested a lot of money in idle capacity. This is the problem that Grid computing solves.

And another problem starved for solving is the IT infrastructure which supported by enterprises tends to be highly distributed but, in many cases, not easily managed, as to support increasing demands of their internal users and customers.

With the introduction of simpler technologies, greater bandwidth, automation, and more easily configurable infrastructures, Grid computing provides a more efficient means of provisioning IT services and manage them very cost effectively.

The impact of leveraging Grid computing will be dramatic. With the inherent ability of a Grid computing to provide nearly 100% uptime at an expected fraction of the costs of managing today's more static and fixed environment, both enterprises and institutional service providers can reap tremendous benefits.

Also, there is a particular appeal to being able to intelligently allocate finite resources to the appropriate business or research applications in a distributed enterprise. This technique offers companies and institutions flexibility, whether it is in the form of being able to redistribute resources to address new opportunities, or to enable applications to better serve current clients.

Many people confuse Grid computing with cluster-based computing, but cluster computing is not truly distributed computing. Cluster computing ties together similar types of resources in a data center with similar operating systems through special purpose connectors to deliver a specific application.

Grid computing, in contrast, offers heterogeneity by supporting different software without the need for special connectors. And Grids are dynamic in nature, while clusters typically contain a static number of processors.

Now more and more projects and applications in the field of Grid computing are researched and deployed. Such as UK e-Science Programme, Large Hadron Collider Computing Grid Project, myGrid DOE Science Grid, Globus Alliance and so on. And as this technology grew, industry leaders and research institutions have become heavily involved in developing and refining standards. While the Global Grid Forum acts as a clearinghouse of Grid standards, industry leaders also formed an Enterprise Grid Alliance, a consortium founded by Oracle, Optena, HP, Sun, and others to define standards and interfaces required to encourage Grid computing for enterprise applications. IBM has also played a significant role in the development of the standards and contributed to the development of the open-source standards-based software, called the Globus Toolkit. As a result, many technology vendors have started incorporating Grid technology across all lines of their business including services, hardware, and software.

At last, let’s give a expectation to Grid computing.

There are over billions PCs around the world, and thus almost every organization is sitting atop enormous, widely distributed, unused computing capacity. An innovative example of a project making use of this unused capacity is SETI@Home, the most popular and well-known distributed computing project on the Internet. This initiative harnesses idle PC computing cycles to work on the search for extraterrestrial intelligence. SETI@Home is now running on more than 2.5 million PCs in 226 countries.

Virtualization, the driving force behind Grid computing, can help reach this unused capacity. Web ser vices, the foundation of dynamic service delivery mechanism in Grid computing, enables enterprises to deliver software as a service more economically and dynamically that can scale in or out as demand shrinks or grows. Third-generation Grid computing platforms (such as

GridSpaces made by Optena) take advantage of these advances in distributed computing.

Although Internet and Grid computing are both new technologies, they have already proven themselves useful and their future looks promising. As technologies, networks and business models mature, I expect that grids will become commonplace for small and large enterprises and research institutions linking their various resources to support human communication, data access and computation. I also expect to see Grid computing emerging as a common service delivery platform as the Internet did

参考文献

[1] Surendra Reddy. “Grid Computing: Crossing the Chasm”. United States. 24 September 2004.http://sciencecareers.sciencemag.org/career_development/previous_issues/articles/3220/grid_computing_crossing_the_chasm

[2] Alan Kotok, “Collaboratories: Encouraging Remote Scientific Collaboration”, http://sciencecareers.sciencemag.org/career_development/previous_issues/articles/3290/collaboratories_encouraging_remote_scientific_collaboration.

[3] Richard T. Kouzes, James D. Myers, and William A. Wulf, "Collaboratories: Doing Science on the Internet." IEEE Computer29 (8), August 1996. http://collaboratory.emsl.pnl.gov/presentations/papers/IEEECollaboratories.html.

[4] Alan Koto,”Science Careers in Software: Feature Overview and Index”, 3 September 2004,http://sciencecareers.sciencemag.org/career_development/previous_issues/articles/3220/science_careers_in_software_feature_overview_and_index.

[5] Alan Kotok，”Collaboratories: Encouraging Remote Scientific Collaboration”, http://sciencecareers.sciencemag.org/career_development/previous_issues/articles/3290/collaboratories_encouraging_remote_scientific_collaboration

综合论文训练记录表

论文题目	网格数据流应用中任务管理和资源分配的算法设计和优化
主要内容以及进度安排	毕业设计的主要工作内容初步设定为三项：一是熟悉网格资源管理与调度模型；二是学习使用 Globus、Condor 等网格计算平台工具以及了解遗传算法等优化策略；三是设计网格计算中任务管理和资源分配的控制策略及算法，并进行优化，探索数据流在网格计算中的应用。 时间安排（按阶段） 第一阶段（1～2 周）此阶段主要进行文献查阅，了解网格计算的现状及发展趋势，完成外文调研综述。 第二阶段（3～5 周）此阶段主要熟悉和使用 Globus、Condor 等平台和工具包，学习遗传算法、模拟退火法等优化策略。 第三阶段（6～10 周）此阶段设计网格计算中任务管理和资源分配的控制策略及算法，搭建网格计算应用网络。 第四阶段（10～14 周）对控制策略及算法进行优化，通过仿真模拟实验及实际网络测试进行验证，得出满意的控制策略及算法。 第五阶段（15～16 周）最后总结，撰写论文，分析工作的收获与不足。 指导教师签字： 考核组组长签字： 08年 3 月 19 日
中期考核意见	该生研究了网格环境下数据流应用的规划问题，搭建了实际运行环境，并实现了 Matlab 仿真程序，并得到了相关结果。。完成了预期工作。同意通过中期考核。 考核组组长签字： 2008年 4 月 24日

指导教师评语	针对网格环境为数据流应用存储、处理和带宽控制的题，本论文提出了相应的算法和策略，并得到仿真和实际运行结果，达到综合训练的要求。 指导教师签字： 2008年6月18日
评阅教师评语	随着网络资源的不断扩展，数据资源的共享和传输变得十分突出，如何充分利用网络带宽，存储资源，是目前网格技术的重要内容，本文针对网格环境中海量数据的传输、计算和存储问题，采用数据流技术，给出相应的方法和策略，取得很好的仿真效果，论证充实，概念清晰。 评阅教师签字： 08年6月16日
答辩小组评语	该生在答辩过程中思路清晰，对所做工作能较完整呈现，回答问题基本正确，概念清楚，符合本科生论文训练要求，同意通过答辩。 答辩小组组长签字： 08年6月18日

总成绩：88

教学负责人签字：

2008年6月20日