面向多计算框架的容器云资源调度研究与实现

（申请清华大学工学硕士学位论文）

培 养 单 位：计 算 机 科 学 与 技 术 系
学　　科：计 算 机 科 学 与 技 术
研　究　生：龚　坤
指 导 教 师：武　永　卫　教　授

二〇一九年六月

Research and Implementation of Container Cloud Resource Scheduling for Multi-dimensional Computing Framework

Thesis Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the professional degree of

Doctor of Engineering

by

Gong Kun

(Computer Science and Technology)

Thesis Supervisor : Professor Wu Yongwei

June, 2019

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：

清华大学拥有在著作权法规定范围内学位论文的使用权，其中包括：（1）已获学位的研究生必须按学校规定提交学位论文，学校可以采用影印、缩印或其他复制手段保存研究生上交的学位论文；（2）为教学和科研目的，学校可以将公开的学位论文作为资料在图书馆、资料室等场所供校内师生阅读，或在校园网上供校内师生浏览部分内容。

本人保证遵守上述规定。

（保密的论文在解密后应遵守此规定）

作者签名：_____　　导师签名：_____

日　　期：_____　　日　　期：_____

摘 要

Docker 容器虚拟化技术能够和宿主机共享系统资源并且实现容器间的隔离，是当前技术研究的热点，已经成为容器的代名词。基于 Docker 的私有容器云平台获得广泛的应用，开始逐步取代传统虚拟机为基础构建的云计算系统。与传统的云计算集群类似，一个高效且强大的容器编排管理系统对数据中心资源利用率和集群性能具有至关重要的作用，Kubernetes 是当前容器云系统中应用最为广泛的容器调度系统，其轻量开源和强大的服务发现、集群监控和错误恢复能力深受用户好评。但 Kubernetes 过度专注于调度器能力和性能，其自身的资源分配和调度算法单一往往导致整个容器云集群资源利用率和均衡性很差，尤其是在多种计算框架同时部署的情况下其调度缺陷暴露无遗。

近年来，OpenShift 容器云平台用户针对 Kubernetes 调度器的不足往往需要开发自己的调度算法来适应特定的应用场景，这对追求性能和资源利用率的普通用户往往具有一定的难度，本文在深入分析 OpenShift 私有容器云平台 Kubernetes 调度器核心调度技术后，提出了一种全新的调度方案，本文主要内容如下:

- 基于开源的 OpenShift Origin 构建私有容器云平台 Paladin，在该平台上开发部署 Hadoop、Spark、MPI、Storm、Regraph 等十多种分布式计算框架，能够根据用户需求快速构建分布式计算平台。
- 深入研究 Kubernetes 调度核心技术，针对其调度算法的不足，提出了一种基于多维资源空闲率权重的评价函数和调度方法，该方法综合考虑物理节点 CPU、内存、磁盘、网络带宽空闲率和已部署的容器应用个数等因素影响，使用模糊层次分析法 FAHP(Fuzzy Analytic Hierarchy Process) 对集群资源自动建模并求解容器应用多维资源权重参数，选取最大评分节点进行容器调度。
- 针对新提出的调度方案，在 CloudSim-4.0 容器云仿真平台进行调度仿真，并与 Kubernetes 默认调度方案、Random 调度方案进行对比，能够极大提升容器云集群资源利用率和实现负载均衡性
- 在私有容器云平台 Paladin 上设计并实现该调度方案，使用多个计算框架同时进行调度性能测试，能够极大缩短多计算框架的应用执行时间。

关键词：Docker；容器云；调度策略；OpenShift 平台；计算框架；FAHP

Abstract

An abstract of a dissertation is a summary and extraction of research work and contributions. Included in an abstract should be description of research topic and research objective, brief introduction to methodology and research process, and summarization of conclusion and contributions of the research. An abstract should be characterized by independence and clarity and carry identical information with the dissertation. It should be such that the general idea and major contributions of the dissertation are conveyed without reading the dissertation.

An abstract should be concise and to the point. It is a misunderstanding to make an abstract an outline of the dissertation and words “the first chapter”, “the second chapter” and the like should be avoided in the abstract.

Key words are terms used in a dissertation for indexing, reflecting core information of the dissertation. An abstract may contain a maximum of 5 key words, with semi-colons used in between to separate one another.

Key Words: Docker; container cloud; scheduling strategy; openshift platform; computing framework; FAHP

目 录

第 1 章　引言

随着计算机互联网技术的飞速发展，网络规模不断扩大，各行业中应用业务量和数据呈现爆炸式的增长，如何快捷处理各种应用产生的海量数据已成为各大互联网公司面临的一个巨大挑战。继分布式计算、网格计算和并行计算后，一种全新的将整个互联网资源聚合起来处理数据的计算模式应运而生: 云计算。这种按需付费、共享资源、统一管理、可伸缩、可度量的计算模式发展迅猛，已经给信息产业带来了巨大的影响。

然而，云计算往往是以虚拟机作为云主机进行构建，将用户对资源的需求和软件服务虚拟化成虚拟机资源，然后进行虚拟机创建、操作系统安装以及应用部署。这虚拟化方式存在巨大的虚拟化开销，并且不同的虚拟机操作系统不同，其应用跨平台性较差。近年来，容器逐步取代虚拟机技术成为云计算的主流技术，给云计算带来新的革命，尤其是以 Docker 为代表的容器虚拟化技术获得了飞速的发展，基于 Docker 技术的容器云如雨后春笋般出现。云平台以其快速的应用部署、启动、交付以及优异的跨平台性能迅速占领市场。

与传统的云计算模式相似，一个强大的资源调度系统对集群性能和资源利用率起到决定性作用，Docker 虚拟化作为一种新型的容器云技术解决方案，其容器编排能力还存在很多不足。Google 开发的 Kubernetes 是容器云中容器调度系统的优秀代表，其轻量开源和强大的编排能力深受深受人们好评，但是其调度算法单一和资源利用不均衡性成为制约其性能的重要因素，本文在深入研究 Kubernetes 调度流程的基础上，设计并实现了一种新的调度方案，部署在基于开源 OpenShift Origin 开发的私有容器云 Paladin 上，极大提升了多种计算框架应用的性能。

1.1　云计算调度概述

云计算是当前较为普遍的一种分布式计算方式，通过将计算资源聚合成资源共享池对外提供按需付费、弹性计算的能力，对当前的计算机技术带来巨大的影响。其服务资源池有基础设施即服务 (IaaS，Infrastructure as a Service)、平台即服务 (PaaS，Platform as a Service) 以及软件即服务 (SaaS，Software as a Service) 三种服务模式。其计算类型根据用户对象的不同可以划分为公有云、私有云和混合云，典型的公有云有 Google Gmail、Amazon 的 EC2、微软 Azure、阿里云 ECS、百度云、腾讯云等。云计算需要底层的虚拟化技术作为支撑，当前维基百科收录的就有超过 60 种虚拟化技术，其中基于 X86 体系的虚拟化超过 50 种，当然也有 RISC 体

系的虚拟化，主要包括硬件虚拟化、操作系统层虚拟化、桌面虚拟化、应用程序虚拟化以及网络虚拟化等。从虚拟化的实现方式上可以划分为宿主架构和裸金属架构两种方式，其中宿主架构中虚拟机作为宿主操作系统的一个进程进行进行调度和管理，主要在个人的 PC 端应用较为广泛，如 VirtualBox、VMware Workstation、WindowVirtual PC 等。裸金属架构则不需要主机操作系统，直接以 Hypervisor 运行于物理硬件上，主要应用于服务器的虚拟化，本文主要关注服务器端大规模的云计算。应用最为广泛的如微软的 Hyper-V、VMWARE 的 ESX、开源的 XEN 和 KVM 等，云计算虚拟化架构中通常以虚拟机的方式提供给用户，用户根据自己的资源需求和软件服务申请合适的虚拟机进行服务，再进行大规模云计算环境构建时应选择合适的虚拟化架构方式，实现统一管理和跨平台的资源调度，综合利用各种虚拟化的性能优势，达到最终的目标。

在云计算中，资源的调度器对集群的性能和资源利用率起到决定性作用，是云计算的核心。传统虚拟机式的云计算集群对调度策略有相当多的研究，主要集中在降低系统能耗、提高数据中心资源利用率、集群服务器的负载均衡以及基于成本模式的资源管理研究。文献 []出一种根据虚拟机负载动态调节处理器电压和频率来降低集群能耗；文献 [] 通过动态分配云计算中心的虚拟机，减少服务器的数量来节约能耗；文献 [] 通过提出一种中心平衡器的平衡算法来实现集群服务器的负载均衡；文献 [] 将应用需求和物理机计算资源建模，基于蚁群算法、粒子群算法等迭代方式求解最佳分配策略，减少服务器的数量，从而提升集群资源的利用率；文献 [] 提出面向市场的体系结构和资源分配调度方法，该体系结构通过 SLA 资源分配器实现用户和服务商的协商，从而实现资源的优化配置。由此看出，在虚拟机式的云计算中，人们云计算资源调度方法进行深入的研究，广泛应用于当前的云计算系统中，对推动云计算的普及起到的巨大的推动作用。

1.2　容器云调度概述

虚拟机是当前云计算的主要实现形式，也是云计算的核心技术之一，除了虚拟机，容器在云计算中应用越加广泛，容器云发展迅猛。当前容器虚拟技术以 Docker 为典型代表，Docker 的底层实现是 LXC(Linux Container)，LXC 的资源管理完全依赖于内核的控制组 (cgroups)。和传统的虚拟技术不同、LXC 提供的虚拟环境是在操作系统层面实现的，主要面向进程，LXC 提供的虚拟环境也就是容器，操作系统可以为容器分配各种 CPU 时间、I/O 时间、内存、访问控制等，并提供单独的命名空间。其隔离性主要依赖于 Linux 内核的 namespace 特性，命名空间让进程之间彼此隔离，这种既能与宿主机共享资源又能同时提供用户隔离的虚拟化方案迅速

受到人们关注。以容器为虚拟化技术可以实现虚拟化较小的开销，应用可以实现快速的部署、交付以及较好的跨平台性。以 Docker 为基础构建的 CaaS(Container as a Service) 应运而生，各大互联网公司投入巨资进行研发，根据 451 Research 预测，容器作为一种高速成长型的工具，年增幅高达 40%。将作为应用最为广泛的云工具，超过 OpenStack、PaaS 以及其他相关的产品，根据其预测，应用容器将从 2016 年的 7.62 亿美元增长到 2020 的 27 亿美元，其预测是根据 125 家应用容器厂商为基础做出的。容器的管理和调度市场也在进行快速的组合并购，Apprenda 收购 Kubernetes 支持者 Kismatic，思科收购 Docker Swarm 支持者 ContainerX 等，这些活动都加速了容器云的飞速发展，当前较为出色的有 Google Container Engine、SAE、Cloud Foundry、AWS ECS、Red Hat OpenShift 等

容器云和传统的以虚拟机与基础构建的云计算一样需要一个性能强大的容器编排管理器负责容器的调度、创建、销毁、监控、重启、错误恢复以及服务的组合灯工作，决定容器云集群的性能和资源利用率，容器调度器同时也是推动容器迅速实现应用的重要因素。当前主要有三大主流的容器调度框架: Docker Swarm、Apache Mesos 以及 Google Kubernetes，其中应用最为广泛的要属开源轻量，性能强大的 Kubernetes。在调度架构和调度模式上三种调度框架也各不相同，Swarm 中调度算法主要包括最少容器、最多容器和随机调度三种；Memsos 则侧重于负载均衡，更多使用传统的虚拟机调度方法，如 DRF(Dominant Resource Fairness) 实现资源分配；Kubernetes 使用两阶段过滤评分选取最大评分的节点实现资源调度。几种调度方式各有优缺点，用户可以根据自己的需求不同选取响应的容器调度框架构建不同的容器云计算环境，实现资源更好的分配和调度，本文主要对 Kubernetes 的调度方式进行研究，在多种计算框架应用部署的情况下实现资源利用最大化，使应用的执行时间更短。

1.3　论文主要工作和结构安排

1.3.1　论文的主要工作

基于 Docker 容器虚拟化技术的 PasS 层 OpenShift 容器云平台使用 Kubernetes 进行容器管理和调度，该调度方式通过预选和优选两阶段选取评分最优的节点作为容器调度的目标，在优选阶段仅考虑内存和 CPU 的影响因素。本文针对其调度方式造成的资源利用率较低和负载不均衡的缺点，设计和实现了一个基于数学方法 FAHP 集群资源建模和参数自动求解的调度器，主要工作如下:

(1) 基于开源的 OpenShift Origin 构建实验室私有 PaaS 层容器云平台 Paladin，在该平台上开发部署十多种分布式计算框架，普通用户可以根据实际需求快速

配置和构建自己所需的计算环境。

(2) 深入分析 OpenShift 容器编排管理器 Kubernetes 的调度流程和不足之处，提出了一种综合考虑应用特性的多维资源空闲率权重的评价函数和调度方法，该方法充分考虑容器应用 CPU、内存、磁盘、网络带宽和已部署 Pod 数量的影响，最后通过对集群物理节点资源和应用资源的数学建模，利用 FAHP 方法自动构建满足一致性要求的模糊成对比矩阵和判断矩阵，实现应用参数权重的自动求解，选取评分最高的节点作为容器应用的调度目标。

(3) 在容器仿真平台 CloudSim-4.0 上对新的调度方案进行仿真，对比分析 Kubernetes 调度方法以及 Random 调度方法的性能，实验表明能够极大提高集群资源利用率和实现负载均衡。

(4) 在私有容器云平台 Paladin 上设计开发该调度方案，部署多种计算框架应用进行应用性能测试，实验表明新的调度方案能极大缩短多种计算框架的执行时间，提升私有容器云集群性能。

1.3.2　本文结构安排

本论文总共分为六个章节，第一章主要阐述传统虚拟机技术构建的云计算和新兴 Docker 容器技术构建的容器云，云计算底层虚拟化技术的基本架构、典型的云计算服务和云计算中资源调度器方法。接着介绍容器云底层的容器虚拟化技术支撑，代表性的容器云服务以及三大主流的容器编排器，进而引出 Kubernetes 容器编排器的不足和本文需要研究解决的问题。第二章首先对比 Docker 容器虚拟化技术和传统虚拟机技术以及 Docker 构建的容器云平台，接着比较分析容器云中三种主要的调度系统，最后针对 Kubernetes 容器编排管理器的组织架构、调度流程和原理以及其调度的不足进行深入分析。第三章针对 Kubernetes 调度器不足，设计一种新的调度方法和调度流程，新的调度方案充分考虑容器应用特点和集群物理资源的特点，主要包括其调度流程、多维资源建模、反馈器的设计以及全新的评价函数构建。第四章使用 FAHP 方法解决法解决新的调度方案中多维资源权重的问题，对应用和集群资源进行数学建模、自动构满足一致性要求的建模糊成对比矩阵和判断矩阵，自动求解应用资源权重参数。第五章首先使用 CloudSim-4.0 进行仿真环境的搭建，对比分析新的调度方法和 Kubernetes 默认调度方案以及 Random 调度方法在集群资源利用率、负载均衡性方面的性能。然后在实验室私有容器云平台 Paladin 上开发部署十几种分布式计算框架，将新的调度方案应用于 Paladin 中，使用多个计算框架应用对其性能进行测试。最后一章对本文工作进行总结，展望未来的研究方向。

第 2 章　中华人民共和国

2.1　其它例子

在第 1 章中我们学习了贝叶斯公式 (**??**)，这里我们复习一下：

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{2-1}$$

2.1.1　绘图

本模板不再预先装载任何绘图包（如 pstricks，pgf 等），完全由用户来决定。个人觉得 pgf 不错，不依赖于 Postscript。此外还有很多针对 LaTeX 的 GUI 作图工具，如 XFig(jFig), WinFig, Tpx, Ipe, Dia, Inkscape, LaTeXPiX, jPicEdt, jaxdraw 等等。

2.1.2　插图

强烈推荐《LaTeX 2_ε 插图指南》！关于子图形的使用细节请参看 subcaption 宏包的说明文档。

2.1.2.1　一个图形

一般图形都是处在浮动环境中。之所以称为浮动是指最终排版效果图形的位置不一定与源文件中的位置对应[①]，这也是刚使用 LaTeX 同学可能遇到的问题。如果要强制固定浮动图形的位置，请使用 float 宏包，它提供了 `[H]` 参数，比如图 3.1。

图 2.1　利用 Xfig 制图

大学之道，在明明德，在亲民，在止于至善。知止而后有定；定而后能静；静而后能安；安而后能虑；虑而后能得。物有本末，事有终始。知所先后，则近道矣。古之欲明明德于天下者，先治其国；欲治其国者，先齐其家；欲齐其家者，先修其身；欲修其身者，先正其心；欲正其心者，先诚其意；欲诚其意者，先致其知；致

① This is not a bug, but a feature of LaTeX!

知在格物。物格而后知至；知至而后意诚；意诚而后心正；心正而后身修；身修而后家齐；家齐而后国治；国治而后天下平。自天子以至于庶人，壹是皆以修身为本。其本乱而未治者否矣。其所厚者薄，而其所薄者厚，未之有也！

——《大学》

2.1.2.2　多个图形

如果多个图形相互独立，并不共用一个图形计数器，那么用 `minipage` 或者 `parbox` 就可以。否则，请参看图 3.2，它包含两个小图，分别是图 3.2(a)和图 3.2(b)。推荐使用 `\subcaptionbox`，因为可以像图 3.2 那样对齐子图的标题，也可以使用 subcaption 宏包的 `\subcaption`（放在 minipage 中，用法同`\caption`）或是 subfigure 、subtable 环境，像图 3.3，不要再用 `\subfloat`、`\subfigure` 和 `\subtable`。

(a) 第一个小图形

(b) 第二个小图形，注意这个图略矮些。如果标题很长的话，它会自动换行

图 2.2　包含子图形的大图形 (subcaptionbox 示例)

(a) 第一个小图形

(b) 第二个小图形，注意这个图略矮些。subfigure 中同一行的子图在顶端对齐。

图 2.3　包含子图形的大图形 (subfigure 示例)

古之学者必有师。师者，所以传道受业解惑也。人非生而知之者，孰能无惑？惑而不从师，其为惑也，终不解矣。生乎吾前，其闻道也固先乎吾，吾从而师之；

图 2.4　并排第一个图

图 2.5　并排第二个图

生乎吾後，其闻道也亦先乎吾，吾从而师之。吾师道也，夫庸知其年之先後生於吾乎！是故无贵无贱无长无少，道之所存，师之所存也。

嗟乎！师道之不传也久矣，欲人之无惑也难矣。古之圣人，其出人也远矣，犹且从师而问焉；今之众人，其下圣人也亦远矣，而耻学於师。是故圣益圣，愚益愚。圣人之所以为圣，愚人之所以为愚，其皆出於此乎？爱其子，择师而教之，於其身也，则耻师焉，惑焉。彼童子之师，授之书而习其句读者，非吾所谓传其道、解其惑者也。句读之不知，惑之不解，或师焉，或不焉，小学而大遗，吾未见其明也。巫医、乐师、百工之人不耻相师，士大夫之族曰“师”曰“弟子”之云者，则群聚而笑之。问之，则曰：彼与彼年相若也，道相似也，位卑则足羞，官盛则近谀。呜呼！师道之不复，可知矣。巫医、乐师、百工之人。吾子不齿，今其智乃反不能及，其可怪也欤！圣人无常师。孔子师郯子、苌子、师襄、老聃。郯子之徒，其贤不及孔子。孔子曰：“三人行，必有我师。”是故弟子不必不如师，师不必贤於弟子。闻道有先後，术业有专攻，如是而已。

如果要把编号的两个图形并排，那么小页就非常有用了：

李氏子蟠，年十七，好古文、六艺，经传皆通习之，不拘於时，学於余。余嘉其能行古道，作师说以贻之。

——韩愈（唐）

第 3 章　中华人民共和国

3.1　其它例子

在第 1 章中我们学习了贝叶斯公式 (**??**)，这里我们复习一下：

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{3-1}$$

3.1.1　绘图

本模板不再预先装载任何绘图包（如 pstricks，pgf 等），完全由用户来决定。个人觉得 pgf 不错，不依赖于 Postscript。此外还有很多针对 LaTeX 的 GUI 作图工具，如 XFig(jFig), WinFig, Tpx, Ipe, Dia, Inkscape, LaTeXPiX, jPicEdt, jaxdraw 等等。

3.1.2　插图

强烈推荐《LaTeX 2_ε 插图指南》！关于子图形的使用细节请参看 subcaption 宏包的说明文档。

3.1.2.1　一个图形

一般图形都是处在浮动环境中。之所以称为浮动是指最终排版效果图形的位置不一定与源文件中的位置对应[①]，这也是刚使用 LaTeX 同学可能遇到的问题。如果要强制固定浮动图形的位置，请使用 float 宏包，它提供了 `[H]` 参数，比如图 3.1。

图 3.1　利用 Xfig 制图

大学之道，在明明德，在亲民，在止于至善。知止而后有定；定而后能静；静而后能安；安而后能虑；虑而后能得。物有本末，事有终始。知所先后，则近道矣。古之欲明明德于天下者，先治其国；欲治其国者，先齐其家；欲齐其家者，先修其身；欲修其身者，先正其心；欲正其心者，先诚其意；欲诚其意者，先致其知；致

① This is not a bug, but a feature of LaTeX!

知在格物。物格而后知至；知至而后意诚；意诚而后心正；心正而后身修；身修而后家齐；家齐而后国治；国治而后天下平。自天子以至于庶人，壹是皆以修身为本。其本乱而未治者否矣。其所厚者薄，而其所薄者厚，未之有也！

——《大学》

3.1.2.2　多个图形

如果多个图形相互独立，并不共用一个图形计数器，那么用 `minipage` 或者 `parbox` 就可以。否则，请参看图 3.2，它包含两个小图，分别是图 3.2(a)和图 3.2(b)。推荐使用 `\subcaptionbox`，因为可以像图 3.2 那样对齐子图的标题，也可以使用 subcaption 宏包的 `\subcaption`（放在 minipage 中，用法同`\caption`）或是 subfigure 、subtable 环境，像图 3.3，不要再用 `\subfloat`、`\subfigure` 和 `\subtable`。

(a) 第一个小图形

(b) 第二个小图形，注意这个图略矮些。如果标题很长的话，它会自动换行

图 3.2　包含子图形的大图形 (subcaptionbox 示例)

(a) 第一个小图形

(b) 第二个小图形，注意这个图略矮些。subfigure 中同一行的子图在顶端对齐。

图 3.3　包含子图形的大图形 (subfigure 示例)

古之学者必有师。师者，所以传道受业解惑也。人非生而知之者，孰能无惑？惑而不从师，其为惑也，终不解矣。生乎吾前，其闻道也固先乎吾，吾从而师之；

图 3.4　并排第一个图

图 3.5　并排第二个图

生乎吾後，其闻道也亦先乎吾，吾从而师之。吾师道也，夫庸知其年之先後生於吾乎！是故无贵无贱无长无少，道之所存，师之所存也。

嗟乎！师道之不传也久矣，欲人之无惑也难矣。古之圣人，其出人也远矣，犹且从师而问焉；今之众人，其下圣人也亦远矣，而耻学於师。是故圣益圣，愚益愚。圣人之所以为圣，愚人之所以为愚，其皆出於此乎？爱其子，择师而教之，於其身也，则耻师焉，惑焉。彼童子之师，授之书而习其句读者，非吾所谓传其道、解其惑者也。句读之不知，惑之不解，或师焉，或不焉，小学而大遗，吾未见其明也。巫医、乐师、百工之人不耻相师，士大夫之族曰“师”曰“弟子”之云者，则群聚而笑之。问之，则曰：彼与彼年相若也，道相似也，位卑则足羞，官盛则近谀。呜呼！师道之不复，可知矣。巫医、乐师、百工之人。吾子不齿，今其智乃反不能及，其可怪也欤！圣人无常师。孔子师郯子、苌子、师襄、老聃。郯子之徒，其贤不及孔子。孔子曰：“三人行，必有我师。”是故弟子不必不如师，师不必贤於弟子。闻道有先後，术业有专攻，如是而已。

如果要把编号的两个图形并排，那么小页就非常有用了：

李氏子蟠，年十七，好古文、六艺，经传皆通习之，不拘於时，学於余。余嘉其能行古道，作师说以贻之。

——韩愈（唐）

插图索引

表格索引

公式索引

参考文献

[1] 薛瑞尼. ThuThesis: 清华大学学位论文模板[EB/OL]. 2017. https://github.com/xueruini/thuthesis.

致 谢

衷心感谢导师 xxx 教授和物理系 xxx 副教授对本人的精心指导。他们的言传身教将使我终生受益。

在美国麻省理工学院化学系进行九个月的合作研究期间，承蒙 xxx 教授热心指导与帮助，不胜感激。感谢 xx 实验室主任 xx 教授，以及实验室全体老师和同学们的热情帮助和支持！本课题承蒙国家自然科学基金资助，特此致谢。

感谢 LaTeX 和 ThuThesis[1]，帮我节省了不少时间。

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　外文资料原文

The title of the English paper

Abstract: As one of the most widely used techniques in operations research, *mathematical programming* is defined as a means of maximizing a quantity known as *bjective function*, subject to a set of constraints represented by equations and inequalities. Some known subtopics of mathematical programming are linear programming, nonlinear programming, multiobjective programming, goal programming, dynamic programming, and multilevel programming[1].

It is impossible to cover in a single chapter every concept of mathematical programming. This chapter introduces only the basic concepts and techniques of mathematical programming such that readers gain an understanding of them throughout the book[2,3].

A.1　Single-Objective Programming

The general form of single-objective programming (SOP) is written as follows,

$$\begin{cases} \max \ \ f(x) \\ \text{subject to:} \\ \qquad g_j(x) \le 0, \quad j = 1, 2, \cdots, p \end{cases} \tag{123}$$

which maximizes a real-valued function f of $x = (x_1, x_2, \cdots, x_n)$ subject to a set of constraints.

Definition A.1：　In SOP, we call x a decision vector, and $x_1, x_2, \cdots, x_n$ decision variables. The function f is called the objective function. The set

$$S = \left\{ x \in \Re^n \;\middle|\; g_j(x) \le 0,\ j = 1, 2, \cdots, p \right\} \tag{456}$$

is called the feasible set. An element x in S is called a feasible solution.

Definition A.2：　A feasible solution x^* is called the optimal solution of SOP if and only if

$$f(x^*) \ge f(x) \tag{A-1}$$

for any feasible solution x.

One of the outstanding contributions to mathematical programming was known as the Kuhn-Tucker conditionsA-2. In order to introduce them, let us give some definitions. An inequality constraint $g_j(x) \leq 0$ is said to be active at a point x^* if $g_j(x^*) = 0$. A point x^* satisfying $g_j(x^*) \leq 0$ is said to be regular if the gradient vectors $\nabla g_j(x)$ of all active constraints are linearly independent.

Let x^* be a regular point of the constraints of SOP and assume that all the functions $f(x)$ and $g_j(x), j = 1, 2, \cdots, p$ are differentiable. If x^* is a local optimal solution, then there exist Lagrange multipliers $\lambda_j, j = 1, 2, \cdots, p$ such that the following Kuhn-Tucker conditions hold,

$$\begin{cases} \nabla f(x^*) - \sum\limits_{j=1}^{p} \lambda_j \nabla g_j(x^*) = 0 \\ \lambda_j g_j(x^*) = 0, \quad j = 1, 2, \cdots, p \\ \lambda_j \geq 0, \quad j = 1, 2, \cdots, p. \end{cases} \tag{A-2}$$

If all the functions $f(x)$ and $g_j(x), j = 1, 2, \cdots, p$ are convex and differentiable, and the point x^* satisfies the Kuhn-Tucker conditions (A-2), then it has been proved that the point x^* is a global optimal solution of SOP.

A.1.1 Linear Programming

If the functions $f(x), g_j(x), j = 1, 2, \cdots, p$ are all linear, then SOP is called a *linear programming*.

The feasible set of linear is always convex. A point x is called an extreme point of convex set S if $x \in S$ and x cannot be expressed as a convex combination of two points in S. It has been shown that the optimal solution to linear programming corresponds to an extreme point of its feasible set provided that the feasible set S is bounded. This fact is the basis of the *simplex algorithm* which was developed by Dantzig as a very efficient method for solving linear programming.

Roughly speaking, the simplex algorithm examines only the extreme points of the feasible set, rather than all feasible points. At first, the simplex algorithm selects an extreme point as the initial point. The successive extreme point is selected so as to improve the objective function value. The procedure is repeated until no improvement in objective function value can be made. The last extreme point is the optimal solution.

Table 1　This is an example for manually numbered table, which would not appear in the list of tables

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

A.1.2　Nonlinear Programming

If at least one of the functions $f(x), g_j(x), j = 1, 2, \cdots, p$ is nonlinear, then SOP is called a *nonlinear programming*.

A large number of classical optimization methods have been developed to treat special-structural nonlinear programming based on the mathematical theory concerned with analyzing the structure of problems.

Figure 1　This is an example for manually numbered figure, which would not appear in the list of figures

Now we consider a nonlinear programming which is confronted solely with maximizing a real-valued function with domain $\mathfrak{R}^n$. Whether derivatives are available or not, the usual strategy is first to select a point in $\mathfrak{R}^n$ which is thought to be the most likely place where the maximum exists. If there is no information available on which to base such a selection, a point is chosen at random. From this first point an attempt is made to construct a sequence of points, each of which yields an improved objective function value over its predecessor. The next point to be added to the sequence is chosen by analyzing the behavior of the function at the previous points. This construction continues until some termination criterion is met. Methods based upon this strategy are called *ascent methods*, which can be classified as *direct methods*, *gradient methods*, and *Hessian methods* according to the information about the behavior of objective function f. Direct methods require only that the function can be evaluated at each point. Gradient methods require the evaluation of first derivatives of f. Hessian methods require the evaluation of second

derivatives. In fact, there is no superior method for all problems. The efficiency of a method is very much dependent upon the objective function.

A.1.3 Integer Programming

Integer programming is a special mathematical programming in which all of the variables are assumed to be only integer values. When there are not only integer variables but also conventional continuous variables, we call it *mixed integer programming*. If all the variables are assumed either 0 or 1, then the problem is termed a *zero-one programming*. Although integer programming can be solved by an *exhaustive enumeration* theoretically, it is impractical to solve realistically sized integer programming problems. The most successful algorithm so far found to solve integer programming is called the *branch-and-bound enumeration* developed by Balas (1965) and Dakin (1965). The other technique to integer programming is the *cutting plane method* developed by Gomory (1959).

Uncertain Programming　(*BaoDing Liu,* 2006.2)

References

NOTE: These references are only for demonstration. They are not real citations in the original text.

[1] Donald E. Knuth. The TEXbook. Addison-Wesley, 1984. ISBN: 0-201-13448-9

[2] Paul W. Abrahams, Karl Berry and Kathryn A. Hargreaves. TEX for the Impatient. Addison-Wesley, 1990. ISBN: 0-201-51375-7

[3] David Salomon. The advanced TEXbook. New York : Springer, 1995. ISBN:0-387-94556-3

附录 B　外文资料的调研阅读报告或书面翻译

英文资料的中文标题

摘要：本章为外文资料翻译内容。如果有摘要可以直接写上来，这部分好像没有明确的规定。

B.1　单目标规划

北冥有鱼，其名为鲲。鲲之大，不知其几千里也。化而为鸟，其名为鹏。鹏之背，不知其几千里也。怒而飞，其翼若垂天之云。是鸟也，海运则将徙于南冥。南冥者，天池也。

$$p(y|\mathbf{x}) = \frac{p(\mathbf{x}, y)}{p(\mathbf{x})} = \frac{p(\mathbf{x}|y)p(y)}{p(\mathbf{x})} \tag{123}$$

吾生也有涯，而知也无涯。以有涯随无涯，殆已！已而为知者，殆而已矣！为善无近名，为恶无近刑，缘督以为经，可以保身，可以全生，可以养亲，可以尽年。

B.1.1　线性规划

庖丁为文惠君解牛，手之所触，肩之所倚，足之所履，膝之所倚，砉然响然，奏刀騞然，莫不中音，合于桑林之舞，乃中经首之会。

表 1　这是手动编号但不出现在索引中的一个表格例子

Network Topology		# of nodes	# of clients			Server
GT-ITM	Waxman Transit-Stub	600	2%	10%	50%	Max. Connectivity
Inet-2.1		6000				
Xue	Rui	Ni	ThuThesis			
	ABCDEF					

文惠君曰：“嘻，善哉！技盖至此乎？”庖丁释刀对曰：“臣之所好者道也，进乎技矣。始臣之解牛之时，所见无非全牛者；三年之后，未尝见全牛也；方今之时，臣以神遇而不以目视，官知止而神欲行。依乎天理，批大郤，导大窾，因其固然。技经肯綮之未尝，而况大軱乎！良庖岁更刀，割也；族庖月更刀，折也；今臣之刀十九年矣，所解数千牛矣，而刀刃若新发于硎。彼节者有间而刀刃者无厚，以无厚入有间，恢恢乎其于游刃必有余地矣。是以十九年而刀刃若新发于硎。虽然，每至

于族，吾见其难为，怵然为戒，视为止，行为迟，动刀甚微，謋然已解，如土委地。提刀而立，为之而四顾，为之踌躇满志，善刀而藏之。”

文惠君曰：“善哉！吾闻庖丁之言，得养生焉。”

B.1.2　非线性规划

孔子与柳下季为友，柳下季之弟名曰盗跖。盗跖从卒九千人，横行天下，侵暴诸侯。穴室枢户，驱人牛马，取人妇女。贪得忘亲，不顾父母兄弟，不祭先祖。所过之邑，大国守城，小国入保，万民苦之。孔子谓柳下季曰：“夫为人父者，必能诏其子；为人兄者，必能教其弟。若父不能诏其子，兄不能教其弟，则无贵父子兄弟之亲矣。今先生，世之才士也，弟为盗跖，为天下害，而弗能教也，丘窃为先生羞之。丘请为先生往说之。”

图 1　这是手动编号但不出现索引中的图片的例子

柳下季曰：“先生言为人父者必能诏其子，为人兄者必能教其弟，若子不听父之诏，弟不受兄之教，虽今先生之辩，将奈之何哉？且跖之为人也，心如涌泉，意如飘风，强足以距敌，辩足以饰非。顺其心则喜，逆其心则怒，易辱人以言。先生必无往。”

孔子不听，颜回为驭，子贡为右，往见盗跖。

B.1.3　整数规划

盗跖乃方休卒徒大山之阳，脍人肝而餔之。孔子下车而前，见谒者曰：“鲁人孔丘，闻将军高义，敬再拜谒者。”谒者入通。盗跖闻之大怒，目如明星，发上指冠，曰：“此夫鲁国之巧伪人孔丘非邪？为我告之：尔作言造语，妄称文、武，冠枝木之冠，带死牛之胁，多辞缪说，不耕而食，不织而衣，摇唇鼓舌，擅生是非，以迷天下之主，使天下学士不反其本，妄作孝弟，而侥幸于封侯富贵者也。子之罪大极重，疾走归！不然，我将以子肝益昼餔之膳。”

附录 C　其它附录

前面两个附录主要是给本科生做例子。其它附录的内容可以放到这里，当然如果你愿意，可以把这部分也放到独立的文件中，然后将其 `\input` 到主文件中。

个人简历、在学期间发表的学术论文与研究成果

个人简历

xxxx 年 xx 月 xx 日出生于 xx 省 xx 县。

xxxx 年 9 月考入 xx 大学 xx 系 xx 专业，xxxx 年 7 月本科毕业并获得 xx 学士学位。

xxxx 年 9 月免试进入 xx 大学 xx 系攻读 xx 学位至今。

发表的学术论文

[1] Yang Y, Ren T L, Zhang L T, et al. Miniature microphone with silicon- based ferroelectric thin films. Integrated Ferroelectrics, 2003, 52:229-235. (SCI 收录, 检索号:758FZ.)

[2] 杨轶, 张宁欣, 任天令, 等. 硅基铁电微声学器件中薄膜残余应力的研究. 中国机械工程, 2005, 16(14):1289-1291. (EI 收录, 检索号:0534931 2907.)

[3] 杨轶, 张宁欣, 任天令, 等. 集成铁电器件中的关键工艺研究. 仪器仪表学报, 2003, 24(S4):192-193. (EI 源刊.)

[4] Yang Y, Ren T L, Zhu Y P, et al. PMUTs for handwriting recognition. In press. (已被 Integrated Ferroelectrics 录用. SCI 源刊.)

[5] Wu X M, Yang Y, Cai J, et al. Measurements of ferroelectric MEMS microphones. Integrated Ferroelectrics, 2005, 69:417-429. (SCI 收录, 检索号:896KM)

[6] 贾泽, 杨轶, 陈兢, 等. 用于压电和电容微麦克风的体硅腐蚀相关研究. 压电与声光, 2006, 28(1):117-119. (EI 收录, 检索号:06129773469)

[7] 伍晓明, 杨轶, 张宁欣, 等. 基于 MEMS 技术的集成铁电硅微麦克风. 中国集成电路, 2003, 53:59-61.

研究成果

[1] 任天令, 杨轶, 朱一平, 等. 硅基铁电微声学传感器畴极化区域控制和电极连接的方法: 中国, CN1602118A. (中国专利公开号)

[2] Ren T L, Yang Y, Zhu Y P, et al. Piezoelectric micro acoustic sensor based on ferroelectric materials: USA, No.11/215, 102. (美国发明专利申请号)

综合论文训练记录表

学生姓名		学号		班级	
论文题目					
主要内容以及进度安排	指导教师签字： 考核组组长签字： 年　　月　　日				
中期考核意见	考核组组长签字： 年　　月　　日				

指导教师评语	指导教师签字：_____ 年　　月　　日
评阅教师评语	评阅教师签字：_____ 年　　月　　日
答辩小组评语	答辩小组组长签字：_____ 年　　月　　日

总成绩：_____

教学负责人签字：_____

年　　月　　日