弹性云计算资源调度进化式仿真优化研究

（申请清华大学工学博士学位论文）

培养单位：　自动化系

学　　科：　控制科学与工程

研 究 生：　张　帆

指导教师：　吴　澄　教　授

二〇一一年十月

弹性云计算资源调度进化式仿真优化研究

张 帆

Evolutionary Simulation based Optimization for Resource Scheduling on Elastic Cloud Platforms

Dissertation Submitted to

Tsinghua University

in partial fulfillment of the requirement

for the degree of

Doctor of Engineering

by

Fan Zhang

(Control Science and Engineering)

Dissertation Supervisor: Professor Cheng Wu

October, 2011

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；（3）根据《中华人民共和国学位条例暂行实施办法》，向国家图书馆报送可以公开的学位论文。

本人保证遵守上述规定。

（保密的论文在解密后遵守此规定）

作者签名：_____ 导师签名：_____

日　　期：_____ 日　　期：_____

摘　要

构建高效的数据中心以提供可伸缩的弹性服务一直是云计算技术领域面临的重要科学问题。其实际意义在于，用最合适的计算资源满足云服务提供商和终端用户在 Service Level Agreement(SLA)上的协议，提高资源利用率和用户满意度。针对本科学问题，本文利用自底向上的方法将其分解为五个循序渐进的科学任务并逐一提出解决方案。本文的核心贡献包括：

（1）致力于解决虚拟资源自适应供给问题。将弹性云计算平台建模成多层次，内部并联外部串联的队列，提出针对任意结构模式下表达用户请求到达率和平均响应时间的逻辑关系，并因此灵活自适应和弹性的提供虚拟资源。实验结果证实仿真模型和真实实验结果比较一致，并证实本方法相对传统基于利用率调度方法能更加高效低成本的满足 SLA 协议；

（2）针对队列模型过于简化运行期的随机性因素问题，文中引入仿真优化粗糙模型的思想，提出了一种低开销仿真优化调度方法从而实现虚拟资源的敏捷供给。实验结果证实了本低开销调度方法能够减少仿真优化阶段的调度时间，并能获得具有良好性能的满意调度策略；

（3）针对弹性云计算平台的多阶段、实时性和动态性情况，本文充分利用低开销仿真优化的调度优势和多任务的松耦合特性，构建基于连续仿真优化方法的多阶段迭代调度模型，理论和实验均证实了方法的低开销特性。基于此优势，在更短时间能做出的决策方案能够更加细粒度的捕获工作负载强度的局部特性，整体上提升系统的调度性能；

（4）针对弹性云计算平台多任务负载的局部特性，分析了相邻阶段多任务负载的相似性特点从而自适应的动态划分决策阶段，将迭代仿真优化方法拓展成一种进化仿真优化方法，从而自适应的在多任务负载变化剧烈和平缓处分割和合并调度区间，实现整体仿真调度性能的提升；

（5）拓展低开销仿真优化方法根据多任务负载相似性分割和合并其调度区间的思想，在数据中心多评价指标体系中根据指标相似性自适应剔除和合并相似性较弱和较强的关联指标，从而构建出最佳评价策略集合。

关键词：弹性云计算；仿真优化；序优化；虚拟资源管理；自适应供给

Abstract

It is fundamentally important to construct highly efficient and reliable data center in order to provide elastic service. Utilizing proper computing resources to satisfy the Service Level Agreement (SLA) between cloud service providers and terminal users is its ultimate significance for further improving resource utilization and user satisfaction. Focused on this scientific problem, this dissertation pins it down to five detailed scientific assignments using down-up design style and provides solutions accordingly. Main contributions are summarized as follows.

(1) Adaptive provisioning of virtual resources is firstly discussed. It models the Elastic Computing Platform (ECP) as a multi-tier, internally parallel and externally sequentially connected queue, and provides the logical relationship between the arrival rate of user requests and average response time, and provisions virtual resource adaptively, flexibly and elastically. Experiments verify the queuing model matches with real benchmarking results. Furthermore, it is substantiated that the method outperforms traditional utilization based method, both in terms of the satisfaction of SLA and cost minimization;

(2) Due to the over simplification of the runtime conditions of the queuing model, it brings in the concept of simulation based optimization, proposes a Low Overhead Scheduling (LOS) based method to provision virtual resources agily and elasticly. Simulation results show this method narrows down the simulation time in scheduling stage, and delivers good-enough schedules to apply;

(3) Multi-stage, real-time and dynamic provisioning as it is in ECP, it utilizes the loosely coupled multitasking essence, and constructs a multi-phase iterative simulation based scheduling model, and proves the LOS both in theoretical and experimental verifications. Based on this advantage, it captures the fine-grained characteristics of workload and the scheduling performance is overall improved.

(4) Characterizing the local patterns in multitaskingworkload and analyzing

the similarities between inter-phase and intra-phase, it generates decision stage automatically, thereby extending the iterative LOS into an evolutionary LOS method. It adaptively partitions or merges scheduling phase when workload similarity becomes large or small to improve performance to a higher level.

(5) It applies this concept in evaluating and reducing multiple performance metrics of ECP. It automatically gets rid of or merges performance metrics based on their similarities to construct an optimal performance metric set.

Key words: Elastic Cloud Computing; Simulation based Optimization; Ordinal Optimization; Virtual Resource Management; Adaptive Provisioning

目 录

第 1 章 引言 1
1.1 研究背景：分布式计算体系的变迁 1
1.1.1 复杂应用催生的分布式计算 1
1.1.2 满足普通用户需求的云计算 2
1.1.3 资源自适应供给的弹性云平台 3
1.2 研究内容：弹性云平台的仿真优化 5
1.2.1 弹性云平台虚拟资源自适应供给 6
1.2.2 弹性云平台虚拟资源低开销仿真优化和供给 6
1.2.3 弹性云平台虚拟资源低开销迭代式仿真优化和供给 7
1.2.4 弹性云平台虚拟资源低开销进化式仿真优化和供给 7
1.2.5 弹性云平台多指标综合评价模型 8
1.3 研究意义：国内国际重大战略需求 8
1.3.1 国家战略需求 8
1.3.2 国际合作需求 9
1.3.3 本文研究意义 10
1.4 研究贡献：理论创新以及应用创新 10
1.5 本文的指导思想和应用平台选择 11
1.5.1 研究指导思想 11
1.5.2 课题平台的选择 12
1.6 论文的结构和各章内容概述 15
第 2 章 相关工作总览 18
2.1 弹性云平台虚拟资源的自适应供给 18
2.1.1 单层队列资源供给建模 18
2.1.2 多层队列资源供给建模 19
2.1.3 网络队列资源供给建模 19
2.1.4 机器学习模型 20
2.2 弹性云平台多任务敏捷管理调度 21
2.2.1 单目标管理调度方法 21
2.2.2 双目标管理调度方法 23

2.3 弹性云平台多指标综合评价模型 24
2.3.1 综合评价指标体系构建 25
2.3.2 仿真优化最优计算量分配 25
2.3.3 多目标最优计算量分配 26
2.3.4 最优计算量分配的应用 27
2.4 相关工作小结 28
第 3 章 弹性云平台资源自适应供给 29
3.1 本章引论 29
3.2 弹性云平台系统总体结构概览 29
3.2.1 弹性云计算平台简介 29
3.2.2 多层次排队网络模型 31
3.3 虚拟资源调度数学模型和求解 33
3.3.1 层次结构建模 33
3.3.2 执行算法分析 35
3.3.3 主要参数估计 38
3.4 实验设置和主要结果分析 39
3.5 本章小结 43
第 4 章 弹性云平台资源低开销仿真优化供给 45
4.1 本章引论 45
4.2 低开销仿真优化调度问题背景介绍 46
4.3 虚拟集群仿真优化调度问题困难点 49
4.4 低开销调度（LOS）仿真优化和其他方法 50
4.4.1 蒙特卡洛法和盲选法 50
4.4.2 LOS 的基本概念 50
4.4.3 LOS 的实现步骤 53
4.5 实验设置与结果分析 55
4.5.1 测试平台实验方案设置 55
4.5.2 实验结果分析 56
4.6 本章小结 61
第 5 章 弹性云平台资源低开销迭代仿真优化供给 63
5.1 本章引论 63

5.2 三种方法调度开销对比 63
5.3 LOS 方法的调度优势 65
5.4 迭代仿真优化方法 66
5.4.1 迭代蒙特卡洛法和迭代盲选法 66
5.4.2 迭代低开销仿真优化方法 68
5.5 实验结果分析 69
5.6 本章小结 71
第 6 章 弹性云平台资源低开销进化仿真优化供给 72
6.1 本章引论 72
6.2 多任务负载类型 73
6.2.1 多任务负载简述 73
6.2.2 多任务负载相似度分析 74
6.3 进化仿真优化方法 77
6.3.1 进化仿真优化方法算法分析 77
6.3.2 各种仿真优化方法的比较 80
6.4 实验结果分析 81
6.5 本章小结 87
第 7 章 弹性云平台多指标综合评价模型 88
7.1 本章引论 88
7.2 数据中心综合性能优化模型 88
7.2.1 数据中心综合性能评价指标体系 88
7.2.2 综合评价模型符号说明 90
7.2.3 综合评价模型基本假设 91
7.2.4 数据中心优化模型 91
7.3 多指标约简方法 91
7.3.1 多传感器信息融合模型 93
7.3.2 进化仿真优化方法 98
7.3.3 目标优选算法 104
7.4 实例结果分析 110
7.4.1 问题 1：中等指标规模的进化仿真优化问题 111
7.4.2 问题 2：大规模指标进化仿真优化及其可推广性研究： 113

7.5 小结 116
第 8 章 结论 118
8.1 全文研究工作总结 118
8.2 后续研究工作展望 120
参考文献 121
致　谢 131
声　明 132
个人简历、在学期间发表的学术论文与研究成果 133

主要符号对照表

HTC	高通量计算（High Throughput Computing）
FLOPS	每秒浮点运算次数（FLoating point OPerations per Second）
HPC	高性能计算（High Performance Computing）
IaaS	基础架构即服务（Infrastructure as a service）
PaaS	平台即服务（Platform as a service）
SaaS	软件即服务（Software as a service）
QoS	服务质量（Quality of Service）
CI	赛百平台（Cyberinfrastructure）
LQN	层叠排队网络（Layered Queueing Network）
TANs	增强树型贝叶斯网络（Tree-Augmented Bayesian Networks）
QOG	基于猜测的快速仿真（Quick Optimization via Guessing）
SHC	爬小山 （Smart Hill-Climbing）
CMA	协方差矩阵算法（Covariance Matrix Algorithm）
OLB	乐观负载均衡（Opportunistic Load Balancing）
MET	最短执行时间（Minimum Execution Time）
MCT	最短完成时间（Minimum Completion Time）
GA	遗传算法（Genetic Algorithm）
SA	模拟退火（Simulated Annealing）
GSA	遗传模拟退火（Genetic Simulated Annealing）
OCBA	最优计算量分配（Optimal Computing Budget Allocation）
IMSL	国际数学和统计库（International Mathematics and Statistical Library）
SLA	服务层次协议（Service Level Agreement）
LOS	次优低开销（Suboptimal Low-Overhead）
LIGO	激光干涉引力波天文台（Laser Interferometer Gravitational Wave Observatory）项目

第1章 引言

人类生产实践的复杂应用需求推动了高性能计算技术向前发展，同时又带来了更复杂的应用需求，两者相辅相成、互相促进。从 70 年代基于中低集成度主板的大型计算机（合）到 80 年代的开始平民化普及的高集成度桌面小型计算机（分），从 90 年代致力于将分布式计算资源整合的集群计算机、网格计算机（合），到如今无处不在按需供给的云计算和物联网（分）。五十年计算机体系结构的发展史，谱写和印证了“天下大势，分久必合，合久必分”一般性规律。本章将首先阐述论文的研究背景，介绍复杂应用驱动下分布式计算体系的变迁，提出当今云计算体系中弹性云平台需要解决的若干科学问题，继而并立足于国内国际重大战略合作需求，概括本文的研究意义和主要贡献，最后介绍论文的指导思想和应用背景。

1.1 研究背景：分布式计算体系的变迁

1.1.1 复杂应用催生的分布式计算

复杂科学和工程领域的应用需求促进高性能计算技术的飞速发展。分布式计算技术概念提出伊始，人们一直致力于利用分布式异构环境下的多台计算机实现高性能计算（High Performance Computing，简称 HPC）解决重大科学问题。例如 Berkeley 大学的寻找地外生物项目[1] (SETI@HOME)、高能物理方面验证广义相对论[2]的实验以及气候建模[3]等。这些问题利用目前单台计算机的处理能力几乎是无法求解的。因此我们需要把一些计算机连接起来，相互利用彼此的 CPU、存储器、数据、程序等资源，来共同解决大规模计算问题。

集群计算（Cluster Computing）技术是分布式计算早期的产物。根据功能不同，集群计算可以分为高可用集群（High-availability Clusters），负载均衡集群（Load-balance Clusters）和计算集群（Compute Cluster）。顾名思义，高可用集群的主要目的是通过冗余提供数据的多重备份以实现高度可用性；负载均衡集群则需要对大量多任务负载进行分流实现资源合理利用；而计算集群则根据集群之间的通信耦合程度不同分为经典的计算集群（Beowulf Cluster）和网格计算（Grid Computing）。紧耦合通信集群也称为超级计算（Super Computing），其本身也是应对复杂应用需求而产生的。

文献[4]将网格定义为“动态的、多机构虚拟组织之间的受控协同资源共享以及问题解决”。文献[5]针对网格环境中的虚拟组织，提出了基于协议的五层沙漏模型，该模型的初衷是提出一个类似于 TCP/IP 协议的基本框架，以便在其基础上搭建具体应用。随着商业组织不断地融入网格研究，网格技术的发展也逐渐向工业标准靠拢，文献[6]提出了以服务为中心的网格体系结构 OGSA，该体系结构是网格技术与 Web Service 技术相融合的产物，在此体系结构下，所有的资源都被建模成服务，于是服务计算（Service Computing）也伴随着网格的兴起而逐步演变成服务科学。本体系结构由于与工业界结合紧密，故而得到了工业界的大力支持，目前这一体系结构也是学术界研究与工业界应用的主流。

随着时间的推移，关注分布式计算资源的物理分离和逻辑融合，并主要致力于为科学技术领域提供应用环境的传统高性能计算并不足以满足普通用户日益增长的普及型计算要求，即用户更加关心是否能实现无处不在的按需获取计算存储网络等核心资源，关心使用起来是否能够成本更低。因此，到 2007 年底，致力于迎合大众用户需求的云计算技术得到了广泛推崇。

1.1.2 满足普通用户需求的云计算

传统高性能计算密切关注计算机的处理速度，其衡量指标为每秒所执行的浮点运算次数（FLoating point OPerations per Second，简称 FLOPS）。而在传统高性能计算机不再主要服务于科学专业用户的趋势推动下，取而代之的是一种新型的计算模式来满足大多数普通用户的需求，云计算文献[7-9]便在这个背景下应运而生。在云计算模式下，普通用户不再需要将自己的计算机和数据随身携带，而只需要将之存放到远端的数据中心里。数据中心服务软件提供接口，利用桌面虚拟化[10,11]、分布式编程和调度模型[12]、海量数据分布存储管理技术[13]等让用户随时随地可以用任何网络终端即时访问到自己的数据，而用户仅需对提供服务的资源缴纳费用。

当成千上万的用户同时访问远程数据中心时，服务提供方不再如同高性能计算一般关注每个任务的运算速度，而更多关注于其能否“海纳百川”的将越来越多的用户请求忠实的执行，此时吞吐率是他们更多考虑的性能指标。与之相对应的是，用户则会关心大部分请求（比如 95%）的响应时间是否能得到保证，其结果是否正确。所以事实上，虽然高通量计算（High Throughput Computing，简称 HTC）[14,15]在上世纪 90 年代就已经提出，但其作为一项被广泛关注的核心指标却和云计算时代的到来相得益彰。

一般情况下，云计算服务模型主要有三种形式：基础架构即服务（Infrastructure as a service，简称 IaaS），平台即服务（Platform as a service，简称 PaaS），软件即服务（Software as a service，简称 SaaS）。

IaaS 将存储、网络和计算等资源透明的租用给用户，用户根据需要指定采用虚拟资源（比如所配置虚拟机的虚拟 CPU、主存、存储设备）的类型、大小和数量来定制计算资源，提交请求后可直接获取物理和虚拟资源，并对其使用状况进行实时监控和管理，从而自由部署和规划资源。典型的 IaaS 服务有 Amazon EC2[16]、S3[17]等。

不同于 IaaS，PaaS 将不同的平台和支撑软件同时提供给用户，用户仅仅根据需要定制操作系统和基础运行平台，并关注底层软件开发、运行和测试等。相比于 IaaS，PaaS 提供了更加方便的管理方案，用户不再需要关注底层计算资源的具体配置，仅需要提供所需计算存储服务或者软件的类型和种类，则载有其所需服务的虚拟资源会被自适应的提供。Google App Engine[18]就是 PaaS 服务最典型的范例。

SaaS 则是将底层信息完全屏蔽，上层仅仅提供各种应用程序及暴露其接口，用户不需担心任何底层相关的信息，只需要根据现有的各种服务来完善和改进自身应用即可。典型的 SaaS 系统有 Google 搜索引擎和地图服务等。

从 IaaS、Paas 到 SaaS，所提供的服务抽象程度越来越高，从用户需要关注所有资源配置的细节到用户只需了解服务的使用，带给了服务提供商越来越多的挑战，即如何合理的将物理计算资源通过划分和切割为虚拟资源的方式进行有效的配置和供给。本文的关注点落脚在本方向，利用学术界和工业界广泛关注的“弹性云计算”技术构建合理的资源供给和优化方案。

1.1.3 资源自适应供给的弹性云平台

弹性（Elasticity）即使用云计算系统中各类资源时的自由伸缩性，是云计算技术中公认的从资源利用角度最重要的特点之一。顾名思义，弹性的主要特征是可大可小、可增可减的利用计算资源。加州大学伯克利分校在 2009 年发布的云计算技术报告[7]被公认为学术界定义云计算的白皮书，其中对弹性的概念进行了多次论述。弹性的主要目的是用户在选择云计算平台时不必担心资源的过渡供给导致额外的使用开销，亦不必担心资源的供给不足导致应用程序不能很好的运行和满足客户需要，所有资源将以自适应伸缩的方式来提供。这种自适应伸缩性表现在资源的实时、动态和按需供给上，即随着任务负载和用户请求的大小来弹性的调整资源的配置。

从国际最大的公有云计算服务提供商 Amazon 提供的产品来说，具有核心战略意义的 EC2(Elastic Compute Cloud) 即以“弹性”二字冠名，其意义是让云计算平台具有充分自如的可升缩性和可扩展性，而 EC2 正是本文最实验采用的计算环境之一。继 EC2 平台的推出以后，其相继推出了基于云计算的 Elastic MapReduce 编程模型，Elastic Beanstalk 创建可伸缩的应用和 Elastic Cache 缓存管理接口。可见 Elastic 对于云计算本身而言不仅是一种特征，也定义了一种趋势，未来对于云计算的理解将会直接和其可伸缩性即弹性衔接起来。

虚拟化是弹性云计算平台的核心使能技术，其通过构建在富足的物理资源（例如物理 CPU，内存，磁盘等）之上的虚拟机来实现计算资源的整合和复用。例如虚拟化技术实现物理 CPU 的最大化利用，其通过对多核 CPU 处理器上构建的虚拟机分配不同数量的虚拟 CPU 来实现，然后将虚拟 CPU 映射到物理 CPU 队列上实现分片调度[19]。如此这般在同一台物理计算机上构建的多台虚拟机就如同各自拥有独立CPU一样。在引入超线程（Hyperthread）技术的 Intel 处理器上将每一个处理内核视为物理耦合逻辑独立的两个处理线程，大大提升了虚拟化水平和调度效率。灵活配置的虚拟 CPU 数量刻画了云计算弹性资源配置的重要方面。同理，对于内存和磁盘大小等资源亦可根据应用的特征通过指定在虚拟资源的配置中灵活和弹性的设置。

虚拟化技术使得弹性的概念在云计算领域成为必不可少的优势。传统分布式计算中虽然也能在一定程度上通过物理资源的复用重组和调度实现弹性的特征，但是其对于弹性的利用程度远没有虚拟化中所构建的虚拟资源一般灵活。云计算所按需构建任意数量的虚拟机；按需将其配置具有指定大小的物理资源；按需将其调度入资源池由监控器（Hypervisor）进行统一的部署和管理；按需将其撤销以最大化资源的利用；按需将虚拟机迁移到指定的物理设备和虚拟设备之上实现数据的安全和资源的整合。所有这些都给弹性带来了极大的便利与成功，标志了云计算平台的重要特性。

虽然虚拟化能够整合并最大化计算资源的使用，但是如何最有效利用庞大的数据中心所构建的物理资源池便成为弹性云计算平台最为关注的方面之一，也是“弹性”作为重要的科学问题引起学术界和工业界重视的理由。具体而言，弹性问题的解决需要研究如下两个具体方面：

其一，虚拟资源何时供给问题。一般而言，弹性云计算平台的资源供给方式是动态和运行期供给。即随着负载大小变化在不中断所服务的同时动态增加和减少虚拟资源。于是何时供给资源的问题就显得尤为重要，其体现了

弹性云计算平台有效性问题。当然本问题受制于云计算本身的特点和负载变化的情况会有不同，是本论文工作研究的核心内容之一。

其二，虚拟资源如何供应问题。在每个供给阶段，如何配置和构建合适数量的虚拟机以最大化资源利用率并节省用户的资源使用成本也是关键的研究问题之一。Amazon EC2 中提供了用户自定义增减虚拟资源的接口。譬如 CPU 在过去一段时间如其利用率高于指定阈值则增加指定数量和配置的虚拟机，以此达到弹性的目的。但是对于高效云计算平台来说，用户一般而言是不可也不必直接干预资源本身的，所有与供给相关的任务都应该由云服务提供商来完成。本文研究工作从多个角度重点论述虚拟资源如何能够自适应的根据负载大小来提供量化供给策略。

弹性云计算进一步缩小了我们的研究背景。本论文工作在上述背景下展开，选题方向定位在分布式计算体系变迁过程里云计算中的核心技术和问题，即弹性云平台的物理资源和虚拟资源的有效管理和调度来展开。其目的在于：充分利用弹性云平台虚拟资源的可伸缩性、可迁移性等实际特点，利用虚拟集群最大化弹性云平台资源供给的效率，实现计算资源的节省和用户满意度的提升。

1.2 研究内容：弹性云平台的仿真优化

不管是高性能计算，还是高通量计算，如何构建高效可靠的数据中心以提供可伸缩的弹性服务一直是分布式计算领域面临的重要问题和核心难题[20]。一方面从云服务提供商角度出发，弹性云平台提供了资源按需供给和动态管理，可以恰到好处将物理资源和虚拟资源合理调配以及自适应规划，避免因为资源供应不足导致用户请求等待时间过长，与此同时，避免资源供给过量导致大量闲置资源空转使得系统利用率降低。另一方面从云计算平台用户角度出发，弹性云计算服务的提供不仅能降低用户对于计算和存储资源的使用成本，而且可以得到各种定制化的服务，包括虚拟服务动态迁移、虚拟资源按需分配并大大降低维护成本，提高服务效率。

本论文沿着一个问题两条路线开展。基于弹性云平台优化问题的复杂性和多样性，本研究内容将自动化领域基于仿真优化方法中的序优化[21]的思想引入到本论文工作中，从而实现了对弹性云平台的敏捷管理和调度。辅线方面侧重于理论的完善，对于仿真优化的经典方法序优化理论进行拓展，实现粗糙模型在多阶段不断智能进化算法并进行一系列与之相关的理论证明。主

线侧重于寻找弹性云计算平台下不同优化目标的需求，在虚拟资源的自适应供给、单目标、双目标和多目标综合评价模型方面为突破口，将进化仿真优化方法应用其中，验证其实验结果。

1.2.1 弹性云平台虚拟资源自适应供给

弹性云平台中既有已经完全部署在资源池中的不同规格虚拟机实例（Instance），同时也有大量的 CPU、内存、磁盘和网络带宽资源可以根据用户的命令按需生成虚拟机。在弹性云平台中，对于不同虚拟机实例的使用需要按照时间长短收取各不相同的资费。不管是基于传统分布式计算平台，还是基于现有云计算平台，相关工作[22, 23]均将后台数据中心模拟成一个多环节的排队论模型，而大量使用仿真方法得出客户端请求的响应时间和请求到达率之间的对应关系，从而依据请求到达率数值的大小来调度虚拟机数量，以提高服务质量（Quality of Service，简称 QoS），提升用户满意度，同时降低资源供给成本。

本部分工作提供了一个基于排队论的多层虚拟机管理和调度的数学优化模型，通过自适应增加或者减少弹性云平台各个层次的虚拟机资源数量，以保证用户的绝大多数请求在给定可接受的时间响应的前提下，最小化用户支付虚拟资源使用成本。

1.2.2 弹性云平台虚拟资源低开销仿真优化和供给

弹性云平台资源的自适应供给问题的解决思路是能够通过构建带约束的整数规划数学模型来满足虚拟资源供应要求，但是其主要不足在于该方法没有充分考虑到系统运行期的动态特性，诸如资源的随机可用性、任务负载的随机到达性等实际问题，从而导致该调度方法无法精准实时的捕获系统局部性、动态性和随机性。而在本虚拟资源自适应供给问题中引入随机性因素的困难之处在于对于需要仿真优化的巨大耗时，其实质是系统运行期动态性的变化需要多次仿真才能较为精确的捕获。传统基于蒙特卡洛方法太依赖于仿真次数以致特别耗时，而基于随机游走的盲选法又过于依赖初次选择空间的选择方法而获得较好策略的概率大大降低。针对以上困难的存在，我们提出了低开销仿真优化和供给方法。

本部分工作充分吸取仿真优化中的序优化思想，利用压缩一次仿真次数构建的粗糙模型获取相对满意策略集合 S，然后在相比原搜索空间小得多的 S 空间中通过还原二次仿真次数构建的真实模型获取使用阶段的优化策略，

从而缩短了调度策略的决策时间，低开销也因此得名。

1.2.3 弹性云平台虚拟资源低开销迭代式仿真优化和供给

基于低开销的思想，显而易见的问题是如何通过低开销仿真优化调度提升调度性能，而非仅仅停留在方法的层面上。由于弹性云计算平台的虚拟资源自适应供给实为序贯性、实时性、动态性多任务负载的场景设计。而多任务负载的剧烈变化对于决策的敏感性比较大。也就是说，当多任务负载变化幅度较大，而决策又不能比较实时的跟上时，整体调度性能将会急剧下降。这就好比在一个较长的宽度区间（调度时间更长，开销更大）去拟合一个复杂多项式曲线的精度会明显比在分割成较短区间（调度时间更短，开销更小）拟合差。另外一方面，弹性云平台中不同用户所产生的多松耦合大量并行任务的存在，给低开销多阶段的不断迭代优化提供了良好的契机。

本部分工作则针对以上实际问题，提出一种低开销迭代仿真优化调度方法。其实质也是利用前一章仿真优化低开销的重要思想，将其迭代的拓展到多阶段应用领域，使其连续往复多次的在任务调度中以更细的粒度予以执行决策，虽然在决策的每一个阶段以较蒙特卡洛仿真相对更少的仿真时间损失了一定的局部优化性能，但是本细粒度的调度方法能够敏捷的捕获到任务负载的局部动态性特征，从而综合提升系统的调度性能。

1.2.4 弹性云平台虚拟资源低开销进化式仿真优化和供给

虽然迭代低开销调度方法能够以更细粒度提供决策从而更好的捕获动态多任务负载的局部特性，然后在多任务负载整体而言变化并不明显，甚至多阶段无太大起伏时，以损失一定性能精度换取的低开销就明显意义不大。换句话说，如何在多任务负载变化起伏剧烈阶段自适应的以低开销方式供给虚拟资源，而在其变化比较平缓时就相应引入更长的仿真时间来换取仿真精度是本章拓展的一个方向。

本章我们介绍弹性云计算平台资源低开销进化仿真优化供给方法，其优点是在多任务负载连续的各个阶段分析其变化规律，从而自适应的实现调度各个阶段仿真时间长短的灵活选择。在多任务负载变化剧烈的阶段，本方法自适应的切割1.2.3中提及的迭代低开销仿真优化的调度时间阶段，从而对于突变多任务负载进行细粒度分析与调度；而对于多任务负载变化比较微弱的调度阶段，本方法自适应的将调度时间阶段进行合并，从而为后阶段换取更多的仿真时间并提升调度性能。

1.2.5 弹性云平台多指标综合评价模型

在面对大规模多层级的综合指标评价体系问题中，如何选取合理的评价指标子集合从而具有对于整个综合评价系统具有最有效的评价性能是非常重要的科学问题。其难点在于对于每个指标的评价过程特别耗时，利用仿真优化方法进行评价时本难点尤甚。如何利用有限的仿真实验时间最大化获取评价指标之间存在的正相关和负相关性的信息，同时构建合理的评价指标体系和模型来取代原始评价体系则是本部分需要重点研究的内容。

1.3 研究意义：国内国际重大战略需求

本论文工作受国家科研项目以及大型跨国合作项目的资助，立足于国内国际重大战略和合作需求，放眼行业内部重要研究方向，密切紧跟世界知名学术界和工业界前沿研究成果，进行了一系列研究工作。

1.3.1 国家战略需求

高效能、低成本、低功耗一直是我国信息技术发展和前进的指导性方向。由国务院在 2006 年初发布的《国家中长期科学与技术发展规划纲要》对我国 2006 年至 2020 年的基础研究工作进行了高度的总结和精确的凝练，其中明确指出重点研究开发针对教育、医疗、传媒、电子商务和电子政务等关乎民计民生各个领域的网格计算平台与基础设施，而与弹性云平台直接相关的虚拟化技术被定义为符合国家中长期重大战略需求并与之相适应的研究方向，具有重大的经济价值和社会价值，对发展国家经济、推动社会进步、提升国家核心竞争力具有战略性、全局性和长远性意义。

同时，最近发布的国民经济和社会发展第十二个五年规划纲要中，将绿色发展作为我国产业转型的前瞻性关注领域。由此可见，绿色经济将会成为我国在未来主要的经济增长方式。弹性云平台计算资源自适应供给本着资源的按需供应、兼顾计算资源消耗成本和终端用户满意程度为优化目标的研究方向，是适应绿色计算节能减排降耗趋势的，可以有效缓解国家计算资源浪费、投入产出比过低等一系列矛盾，为信息产业的发展与进步带来明确的借鉴性意义。

本论文提出开展弹性云计算平台优化方法作为研究方向，就是本着以国家重大战略目标为指导思想，以节省计算资源、优化计算性能、改进计算模式、提升计算效率为实施依据，以完善计算基础设施建设为根本目标，完成

和解决云计算模式下一系列重要问题和核心问题。

1.3.2 国际合作需求

美国国家科学基金委员会斥资数亿美金的激光探测引力波天文台[24]（Laser Interferometer Gravitational Wave Observatory）项目致力于寻找宇宙中引力波存在的证据，并利用其更好的揭秘复杂的天体现象。根据爱因斯坦的广义相对论，引力波是由宇宙中高密度天体剧烈相对运动而产生的时空涟漪，这种涟漪以波的方式从发生地向外传输，对其有效捕捉、获取和分析能够更好的了解天体运动的起源和重力的本质等问题。虽然至今为止科学界还没有直接探测到引力波的存在，但是物理学界通过测量引力波对自旋中子星的影响，发现其与爱因斯坦的广义相对论预测一致，于是科学界更加坚定了引力波的存在并致力于寻找发现其存在的证据。

作为 LIGO 科技合作组织（LIGO Science Collaboration，简称 LSC）的国际成员之一也是唯一中方成员，由清华大学信息技术研究院曹军威研究员带领的科研团队致力于引力波数据分析的科研工作，笔者有幸成为 LSC 组织成员之一加入其合作中。每日由天体产生和引力波天文台收集到的海量数据和需要后期处理的复杂多任务分析，使得分布式计算技术成为支撑其应用的重要使能技术之一。探测引力波的一项重要任务是实现引力波候选信号和噪声信号的有效分离，而为了保证客观性和有效性，整个分离过程中必须满足施加在探测仪端的若干关键业务逻辑的约束[25]。而对于此类业务逻辑约束的验证过程则是一个典型的适合在弹性云计算平台下展开分析的多任务计算的应用场景。基于此，我们积极的参与到 LIGO 项目中，致力于采用先进计算技术协助物理领域科学家进行有效的数据分析。

除此以外，纵观工业界和学术界近五年的研究内容，对于信息技术领域的研究仍然是最为关注的焦点，以至于国际重大专项课题中最集中资助的对象包含大量云计算领域的研究。美国知名市场研究公司 Gartner 每年 10 月都会发布一份今后三年内值得关注新技术的预测报告，在 2011 年国际十大领先战略技术中，云计算位居首位。

在国际竞争愈演愈烈的现实背景下，只有符合国际主流方向的科学研究课题才能获得一席之地。本文在国际重要合作项目的需求驱动下，对于弹性云平台关键技术进行纵览和分析，解决了本领域中若干重要实际问题。

1.3.3 本文研究意义

科研工作的意义在于立足于现实并为现实服务。在国内国际云计算科学不断发展，中国云计算相关科研项目不断推进的现实情形下，本文科研工作服务于国家重大战略需求，其现实意义有如下几点：

其一：本科研工作的理论创新之处，即低开销进化仿真优化方法，提出了在求解问题的多阶段不断学习多任务负载动态特性，从而更好的为后阶段提供更加高效的仿真模型，对于仿真优化方法本身而言是一种重要的推进和延伸。它补充了仿真优化方法在进化思路下对于模型的选择方向，使其更好的运用于多阶段仿真优化问题的实施，优化仿真结果，提升仿真效率。

其二：本科研工作的应用创新，即弹性云平台的仿真优化问题本身，在分布式计算领域鲜有前人涉足。大多数以往研究工作并未将大规模随机因素考虑在内，而是以估计其平均值为典型方案来执行和实施优化调度工作。而弹性云平台运行期的动态性问题的存在导致估计平均并不能如实贴切的反映系统的真实性能，其弊端也不断呈现。本工作致力于应用仿真优化方法真实的捕获系统运行期的动态性特征，利用敏捷资源供给和实施调度方法来快速提升系统性能。本工作弥补了分布式计算系统在仿真优化方面的空白，在面对更加复杂的计算系统形势下提供指导性意义。

其三：本科研工作得到国内国际重要科技项目的支撑。其提炼面向构建云计算的弹性运行环境需要解决的基本科学问题，在物理和虚拟系统的软硬件实现与云计算应用之间搭建桥梁，更好的提升系统性能基础和管理效率。在“十二五”规划的开端，立足于国家对于高效能计算技术高度重视的客观实际，同时借鉴国内外现有研究和科技成果，将理论创新和应用创新两者有机的结合起来，使之更好的为国家科技进步服务。

1.4 研究贡献：理论创新以及应用创新

在分布式计算技术日新月异更迭交替的客观背景下，在发展国家高效能计算机、建设计算基础设施的重大战略需求指导下，立足于国际研究热点问题，本工作的核心成果可以概括为理论创新和应用创新两大方面。

首先，从基于仿真优化的理论问题和方法出发，本工作创新的提出了多阶段迭代仿真优化和进化仿真优化方法。将传统仿真优化方法拓展到连续多阶段决策领域，实现了多次决策的敏捷控制。多阶段迭代仿真优化方法在决策的每个阶段均能更加细粒度的捕获和分析系统的局部信息，从而使得控制方案能够更好的提

升系统的综合性能。而多阶段进化仿真优化方法则是多阶段迭代仿真优化的延伸，通过学习多任务负载本身的特性来自适应划分或者合并调度决策阶段，从而指导后续阶段作出更加敏捷和高效的决策。

然后，从基于弹性云平台虚拟资源自适应供给角度，在给定部署了大量虚拟资源的可伸缩云平台下，创新的建立了用户请求的响应时间和请求到达率之间联系的数学模型，并与之相适应的提出模型的数值求解思路和方法。在基准测试程序下的实验结果显示，本模型计算出的响应时间能够忠实的反应真实情况。同时，本模型亦能给出在多任务负载不断变化的情形中，最优化虚拟资源供给的方案。

其次，从弹性云平台多任务敏捷管理调度角度，在面临巨大的搜索空间并辅以存在大规模不确定性参数的评价优化问题中，创新的利用多阶段迭代仿真优化方法，实现了云平台在满足高通量低成本的前提下的弹性调度。该弹性的作用在虚拟资源的科学性、灵活性和动态性的供给中，实现多重性能目标的协同优化。

再次，从基于多任务负载时缓和变化时急剧变化的实际情况，在具体分析其变化规律和共性以后，创新的提出自适应切割和合并多任务负载的低开销进化式仿真优化方法，作为低开销迭代式仿真优化方法的升级版，本方法能够更好的适应负载变化规律。利用其低开销调度算法，真正实现了同步优化和实时优化。

最后，在弹性云平台多指标综合评价模型角度，在面临多重维度大规模耗时的仿真优化问题中，利用本文所设计的低开销进化仿真优化方法，能够在有限的时间内获取到效果较好的综合模型，从而完善了粗糙模型的进化设计理论方法，补充了美国佐治亚梅森大学 Chun-Hung Chen 教授和新加坡国立大学 Lee Loo Hay 教授在最优计算量分配[26]（Optimal Computing Budget Allocation，简称 OCBA）方法在多指标维度的优选（原始方法的优选构建在策略维度，详细论述见第二章相关工作），并最终实现低开销仿真优化方法的进一步完善。

1.5 本文的指导思想和应用平台选择

1.5.1 研究指导思想

本论文的研究遵循的主要指导思想是问题驱动，即探寻云计算技术领域的若干实际问题，并思考其数学模型的表达和性能优化的需求，以此为基础探讨提升优化性能和完善优化方案的各项措施，具体如下展开。

基于分布式计算系统弹性云计算平台的客观需要，建立底层虚拟资源自适应供给和上层多任务敏捷管理调度为一体的工作体系。鉴于问题的客观复杂性本身，尝试性的将工程领域的仿真优化方法应用其中，拟定仿真优化的

整体框架、探讨仿真优化的实施步骤、综合仿真优化的实验结果。以敏捷资源供给为导向，以系统综合性能提升为目的，以多阶段进化优化为手段，实现弹性云计算平台中各种物理虚拟资源的合理配置按需供应。

同时，基于以上问题产生的客观需要，在序优化理论以及与延生其理论范畴的一系列现有工作中，针对弹性云平台实际存在的若干问题和真实特性，在理论上对其 OCBA 方法本身进行创新、拓展和完善，使其定制在弹性云平台资源优化的应用范畴，在本领域的若干重要和核心问题中发挥作用。

1.5.2 课题平台的选择

本节重点介绍整个论文工作所选择应用介绍，其分为两部分。其一为两类典型的弹性云计算平台本身的选择和使用；其二为弹性云计算平台的基准测试程序的选取和多任务计算用例的设计。

（1）弹性云计算平台的选择

本论文工作选择了两类典型的弹性云计算平台，一类在能够自主控制物理资源的 IBM 大型服务器上构建虚拟化设备的云计算平台上实现；另一类在最广为使用的商业领域亚马逊弹性计算云（Amazon Elastic Compute Cloud，简称 Amazon EC2）上实现。

在图 1.1 中，展示了弹性云平台设计方案在北京 IBM 中国开发试验室上的 10 台服务器中展开的示例。每台服务器的配置是 Intel Xeon MP 7150N 处理器，24 GB 的内存以及 20TB 的硬盘做存储，其中所有物理结点以及构建的虚拟服务器集群设置方案如表 1.1 所示。

每台物理服务器上安装了 14 台虚拟机，物理宿主操作系统采用的 OpenSuSE11/OS，所有的实验验证程序由 Java 编写。在 10 个结点的物理集群中，我们总共构建了最高达 128 个虚拟机的实例以测试论文中的方法在各项应用指标上的性能。测试从 16 个结点到 32、64、128 个结点分别开展。

表1.1 虚拟物理集群配置

集群规模	由10台服务器构成的物理集群
集群架构	IBM X3950 服务器由 16-core Xeon MP 7150N处理器， 24 GB 内存搭以 OpenSuSE 11 操作系统
虚拟机配置	虚拟机分配1个虚拟CPU和1GB内存，操作系统为 OpenSuSE 11
虚拟机构建	每台物理服务器上构建14个虚拟机

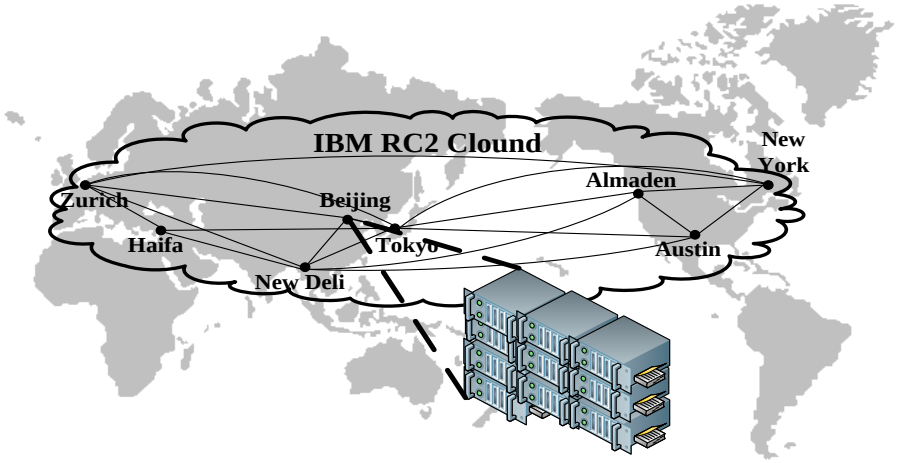


图 1.1 IBM 构建在全世界的 8 个研发中心的研究计算云平台（Research Compute Cloud，RC2）本实验在北京开发中心开展

为了使得实验部分的工作更加完整，我们在后阶段采用了学术界和工业界广为使用的 Amazon EC2 弹性云计算平台。由于该公司在公有云计算方面积累了多年的经验，其弹性云计算平台对于虚拟资源的供给效率相当高，使用起来也比较容易上手。其推出了基于各种开发语言的编程接口能够自如的按需创建、暂停、停止、撤销虚拟机的调用和运行。

采用的虚拟机数量同样从 16，32，64 直到 128 个小实例（Small Instance）。每个虚拟机实例采用缺省配置即 1.7 GB 内存容量，1 个 EC2 计算单元（即每个虚拟机一个虚拟 CPU），160 GB 的本地存储容量，部署在 Linux AMI Base 的 32 位操作系统，EBS boot，32 位的 Amazon EC2 AMI 工具集。

Name	Test10Inst	Instance	AMI ID	Root D	Type	Status	Security Gr	Key Pair Na	Monitoring	Virtualizati
empty		i-1e8693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-1c8693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-1a8693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-188693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-168693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-148693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-128693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-108693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-ee8693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-ec8693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-ea8693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-e88693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-e68693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-e48693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual
empty		i-e28693	ami-8c1fece5	ebs	m1.small	running	default	fzhang_sshk	basic	paravirtual

图 1.2 Amazon EC2 实验环境搭建

（2）基准测试程序的选取

基准测试程序 RUBiS (Rice University Bidding System)[27]是由美国的 Rice University 开发的的多任务计算系统，其类似于 eBay 的交易平台，提供了包括竞标、预定、购买等真实交易类型。

RUBiS 的主要架构为多层分布式，第一层为基于 Apache 的 Web 服务器提供静态页面使能；第二层为基于轻量级 Tomcat 的应用程序服务器提供 Servlet 支持、同步会话、管理业务逻辑并提供对后台数据库访问的接口，最后一层为基于 MySQL 的开源数据库服务器层，提供对所有数据、中间状态、会话状态的存储、读取和管理。其结构如图 1.3 所示。每一层均由若干个虚拟机组成。例如 Web 服务器和数据库层有两台虚拟机而应用程序服务器有三台虚拟机。每一层虚拟机数量可以自适应伸缩以弹性提供虚拟资源。

本基准测试程序提供开源代码支持，可以根据需要对其各项交易功能进行改进，提供了基于 JSP、PHP 等多种脚本语言版本。同时，客户端模拟器能够产生终端用户的登录和执行与各种网上购物相关的的脚本并提供不断变化的用户请求负载的强度来测试弹性云计算平台的自适应伸缩性。本部分工作的实验建模和测试验证过程我们会在第三章具体讨论。

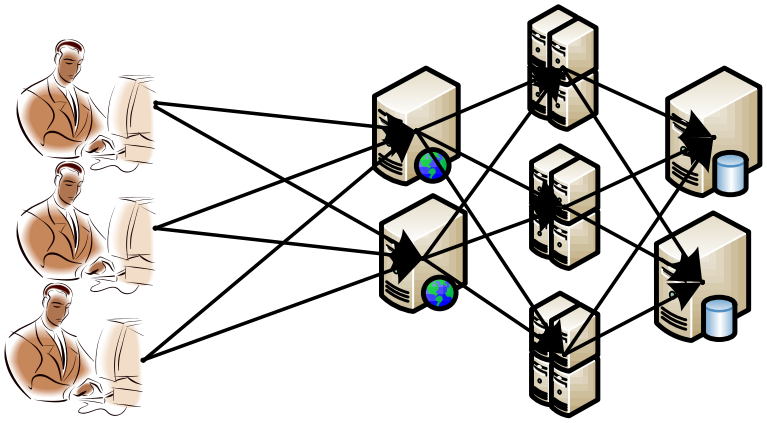


图 1.3 RUBiS 基准测试程序在弹性云计算平台上的的三层架构

如果说上述基准测试程序提供了基于商业应用的案例，下述两类基准测试程序则是基于科学计算领域的实际需求。在 1.3.2 节中提及的引力波探测数据分析中对于探测仪端的若干关键业务逻辑的约束则是典型的多任务计

算的具体应用。我们抽取出七类重要的验证逻辑，每一类验证逻辑根据需要设计成具有多次仿真验证试验的多任务多任务负载来实现，本部分内容我们会在第五章中详细讨论。

另外一类基于科学计算应用领域的基准测试程序是一系列非常典型的科学计算用例，在云计算数据处理的各项应用中都有广泛的用武之地。在第六章中我们选取了 7 类典型应用，主要参考文献[28]的工作，其根据类别可以划分为 CPU 密集型计算、内存密集型计算和 I/O 密集型计算三类：1）比特字节的反转，在网络信号加密中使用得非常广泛；2）合并排序，在基于 MapReduce 的大规模数据并行处理中用得多；3）矩阵乘法，在各类商业化应用中均有其存在，通过控制矩阵的规模来调整多任务强度；4）大数因式分解算法，在数据信号解密中应用得非常广泛；5）k 临近聚类算法，在数据挖掘中应用的较多；6）Hash 算法和 7）运筹学的动态规划算法。

1.6 论文的结构和各章内容概述

本文通过以下七个章节，以弹性云平台虚拟资源动态供给为背景，以仿真优化方法的迭代和进化为思路，自底向上的将论文组织成如下七个章节，如下一页图 1.4 所示。

第一章（即本章）以分布式计算体系变迁为研究背景，概括性的提出了论文的主要研究内容。然后从国内和国际的重大战略需求出发，结合学术和工业界的研究热点，论述本文工作的定位和实际意义。接着从理论和应用两方面论及了本文的创新所在，并提出支撑本文创新的指导思想和项目背景。

第二章分为两部分。第一部分对与弹性云平台相关的各项已有工作进行了综述和分析，指出本论文工作与其他相关工作的异同点，并客观分析本弹性云平台资源管理的必要性。第二部分对序优化方法中的最优计算量分配问题进行了相关工作的综述，主要核心关注点在最优计算量分配方法以及与之相关的拓展工作，并提出本文进化仿真优化方法和以上拓展方向的异同，并探讨工作的的必要性。

第三章主要研究弹性云平台的约束模型调度方法：对虚拟资源的自适应供给问题进行了比较深入的探讨，建立分析多层次虚拟资源敏捷供给的数学模型，并介绍模型的求解方法和实施思路，对比了本方法和现有相关方法的异同之处，并在真实三层云平台环境下搭建了系统实施方案，验证了本模型的正确性和有效性。

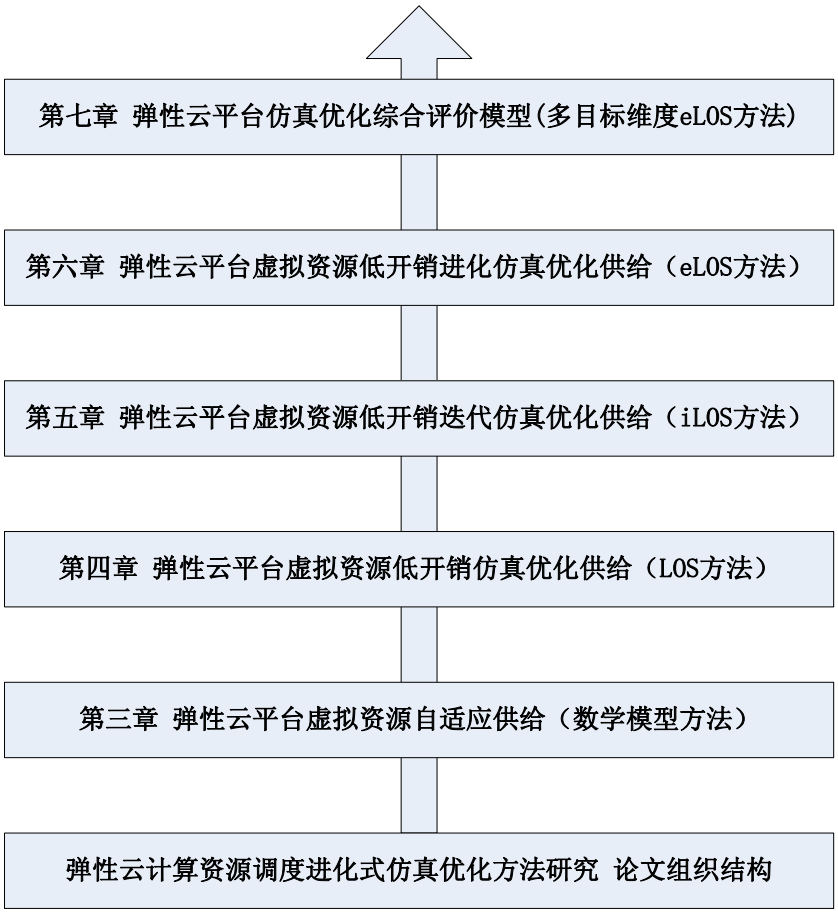


图 1.4 论文组织结构

第四章承接第三章模型优化方法，研究具有更加复杂性的弹性云计算平台虚拟集群运行期随机因素的特性，从而引入仿真思想来优化系统性能。由于仿真优化模型运行期的耗时性问题存在，本章中我们提出低开销调度思想，采用粗糙模型降低每次仿真的时间，从而实现虚拟资源的敏捷和快速供给。最后用不同数量的虚拟机在数量上证明本方法的低开销性优势。

第五章承接和利用第四章仿真优化方法的低开销优势，针对动态时变的多任务负载进行仿真优化。其实质在于建立多阶段优化模型，并拓展低开销仿真优化方法，利用其短仿真耗时的益处，将其反复多次迭代执行从而实现虚拟资源的敏捷决策和供给，以适应动态变化多任务负载强度，从整体上提升系统的调度性能。

第六章继续承接第五章的多阶段低开销迭代仿真优化方法，自适应的切割或者合并第五章中低开销迭代仿真优化的调度时间阶段。本方法有意引入噪声，从而对于突变多任务负载进行细粒度分析与调度；而对于多任务负载变化比较微弱的调度阶段，本方法自适应的将调度时间阶段进行合并，从而

为后阶段换取更多的仿真时间降低仿真噪声并提升调度性能。

第七章继续承接第六章的多阶段低开销进化仿真优化方法，将其应用到弹性云平台的综合评价模型。首先本章建立进化仿真优化方法的核心框架并提出其在多目标仿真评价优化问题中的实施算法过程。然后，根据弹性云平台的大规模目标和策略的多次仿真实际问题出发，构建大量目标评价性能指标的进化仿真优化方法，最后在实验中验证本方法在仿真目标约简和提升评价效率方面的有效性。

第八章对全文进行总结，并提出文章的不足之处以及需要进一步研究和改进的方向。

第2章　相关工作总览

围绕弹性云计算平台资源管理展开的研究工作不胜枚举，但是由于关注的目标层面、研究思路方法的各不相同使其与我们开展的研究路线有所不同，但是诸多已有方法和实验结论却能为我们开展的工作提供良好的借鉴与启示。本章各节分别从弹性云计算平台虚拟资源自适应供给、多任务敏捷管理调度和多指标综合评价模型入手，对已有工作进行调研，并在最后给出相关工作研究现状的总结。

2.1　弹性云平台虚拟资源的自适应供给

目前对于弹性云平台资源供给方法的相关工作主要集中在对于模型表达方式上，可以分为单层队列资源供给模型、多层队列资源供给模型、网络队列资源供给模型和基于机器学习方法的供给模型，下面对这几部分分别做详细介绍。

2.1.1　单层队列资源供给建模

在文献[29]中，Web 服务器由四重队列来进行建模，其中两个对列用于对 Web 服务器本身建模，另外两个对列用于通讯网络建模。本建模方法在服务器类型繁多的今天自有其局限性。文献[30]中将 CPU、内存、磁盘和带宽分别建模，根据其对文件处理速度来进行逐个分析，以提高性能预测的精度，本建模方法在弹性云平台下并不太适用，其原因在于其构建资源模式建立在传统物理资源上，而如今更注重虚拟资源的大规模建模方式。在文献[31]中，经典控制论方法用以控制 Web 服务器性能的观点被提出，并带有基于 Apache Web 服务器的性能测试。文献[32]构建可复制单层 G/G/1 排队模型来处理应用请求，类似在文献[33]中利用 M/M/1 排队论模型来计算网页请求的响应时间。和以上两篇论文类似，文献[34]提及单层排队论模型外加马尔科夫链来计算网页吞吐率。以上文献对于单层服务资源进行供给，远不能满足在云计算平台下虚拟资源的按需供给的应用需求，于是多层资源供给建模方式也随后出现。

2.1.2　多层队列资源供给建模

在多层资源供给建模工作中，文献[35]首先提出采用 G/G/N 队列方式，构建具有 N 个服务台的商业应用平台。随后文献[36]提出采用 M/GI/1/PS 模型构建商业应用，与前者不同的是，本文建议采用单队列来建模多层系统，其原因在于网络系统除了瓶颈环节以外通信开销一般较小，则主要对瓶颈环节进行建模分析。文献[37]随后又提出类似的方法，不过采用的是 M/G/1/PS 模型。以上文献的共同特点是均着重分析瓶颈环节的开销，这样做并不能从全局上把握整个多层系统的特征，模型表达不够准确。

近几年，这方面的工作被广泛关注。引用最高的论文是 Urgaonkar B. 教授所提出的模型，采用多层排队论方法对网络宿主系统的模型利用仿真方法进行建模。在文献[38,39]中模型综合考虑了网络中的负载均衡（Load Balancing）、处理并行请求上限限制（Handling Concurrency Limit）、多会话类（Multiple Session Classes）等，实验结果反映出本模型能够比较真实的获取用户请求到达率和响应时间之间的逻辑关系，达到了良好的效果。随后，一种增强模型由同一作者再次在文献[40,41]中提出，采用了预测（Predictive）和反应（Reactive）相结合的方法对负载进行详细分析，以确定在什么时候如何提供合理的虚拟机数量满足用户请求。

和以上工作相同的是，本论文工作同样采用多层次排队论方法进行建模，同样在队列的每一层自适应提供一定数量虚拟机。和以上工作不同的是，本工作不仅考虑了使用虚拟机的调度成本，采用了不同规格的虚拟机实例作为调度单元。因为在诸如 Eucalyptus，Amazon EC2 等大型云服务提供商的产品中，虚拟机实例具有多种多样的配置，而面对多样配置虚拟机综合调度方法比传统单一配置虚拟机调度困难许多。

2.1.3　网络队列资源供给建模

不同于单层和多层队列资源供给模式在表达服务模型提供方面的不足，有学者提出利用复杂的网络队列模式来进行建模，而层叠排队网络（Layered Queueing Network，简称 LQN）[42-46]就是其中的一种。以上几篇文献有两个类似之处，其一：采用 LQN 将软件服务器放置于整个网络的最上层，然后将其下层的诸如 CPU、内存、磁盘和网络等资源映射成相应的层次结构，从而为复杂的网状任务请求提供多样化的服务。其二，更多的关注应用程序服务器层的资源提供方式，以此组织程序的布局和服务的提供，而对于进行负载均衡 Web 服务器和后端数据库服务器在模型中关注得非常少，以致于多

层网络模式虽然更加合理，但是不太符合现在 Web 服务器、数据库服务器和应用程序服务器相分离的实现模式，更不符合实现虚拟机实例实时迁移的备份服务器检查点更新的多服务纵向管理模式。

基于数值求解的网络队列模型方法在文献[47]中提及。本工作预测两层的 SPECjAppServer2002 系统的性能，并用开源软件进行求解数学模型来实施。在文献[48]中提出了网络队列模型结合搜索方法，并在不同的负载强度下进行了测试，其工作的主要贡献在于适用的估计均值模型可以很好的解决可扩展性问题，并适应于不同的应用场景。在文献[49]中，多种方法均用来评估网络队列模型的性能，包括估计均值方差分析方法、马尔科夫链方法等等，其主要目的还是为了利用此方法搜寻网络队列中瓶颈最严重的阶段，以进行资源供给。和以上相关参考文献不同的是，我们的方法首先适用于虚拟资源的弹性管理，即应对的是动态资源的敏捷调度范畴；其次，原方法并未考虑到使用资源的成本问题，而我们的方法两者兼顾。

实际上，所有这些相关工作内容给本章的研究奠定了良好的工作基础。在复杂数据中心的拓扑结构中，对于虚拟资源的动态管理绝非传统单层或者多层模型下简单的近似特征能获取的。复杂的网络拓扑结构在某种程度上更能体现现有互联网络[50]部署虚拟资源的实现方式。鉴于这个特点，在第三章中介绍的弹性云平台虚拟资源的自适应供给方法中，将采用网络队列资源供给模型来实现。整体上看，是多层虚拟资源供给模式，而在虚拟资源供给的每一层，均可自由伸缩，弹性增减，实质上是以复杂网络结构提供。

2.1.4　机器学习模型

在文献[51]中提到了增强树型贝叶斯网络模型（Tree-Augmented Bayesian Networks，简称 TANs），用于辨识系统级性能指标的参数值，从而获取满足服务层次协议的高性能状态的取值。实验在多种网络环境下展开，结果表明本方法适用于离线故障诊断和在线性能预测。在文献[52]中，一种机器学习的方法被用于推断所有应用层单层的性能。同时，一种协调预测器用于在多层之间预测系统级性能，同时也能辨识出系统变得负载过重时的瓶颈所在，实验结果同样显示出本方法能够以很高百分比的准确率估计出网络流量和负载的运行期数值，并且性能开销很低。类似的工作可以体现在文献[53]的研究中，采用基于回归方法的分析模型对多层应用系统进行了分析。首先，该方法采用基于回归的思想估计 CPU 在某特定客户事务中对资源的

要求，然后在带有不断变化负载的网络队列模型中分析本方法的有效性和正确性；最后通过两种截然不同的负载变化方案的实例，证实了本回归方法能够比较好的支撑和预测对于网络需求的规划和资源的合理供给。文献[54-58]根据多层次排队论模型利用机器学习的思想进行了配置参数调优。

基于机器学习的网络性能预测方法在实验中表现出比较良好的性能，是预测多层网络模型性能和提供资源供给方案常用的方法。但是到目前为止，基于机器学习的网络性能预测和资源供给方法并没有很好的在工程领域发挥重要作用，其中最核心和本质的原因是其可扩展性不强。在规模较小的数据中心资源中，其能体现出优越性；而当数据中心中的物理服务器被动态切割成若干虚拟机实例，并且虚拟机实例被动态组合来执行不同任务请求时，问题的巨大求解空间导致本方法存在相当大的局限性。

在第三章的方法中，通过多层次网络队列方法直接建立数学模型来获取响应时间和用户请求到达率之间的关系，不仅具有良好的扩展性，其资源供给性能也能比较良好的捕捉和跟踪负载动态变化的特点，实现资源的敏捷供给。

2.2　弹性云平台多任务敏捷管理调度

如前所述，云计算平台下最显著特点是大量松耦合多任务需要实时执行。基于此，高通量多任务调度在于如何将物理计算资源进行有效的切割、划分和重组，使得构建在其上的虚拟资源能够弹性和自适应的为多任务分配合适的计算资源，满足各项衡量指标的要求。本节从单目标、双目标出发分别给出传统网格资源调度领域耳熟能详的相关工作。

2.2.1　单目标管理调度方法

传统高性能网格计算平台，由于其不像云计算平台一样具有灵活的弹性和可伸缩性，一般以静态调度方法较多。常见的方法有被 GrADS[59]项目所采用的最小最小（Min-Min）算法[60,61]、最大最小（Max-Min）算法[60,61]和容忍（Sufferage）算法[60,61]，有被 Pegasus[62]项目采用的加权最小最小启发式算法（Weighted Min-min Heuristic Algorithm），有被 Triana 项目[63]中的 Taverna[64]调度器使用的即时调用策略（Just-in-time），有被 UNICORE[65]和 Gridant[66]项目采用的手工调度策略等。在文献[67]中，将 11 种启发式方

法（Opportunistic Load Balancing，Minimum Execution Time， Minimum Completion Time，Min-min，Max-min，Duplex，Genetic Algorithm， Simulated Annealing，Genetic Simulated Annealing，Tabu 和 A*）进行了介绍并作出横向对比，结论是在所测试的实验场景中，简单的 Min-min 方法相对获得了更好的测试性能。

这里将以上论述的 11 种方法整理如下：

（1）Opportunistic Load Balancing[68-70]，简称 OLB：也叫乐观负载均衡方法，OLB 方法将每一个任务以任意次序赋给下一个可用的服务器，而不关心在本服务器上的执行时间。本方法优点是简单，缺点是可能产生很差的调度策略。

（2）Minimum Execution Time[68-70]，简称 MET：也叫最短执行时间方法，MET 方法将每一个任务以任意次序赋给执行当前任务时间最短的服务器上，而不关注本服务器是否当前可用。本方法优点也是简单，缺点是可能造成严重的负载不均衡。

（3）Minimum Completion Time[68-70]，简称 MCT：也叫最短完成时间方法，MCT 方法将每一个任务以任意次序赋给完成当前任务时间最短的服务器上，这种分配方案使得某些任务并不是分配到其执行时间最短的服务器上。本方法结合了（1）和（2）的优点，但是实施起来相对复杂一些。

（4）Min-min[68,69,71]：本方法初始于整个未处理任务集合 T，对于 T 中的所有任务，将具有最小完成时间的任务，服务器对（t_i，s_j）挑出，将本任务 t_i 赋给本服务器 s_j，同时将 t_i 从 T 中移除，本过程反复执行直到 T 为空时停止。本方法相对于 MCT 的好处在于其每次都综合考虑了所有未处理任务集合，一般情况下性能要优于 MCT 方法。

（5）Max-min[68,69,71]：本方法同样初始于整个未处理任务集合 T，对于 T 中的所有任务，将具有最小完成时间的任务，服务器对（t_i，s_j）一一列出，然后选择这一列中具有最长完成时间的（t_i，s_j）作为初始分配方案，同时将 t_i 从 T 中移除，本过程反复执行直到 T 为空时停止。相对于（4）中的方法，本方法的好处在于能够处理任务完成时间差别非常显著的任务集合。

（6）Duplex：本方法结合了 Min-min 和 Max-min 两种方法的优缺点，其同时使用这两种方法做比较，选择较好的方案。

（7）Genetic Algorithm[72-76]，简称 GA：GA 将任务分配方案的染色体进行编码，编码在第 i 位的代表第 i 个任务，其取值代表分配服务器的方案。初始方案共 200 组，通过 Min-min 方法产生一组方案和随机方法产生 199 种

方案。对应于每种染色体编码方案均有其适应度取值，用所有任务的完成总时间来表示。GA 通过在本适应度函数下进行不断的交叉、复制和变异一直选出性能优良的任务分配方案。

（8）Simulated Annealing[77,78]，简称 SA：SA 采用迭代“降温”的方法进行优化。其定义温度为任务的总执行时间，每次迭代的方法是使得温度降低的方向。其初始设置和 GA 一致，均是采用随机分配的染色体编码方案来表达任务分配，其变异方式也和 GA 一致，只不过是变异后的新编码方案是否被接受或者拒绝取决于新的温度表达式，在相关文献中有非常详细的论述。

（9）Genetic Simulated Annealing[79,80]，简称 GSA：GSA 方法实际上是 GA 方法和 SA 方法的混合。其实施流程大部分和 GA 类似，所不同的是，GSA 采用 SA 中定义的温度和降温等方法来决定是否接受当前染色体编码的任务分配方案。

（10）Tabu[81]：也称禁忌搜索，其搜索方法跟踪已经搜索过的任务分配策略区域，其初始任务分配染色体编码方式和 GA 一致。其实施步骤是，对于给定任务对（t_i，t_j），将其分配到所有的服务器对（s_i，s_j）中，而此时其他所有（T-2）个任务都按照初始预定方式分配到相应服务器不作变化，如果当前搜索到的新分配方案比原始的要好（此过程称为一次替代），则采纳新分配方案并以其作为初始状态重新执行上述过程，直到替代次数达到设定的阈值为止。

（11）A*[82-85]：本方法为任务分配方案构建 l 维树状结构，根节点没有任何分配方案，任何孩子节点是父节点添加了任务 t_a 以后的部分分配方案。所以第一层节点是仅仅具有单一任务分配方案的部分方案。为了使得树状结构不至于无限蔓延，本方法设置了剪枝策略，即当节点数量达到最大阈值时，每加入一个新的部分方案节点，就将当前的分配方案中估计任务总执行时间最长的部分分配方案去掉。对于 A*方法中如何设置和估计部分方案的总执行时间下界 $f(n)$，实际代价 $g(n)$以及最优后续方案估计代价 $h(n)$，可以参照文献[86]中的步骤，这里不再详细介绍。

2.2.2　双目标管理调度方法

大部分双目标管理调度方法均是在传统网格数据流中应用，而其优化的主要目标无外乎性能、成本、可靠性等相冲突的性能指标，即如何保证在以较少资源的使用为前提下尽量提升调度性能和可靠性，而一般调度性能以所有任务总完成时间表达居多。文献[87,88]首先定义了异构计算环境下双目标

调度的数学模型，然后提出了两个不同的算法，即双目标动态层级调度算法和双目标遗传算法，两个算法互为补充，可以在总执行时间和可靠性两个指标之间进行合理折衷。在文献[89]中，考虑了网格环境下多任务的目标总完成时间和资源使用成本两个冲突的目标，本方法有效的将静态启发式调度和动态再调度（Rescheduling）技术结合在一起，并用实验证实了方法本身的有效性。

文献[90]中，额外考虑了执行任务的处理器存在一定的失效性问题，即定义指数分布设定处理器在一段时间内以一定的概率失效，目标同样是规划总任务的执行时间最小化和可靠性最大化。本文的主要贡献是让用户在这两个性能指标中自行选择，从而确定相应的调度方案。一种动态受限算法在文献[91]中提出，和其他双目标调度思想不同，本算法确定优先调度目标和次优调度目标。在调度过程中首先满足优先调度目标，在其可活动的阈值范围以内，考虑次优调度目标的优化问题，从而寻求两个目标之间的权衡。基于大量多任务实例和性能指标的搭配构建的实验环境验证了本方法的有效性。而文献[92]则将多目标问题转化为受限约束问题，即在调度中将失效性转化为硬性约束来进行综合调度。

还有将多种目标进行综合分析的调度问题，这里不再一一列举。综合归纳以上求解多目标调度问题的策略无外乎两种：

（1)将其通过加权或者约束转化为单目标调度问题。在类似的工作中，如何确定权值以及注重多目标之间评价的公平性一直是研究的热点。而在含有约束条件的调度问题中，一般引入惩罚因子来对超出约束范围内的策略给出相应的惩罚。

（2）定义类似数学空间上“锥”的结构来进行优化。在第六章介绍的 Pareto 最优性就是此类“锥”的一种，目标之间的相对不可比性只要能在转化后锥的结构中具有可比性即可。

2.3　弹性云平台多指标综合评价模型

弹性云计算平台的综合评价模型存在的意义在于比较各种公有云服务提供商能提供服务的能力。目前指标体系还没有很好的确立，其难点主要有二：第一是多指标的复杂性，指标不仅需要关注传统的吞吐率和响应时间，还要涉及许多诸如云平台使用成本、可扩展性请求响应时间、隔离性和公平性等。第二，对于以上每一类指标体系，需要很长的时间来进行评价和数据

的获取，以此导致评价本身困难重重。下面从分别解决以上两问题的相关工作出发进行横向分析，并分别比较我们所提出的方法与之不同之处。

2.3.1　综合评价指标体系构建

不同于传统高性能计算仅仅关注每秒浮点运算次数（FLOPS），弹性云计算平台真正需要构建的指标体系多种多样，其主要目的是比较不同公有云平台提供商提供服务能力的差别。在文献[93]中，作者提出比较公共云平台基础设施的方法，从四个一级指标（弹性计算集群、持久性存储、云内网络和广域网络），十个二级指标（基准测试程序运行时间、基准测试程序使用成本、可扩展性、任务延迟、任务吞吐率、每个任务的执行成本、存储一致性所需时间、云间网络延迟、云间网络带宽、广域网络延迟）进行了横向分析，并指出基于任务踪迹（Traced-based）的性能预测方法。

在文献[94]中，同上作者采用文献[93]中定义的所有性能指标，对 Amazon EC2[16]，Microsoft 的 Azure[95]、Google 的 AppEngine[18]和 Rackspace[96]的 CloudServers 四种典型云平台下，采用自定义改进后的 SPECjvm2008[97]基准测试程序进行了性能分析测试和比较。

2.3.2　仿真优化最优计算量分配

最优计算量分配（Optimal Computing Budget Allocation，简称 OCBA）的核心思想是：在基于策略优化问题中，将有限的计算量更多的分配给前期仿真结果中表现优良的策略。在基于仿真优化[98,99]序优化的概念提出三年以后，OCBA[26]就第一次被公开发表并引起关注，并随后行成了许多拓展和分支。利用 OCBA 的思想，以提升序优化方法性能为目的来获得最佳仿真策略的工作在文献[100]中提及，其将 OCBA 建模成受限优化问题，目标函数是在有限次观测实验后，所挑选出来的观测最优策略事实上是真实最优策略，本事件发生的概率最大化，而受约束的条件就是问题的求解计算成本总量。文章给出了在不同的仿真阶段，对于前期不同观测性能的策略在后一阶段应该增加观测次数的数值。为了延续序优化目标软化的思想，作者将优化思路进行松弛[101]，即给出了求解目标为最优策略集合（例如真实排名前 5 的策略构成的集合）中的一部分的基本情况，而非前篇文献仅仅关注最优策略本身。如此这般，选择出来的一系列策略能够更好的切合使用本方法用户的需要，因为最优策略一项并非适用于所有用户。作者通过重新定义问题的方案，

同样给出了在任意仿真阶段完成时，后一阶段需要增加的仿真量，从而以最大概率选择出真实最优策略集合中的策略。

前述工作考虑的是策略彼此独立的情况，而文献[102]中又相应给出了采样彼此关联时 OCBA 问题的求解思路。首先从两个策略上得出结论，只要方差差异不是特别显著，一般需要采样相同点，并将以上结果在多策略中进行推广。在文献[103]中，对离散事件仿真实验进行了最优计算量分配，从而动态决定每次试验中策略仿真的长度以更加高效的实施优化策略。一种基于巢分区、序优化和仿真控制方法进行结合在文献[104]中提到，该方法组合不仅能保留巢分区方法的全局性，同时也有序优化方法快速收敛的优势。在文献[105]中，交叉熵方法在 OCBA 中被使用以提升优化的性能。本方法以减小交叉熵加权函数的期望误差均方差为衡量指标来提升采样的分布的更新过程。实验结果证实了交叉熵方法在 OCBA 能够大大提升仿真的效率。

2.3.3　多目标最优计算量分配

文献[106-108]提出了在多目标评价体系下的分级和选择的方法，采用了 Pareto 最优性原理，试图寻找出在 Pareto Front 的策略，而非仅仅只是传统意义上的“最优策略”。其核心思想还是在初始采样中对于那种经常表现具有优超（Non-dominance）其他策略的策略分配更多的计算量，而相对被优超的策略则相应减少计算量。其所对应于序优化中的“序”的比较则转化为第一类错误（真实非 Pareto Front 中的策略在观测性能中被选入 Pareto Front 中）和第二类错误（真实 Pareto Front 中的策略在观测性能中未被选入 Pareto Front 中）的概率。基于此，文中详细的给出了所有观测性能中表现各异的策略在后期阶段分配计算量的量化计算公式。

在文献[107]的实验结果中证实了其搜索性能比理论最优方案（Theoretical Optimal Allocation）、理论均匀分配（Theoretical Uniform Allocation）和均匀计算量分配（Uniform Computing Budget Allocation）方法要强。对于多目标仿真优化中具有相近性能的策略问题，在文献[109]中给予了详细的分析，并引入性能相近策略 Indifference Zone（IZ）的概念，文中介绍了如何确定非优超的概率，如何定义此情形下的 Pareto Front，如何在仿真中将有限计算量进行分配，其成功的将计算量分开，而不至于在 IZ 范围集中区域浪费计算量。

对于多目标 OCBA 来说，和我们的多评价性能指标构建的初衷完全一致，但是实现起来方法完全不同。多目标 OCBA 视作目标本身没有任何差

异性，单次仿真均对所有目标进行同样次数仿真，而表现差异性的地方在不同的策略上。而本文对于云计算平台多指标构建中与此相反的是，对于仿真策略（不同的云计算平台）本身也许并不考虑差异性（这一点在拓展的工作中可以表现差异性，能够更好的提升仿真性能，节省仿真时间），而是在多目标本身体现出衡量标准的计算量分配上。

2.3.4　最优计算量分配的应用

在单目标性能优化领域，OCBA 思想曾发挥过重要作用。文献[110,111]将序优化和 OCBA 方法用于异构设备组合下半导体晶圆制造的复杂调度问题方面，大大降低了仿真时间。文献[112]将 OCBA 方法应用于美国空中交通管理中，其被建模成一个巨大、复杂的随机网络，利用 OCBA 计算时间能减小 54%到 88%。文献[113]中将 OCBA 用于蒙特卡洛仿真的电子电路产品设计问题中，实验结果显示同样能节省大量的仿真时间成本，提升设计效率。在高性能和桌面计算中，国际数学和统计库（International Mathematics and Statistical Library，简称 IMSL）被广为使用。文献[114]针对 IMSL 中的统计聚类数值计算问题，证实了 OCBA 方法相对于传统启发式方法能够能以更高的置信概率，更短的计算时间获得良好的处理性能。

在多目标优化领域，文献[115]考虑了一种含有多种请求类别的差分服务库存问题（Differentiated Service Inventory），作者采用连续审查策略和动态阈值曲线来对不同类别的请求提供差分化的服务。本方法采用巢分区方法搜索最有希望的策略空间，然后利用多目标 OCBA 方法寻找出非优超策略集合，实验结果证实了方法的优越性。在文献[116]中，空中飞行器闲置部件的分配问题被考虑。其采用多目标进化算法寻找性能相对较好的策略空间（此处与文献[115]中巢分区用处类似），同样紧接着用多目标 OCBA 方法寻找非优超策略集合。

纵观仿真优化方法，其优化维度可以有二：首先，在策略维度进行优化。前述提及的与 OCBA 相关的所有工作均是如此，即在给定性能指标的条件下，将初期性能优秀的点以更多的计算量来在后期试验中进行观察，以达到优化仿真目的。而我们在第七章中所提到了进化仿真优化方法，则是在第二个优化维度：指标维度进行的最优计算量分配。即初始阶段不一定需要将所有指标均仿真获取，只需挑出部分仿真指标获取其结果，紧接着实验阶段同样经过本过程但是有学习机制来分析哪些指标对总体性能贡献最大，而在后

续仿真阶段则更多的将前面阶段贡献更大的性能指标分配更多的计算量进行仿真。本过程实质上是在粗糙模型的基础上进行的仿真优化方式。

进化仿真优化方法的提出源于评价弹性云平台多指标综合模型时存在的实际困难，即对于所有指标均进行完整评价的耗时问题。在本类问题中应用进化仿真优化方法，可以将有限的计算量分配到对最终性能最具主导作用的评价指标集合上，从而在指标维度进行仿真优化。

2.4　相关工作小结

本章分别从弹性云计算平台虚拟资源自适应供给、各种物理虚拟资源调度管理方法、多任务负载敏捷资源供给和多指标综合评价模型入手，对已有相关工作进行总结和调研。参考文献来源大多是国际顶级会议、期刊、杂志、研究机构和工业公司的技术总结、文献综述和调研报告等等。经典方法一般选取权威期刊早期且具有最高引用率的论文，技术拓展和思路方法主要选取近几年提出来和发表的专业论文。

首先，在弹性云计算平台虚拟资源自适应供给方面，现有工作主要关注点在传统分布式计算架构的建模上，从单层队列模型到多层队列模型进而到网络队列模型，从采用基本数学模型到采用机器学习方式方法，均体现了分布式系统架构复杂化的趋势。在第三章中，本论文提及的方法是构建在弹性计算云平台基础上，其主要特点是所构建的数学模型侧重于虚拟资源按需划分和自适应提供，这一点和以往相关工作有显著不同。

其次，在弹性云计算平台多任务敏捷管理调度问题上，传统方法要么过多的关注于主要单一性能指标，要么将问题简化为静态调度问题。事实上，多目标调度问题的“鱼和熊掌不可得兼”一直为分布式计算中的困难所在，已经在本章中做了相应总结。第四五章，我们利用仿真优化中的向量序优化思想，将其拓展到多阶段领域，实现了多任务问题低开销的敏捷调度。

最后，在弹性云计算平台多指标综合评价模型中，本方面相关工作甚少，其原因首先在于所需构建数据中心性能指标的多样性实在不胜枚举，其次在于评价海量指标耗时费力。针对后者，OCBA 方法虽然成功的应用在仿真优化领域，但是其将有限计算量过多的分配到最重要的策略方案中的实施思路，正是我们可以借鉴之所在。本节对 OCBA 从原理、单目标、多目标和应用方面进行了全面的总结。第六章我们将尝试着借鉴 OCBA 思想，将其体现在粗糙模型的进化思路上。

第3章　弹性云平台资源自适应供给

3.1　本章引论

弹性云平台最显著的特点是虚拟资源的按需、动态和自适应供给，其目的是优化的配置物理资源使其能够适应不断增大的客户规模和请求数量，同时也能高效的保证请求的执行时间和用户的满意度。由于虚拟资源可以跨域分布自组织灵活的部署在物理集群上，从而弹性云平台在本质上为服务自适应供给提供了天然的优势。如何对其进行合理的建模，使得模型本身能够捕获系统的各项特征是本章需要解决的首要问题。

针对这个需要，本章提供了一套虚拟资源自适应供给的方法，提出和分析了弹性云平台的模型特点，给出了确定用户请求响应时间的求解思路，同时在基准测试程序上进行了验证。验证结果说明本模型能够比较真实的捕获系统运行期特点，并能为虚拟资源自适应供给提供良好的依据。同时我们还将本方法和传统基于虚拟资源利用率的方法进行了比较，实验结果说明本方法能够比较快速的捕获到由于请求动态变化导致虚拟机管理决策需要，从而敏捷进行的资源调度。

本章的组织结构如下：3.2 节介绍弹性云平台的总体架构和多层排队模型概览；3.3 节对于上述模型的详细建立思路、求解过程的执行步骤以及模型中主要未知参数的估计方法；3.4 节通过两部分实验结果对模型本身的有效性和适用性进行了详细分析；最后 3.5 节是本章工作的总结。

3.2　弹性云平台系统总体结构概览

本节对弹性云平台的背景做了详细的介绍，探讨物理服务器、物理集群、虚拟机、虚拟集群、资源池的构成等概念和组成方法，并对弹性云平台多层排队模型进行了总体性建模，同时提出应用的本模型中的所有符号及其释义。

3.2.1　弹性云计算平台简介

弹性云平台由若干物理集群（Physical Cluster）组合而成，每个物理集群由若干个物理服务器（Physical Server）结点组合而成。物理集群通过硬件资源池进行资源共享，以在互联物理服务器内部或者之间，将其划分成若干虚拟机（Virtual Machine），整个弹性云平台里便留存大量可按需提供已

经配置好的虚拟机资源。虚拟资源根据应用需求和各自所分配 CPU 核数量、内存、磁盘等物理配置划分成虚拟集群（Virtual Cluster）。虚拟集群能够非常灵活的创建、重组、取消、添加虚拟机资源，删减虚拟机资源等。

图 3.1 综合的图示化概括了以上各概念。其包含 3 个物理集群，每个物理集群由三台物理服务器组成，每个物理集群上划分出 12 个虚拟机，用矩形方框表示，隶属于不同物理集群的虚拟机采用不同深度的颜色填充。当任务需要执行时，根据任务需求，可以将虚拟机按照图中方式进行组织搭建虚拟集群。图中展示了四个虚拟集群，分别用外框不同形状的实线、虚线和点划线表示。

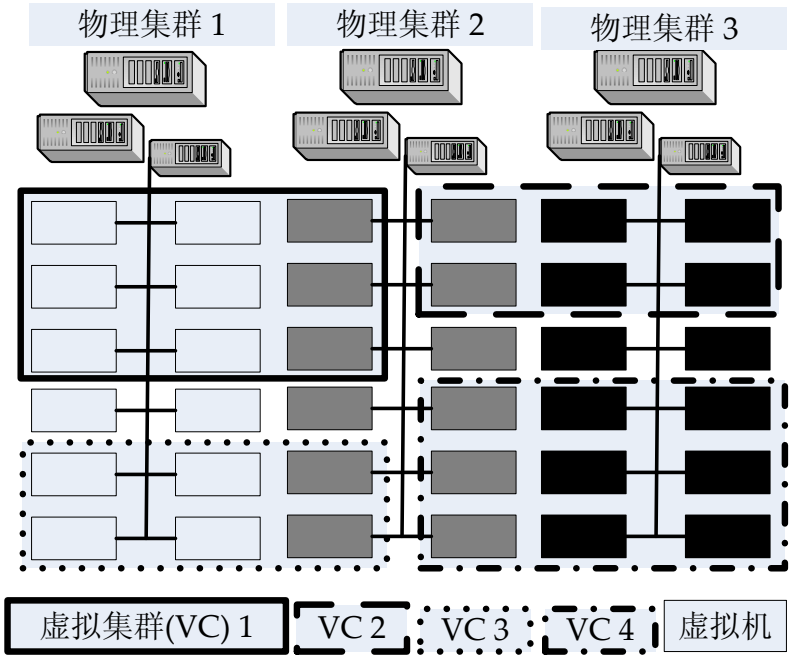


图 3.1　物理集群和虚拟集群结构示意图

本章划分虚拟集群的目的是根据不同的工作负载（Workload）的大小，提供具有自由弹性、可以灵活伸缩的虚拟资源供给方案。本方案为了完全实现资源的按需分配，自适应供应。一方面能够提升服务端数据中心资源的利用率，不至于产生大量的服务器的低利用率空转；另一方面能够向用户收取最合适的费用来满足和用户达成的服务层次协议（Service Level Agreement，简称 SLA）。

通常云计算服务商将预设好不同种类的虚拟机实例以供用户调度使用，此不同种类的虚拟机部署大小数量各不相同物理资源。例如在 Amazon EC2 的虚拟机实例中，包含诸如标准实例、微小实例、高内存实例、高 CPU 实

例、集群计算实例、集群 GPU 实例等满足不同用户的需求。其中每种实例亦可划分成若干子实例类别。本弹性云平台不需要设计得如此复杂，仅提供两种简单实例进行选择。一种为瘦虚拟机实例，其标准资源部署为每个实例 1 个虚拟 CPU 核和 1 GB 的内存；另一种为胖实例类别，其标准资源部署为每个实例 2 个虚拟 CPU 核和 2 GB 的内存。不同实例提供不同的计算能力和处理能力，所以要分别建模和分析。

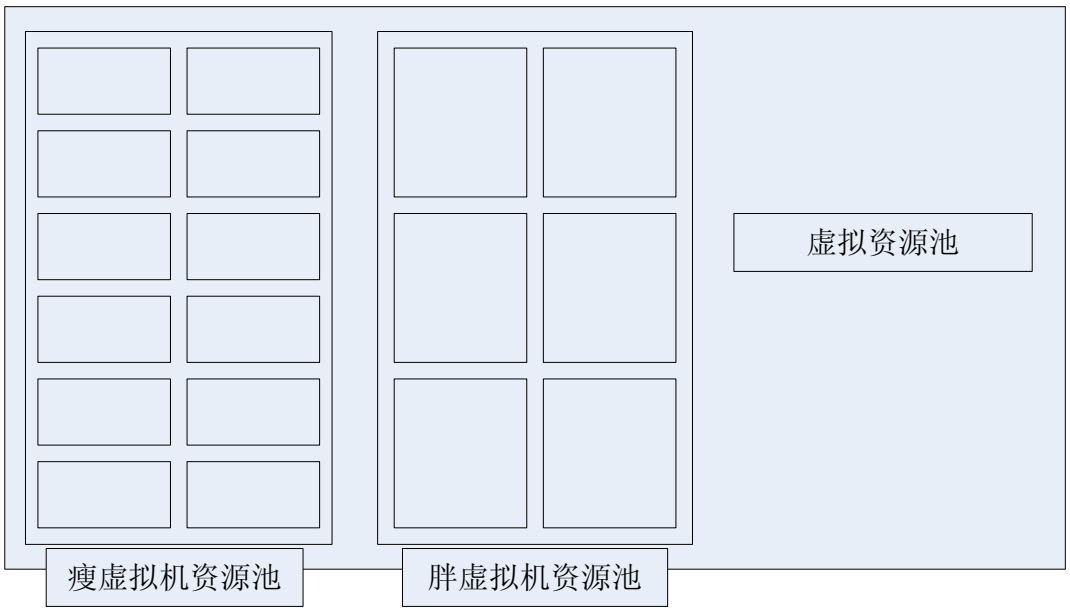


图3.2　虚拟资源池以及胖瘦虚拟机资源图示

3.2.2　多层次排队网络模型

为了便于建立数学模型，我们将弹性云计算平台构建成多层次排队论模型，如图 3.3 所示。本模型中共有 J 层服务台，每个服务台用一个队列表示。为了便于整体介绍网络模型，本图所示服务台是一个单进单出队列，事实上本服务台为一个多进单出队列，详细图示见图 3.4 介绍。

左上端的 Q_0 代表一共有 n 个会话（S_1 到 S_n）同时发送请求，每个会话可以理解成一个独立的用户。每一个到达第 j 层的请求的下一步跳跃有两种情况：它可能是源于从会话端发出还未处理的请求，则其会以 P_j 的概率跳到它的后一层即 j+1 层；它亦可能是来源于后层已经处理过的请求，则其会以$(1-P_j)$ 的概率跳到它的前一层即 j–1 层。到达第 J 层请求会以概率为 1 返回到第 J–1 层。因此 $P_J = 0$。

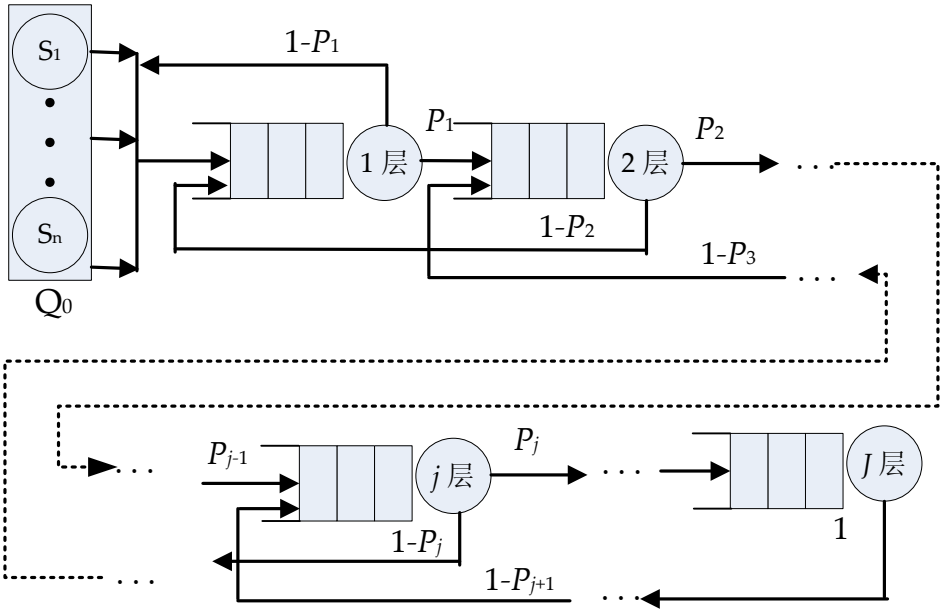


图3.3　虚拟集群多层次队列模型图示

调度的目标是通过合理部署每一层虚拟机的数量。换言之，就是并列构造胖瘦虚拟机实例的个数，从而满足用户的各项要求。显然由于计算资源竞争、I/O 资源竞争、网络时延等各方面因素导致在请求数量特别大的时候，或者请求激增的时候会有不断的丢包存在。因为我们设定 95%的用户请求能够在规定的时间正确的响应即可，这就是用户和弹性云计算平台服务提供商签订的 SLA 的协议。

在本模型中，网络延时并未作为一个独立的因素考虑进来，其原因在于本实验所构建的网络环境本身实验比较小，网络通信延时不是影响系统调度性能的主要因素；另外由于通过网络传递各层之间的请求时间相比于请求被缓存到各层服务器的等待时间可以忽略，而同层之间由于构建了虚拟集群，一般来讲 Hypervisor 会将虚拟机构建到最适宜且延迟最小的物理实例中，从而使得同层之间的延迟也不会明显。但是本问题在多任务负载强度变化非常剧烈时，由于缓存资源使用迅速增多，排队等候请求也会急剧增多，因此网络延时的问题会凸显出来。本章中我们采用多任务负载变化相对并不太大的实际情况来实现，后述章节我们会考虑负载变化剧烈的问题。

由于参数较多，为了便于读者更好的理解各个参数的意义，我们在表 3.1 中进行了汇总和概括。对于各个符号详细的意思和释义，会在下一章虚拟资源调度数学模型和求解中详细论述。

表3.1　弹性云平台多层次排队模型的符号和定义

符号	意义
i or i-1	当前或者之前的一个调度阶段
j or J	当前或者总队列长度
$N_j^F(i)$，$N_j^T(i)$	第 *i* 个调度阶段第 *j* 层中胖/瘦虚拟机实例分配数量
$\mu_j^F(i)$，$\mu_j^T(i)$	第 *i* 个调度阶段第 *j* 层中胖/瘦虚拟机实例平均服务率
$\rho_j^F(i)$，$\rho_j^T(i)$	第 *i* 个调度阶段第 *j* 层中胖/瘦虚拟机实例的服务强度
AST_j^F，AST_j^T	请求在第 *j* 层中胖/瘦虚拟机实例平均停留时间
AAR_j，$\lambda_j(i)$	请求在第 *j* 层中的平均到达率
$\lambda'_j(i)$	请求在第 *j* 层中每个虚拟机实例的平均到达率
$AAR_{j-1,j}$, $AAR_{j+1,j}$	请求从第(*j*-1)层或第(*j*+1)层到达第 *i* 层的平均到达率
$ADR_{j,j-1}$, $DR_{j,j+1}$	请求从第 *j* 层到第(*j*-1)层或第(*j*+1)层平均到达率
ADR_j	请求在第 *j* 层的平均离开率
AST_j	请求在第 *j* 层的平均服务时间
C_j	第 *j* 层虚拟机所能接受最大请求数目
R(i)	第 *j* 个调度阶段请求的产生数量
P_j	请求从第 *j* 层到第 (*j*+1) 层的概率
Θ_F *or* Θ_T	胖瘦虚拟机总数量

3.3　虚拟资源调度数学模型和求解

本节首先讨论弹性云平台虚拟资源调度的多层次排队网络模型，以及响应时间的推导公式，然后我们介绍求解模型的详细步骤。最后，我们介绍模型中需要用到的参数并讨论如何获取它们的取值。

3.3.1　层次结构建模

假设在第 *i* 个调度阶段，弹性云平台的第 *j* 层使用的胖虚拟机实例和瘦虚拟机实例分别为 $N_j^F(i)$ 和 $N_j^T(i)$ 可以计算为：

$$N_j^F(i) = N_j^F(i-1) + \Delta N_j^F(i-1) \tag{3.1.a}$$

$$N_j^T(i)=N_j^T(i-1)+\Delta N_j^T(i-1) \tag{3.1.b}$$

即当前调度阶段虚拟机实例的数量由上一调度阶段调整而来，分别增加 $\Delta N_j^F(i-1)$ 和 $\Delta N_j^F(i-1)$ 个胖瘦虚拟机实例，其取值可以为负。

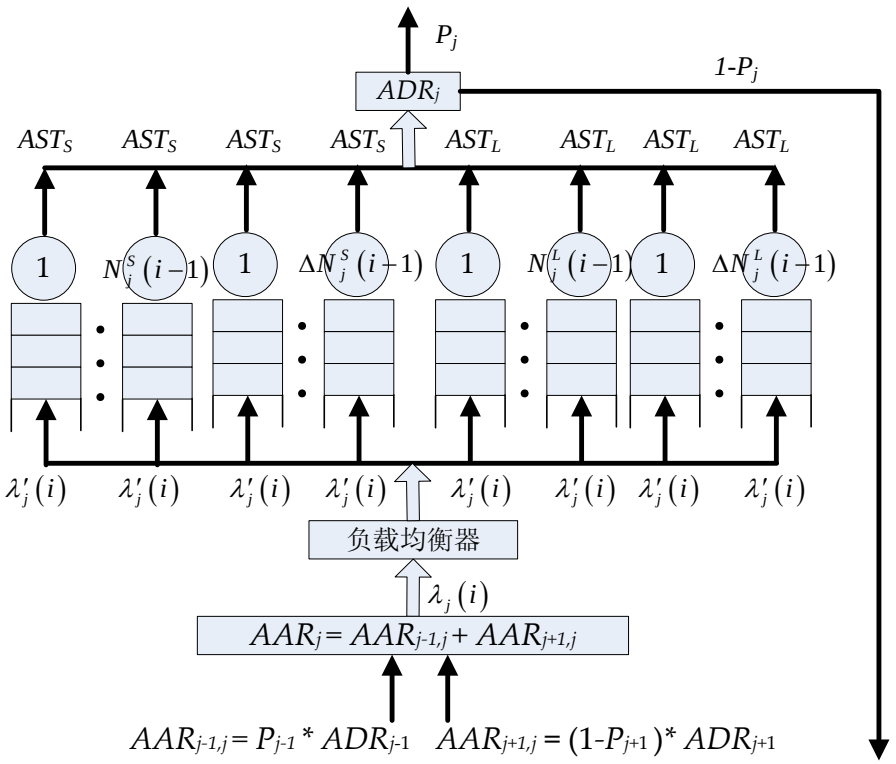


图3.4 弹性云平台第 j 层队列内部虚拟资源布局图示

在图 3.4 中，我们展示了弹性云平台第 j 层的内部结构。在当前调度阶段，第 j 层请求的期望到达率表示为 $\lambda_j(i)$，于是经过负载均衡器，每个队列的请求到达率表示为：

$$\lambda_j'(i)=\lambda_j(i)/\left[N_j^T(i)+N_j^F(i)\right] \tag{3.2}$$

和$\lambda_j(i)$一样，AAR_j 也代表第 j 层请求的期望到达率，其中包含两个部分：第一部分是来自于前一层，也就是第 $(j\text{-}1)$ 层的请求，由 $AAR_{j-1,j}$ 来表示；第二部分是来自于后一层，也就是第 $(j+1)$ 层的请求，由 $AAR_{j+1,j}$ 来表示。以上式子表示为：

$$\lambda_j(i) \text{ 或者 } AAR_j = AAR_{j-1,j} + AAR_{j+1,j} \ (j\in[1, J\text{-}1]) \tag{3.3}$$

由所有用户发出的请求 Q_0 直接到达第一层，用 $AAR_{0,1}$ 来表示，而在最

后第 J 层，由于没有请求从后层返回，则可以表示为：

$$\lambda_J(i) \text{ 或者 } AAR_J = ADR_{J-1,J} \tag{3.4}$$

每个虚拟机实例可以被视作一个 M/M/1/C/∞/FIFO 的排队系统，C_j 代表了每个虚拟机实例所能同时接受请求数量的上限，因此我们根据排队论方法，可以得出每个请求在第 j 层胖或者瘦虚拟机实例中的平均等待时间为：

$$AST_j^F = \frac{1}{\mu_j^F(i) - \lambda_j'(i)} - \frac{C_j \rho_F^{C_j+1}}{\lambda_j'(i)(1-\rho_F^{C_j})}, \rho_j^F(i) = \frac{\lambda_j'(i)}{\mu_j^F(i)} \tag{3.5.a}$$

$$AST_j^T = \frac{1}{\mu_j^T(i) - \lambda_j'(i)} - \frac{C_j \rho_T^{C_j+1}}{\lambda_j'(i)(1-\rho_T^{C_j})}, \rho_j^T(i) = \frac{\lambda_j'(i)}{\mu_j^T(i)} \tag{3.5.b}$$

基于以上分析，我们得出第 j 层请求的离开率可以表示为：

$$ADR_j = N_j^F(i) / AST_j^F + N_j^T(i) / AST_j^T \tag{3.6}$$

接着可以获得请求到达第 j 层以后的平均等待时间，为平均离开率的倒数，表示为：

$$AST_j = 1/ADR_j \tag{3.7}$$

然后可以算出到达第 j 层的请求到达第$(j+1)$层的到达率，表示为：

$$AAR_{j,j+1} = ADR_j * P_j \tag{3.8}$$

同样的道理，我们可以算出到达第 j 层的请求到达第$(j-1)$层的到达率，表示为：

$$AAR_{j,j-1} = ADR_j * (1 - P_j) \tag{3.9}$$

3.3.2　执行算法分析

本节我们将系统介绍求解以上数学模型的方法，即在给定第 i 个调度阶段，第 j 层胖瘦虚拟机的数量，求出在当前请求到达率情况下系统的平均响应时间。整个求解过程分两个阶段五个步骤，先从后向前阶段每步带有一个未知数，然后从前向后阶段求解每一层的未知数继而获得整体平均响应时间。

（1）分析第 J 层

根据 (3.5) 和 (3.6) 式，第 J 层的离开率可以表达为 ADR_J 可以表达为第 J 层到达率 $\lambda_J(i)$ 的函数，而 $\lambda_J(i)$ 此时未知。同时，我们亦可获得第 J 层上请求的平均停留时间如(3.8) 式为：$AST_J = 1/ADR_J$ 。于是在 AST_J 的表达式中，存在一个未知变量$\lambda_J(i)$。同时为了便于整篇分析，我们将式(3.5)简

化为：

$$ADR_J = f_1(\lambda_J(i)) \tag{3.10}$$

（2）分析第 $(J-1)$ 层

由于第 $(J-1)$ 层的到达率可以表示为：

$$\lambda_{J-1}(i) = AAR_{J-2,J-1} + AAR_{J,J-1} = AAR_{J-2,J-1} + ADR_J \tag{3.11}$$

则此时含有两个未知变量：$AAR_{J-2,J-1}$ 和 ADR_J，同时我们也可根据(3.5)和 (3.6) 式列出 ADR_{J-1} 的表达式，其同样包含以上两个未知变量，假设此表达式简化为：

$$ADR_{J-1} = g_2\ (AAR_{J-2,J-1}, ADR_J) = g_2\ (AAR_{J-2,J-1}, f_1(\lambda_J(i))) \tag{3.12}$$

再根据表达式 (3.8)，我们有：

$$\lambda_J(i) = ADR_{J-1} * P_{J-1} \tag{3.13}$$

因为 P_{J-1} 可以通过预先观察估计出来，而其观察方法会在 3.3.3 中作出详细的介绍。于是我们将式 (3.12) 和 (3.13) 式联立求解，可以消元$\lambda_J(i)$，得到 ADR_{J-1} 和 $AAR_{J-2,J-1}$ 的对应表达式函数。我们记为：

$$ADR_{J-1} = f_2(AAR_{J-2,J-1}) \tag{3.14}$$

同样我们也能获得 $AST_{J-1} = 1 / f_2(AAR_{J-2,J-1})$。由于本式子涉及消元比较复杂，我们对 $\rho_j^F(i)$ 和 $\rho_j^T(i)$ 的表达进行了近似，将分母项以常数估计。

以此类推，我们不断往前推进，对于任意一层 j，我们进入第三步：

（3）分析第 j 层，j 的取值从 $J-2$ 到 1

对于任意一层 j，我们依然执行（2）中的步骤，可以得到表达式：

$$ADR_j = f_{J-(j-1)}(AAR_{j-1,j}) \tag{3.15}$$

同时也可以得到 $AST_j = 1 / f_{J-(j-1)}(AAR_{j-1,j})$。

而当我们的递推求解到达第一层的时候，对于 $AAR_{0,1}$ 则为当前请求从客户端 Q_0 的平均产生率，为假设已知量，这样通过 $j = 1$ 时求解式(3.15)，获得 ADR_1 和 AST_1 的值。而假设已知量 $AAR_{0,1}$ 的获取则是通过一般预测模型根据过往请求到达率所推断的，所以是一种近似。

（4）再次分析第 j 层，j 的取值从 2 到 J

在前述步骤中，我们已经计算出 ADR_1 的取值，然后根据(3.8)式，$AAR_{1,2}$ 的取值亦可以计算得出，其中 $j = 1$。因为根据 (3.15) 式，同样能计算出 ADR_2 和 AST_2 的取值。以此类推，我们逐步往后求解可以将每一层 ADR_j 和 AST_j 的取值获得。

（5）计算平均响应时间

假设在调度的第 i 个阶段总共有 $R(i)$个请求在弹性云平台中，由于不同请求所经历的层数不同，我们用符号 $Num(R_j(i))$ 代表从第 1 层一直到第 j 层，然后返回直到第 1 层直到返回给用户的请求数量总数，也就是说这部分仅仅经过 j 层。以上类型的所有请求在弹性云平台中的平均遍历时间用 $TraverseTime(R_j(i))$表示，则所有请求的平均响应时间由(3.16)式给出。

表 3.2 描述了 j，$Num(R_j(i))$和 $TraverseTime(R_j(i))$的关系，其均用于计算第平均响应时间。

表3.2 j，$Num(R_j(i))$和$TraverseTime(R_j(i))$的关系

j	$Num(R_j(i))$	$TraverseTime(R_j(i))$
1	$R(i)(1-P_1)$	$R(i)(1-P_1)*AST_1$
2	$R(i)P_1(1-P_2)$	$R(i)P_1(1-P_2)*(AST_2+2*AST_1)$
---	---	---
j	$R(i)P_1P_2\text{---}\ P_{j-1}(1-P_j)$	$R(i)P_1P_2\text{---}P_{j-1}(1-P_j)*(AST_j+2*AST_{j-1}+\ldots+2*AST_1)$
---	---	---
$J-1$	$R(i)P_1P_2\text{---}P_{J-2}(1-P_{J-1})$	$R(i)P_1P_2\text{---}P_{J-2}(1-P_{J-1})*(AST_{J-1}+2*AST_{J-2}+\ldots+2*AST_1)$
J	$R(i)P_1P_2\text{---}P_{J-2}\ P_{J-1}$	$R(i)P_1P_2\text{---}\ P_{J-1}(1-P_J)*(AST_J+2*AST_{J-1}+\ldots+2*AST_1)$
1	$R(i)(1-P_1)$	$R(i)(1-P_1)*AST_1$

则平均响应时间计算如下：

$$\sum_{j=1}^{J} TraverseTime\left(R_j(i)\right)\Big/R(i)$$

$$=\sum_{j=1}^{J}\left(\prod_{k=0}^{j-1}P_k\right)*\left(1-P_j\right)*\left(2*\sum_{l=1}^{j}AST_l-AST_j\right) \tag{3.16}$$

由于客户端请求会无条件跳到第一层，则 $P_0 = 1$。

假设使用胖瘦虚拟机实例的成本分别为 $Cost_F$ and $Cost_T$，则整个弹性云平台的使用成本可以如下表示：

$$Cost(i)=Cost_F*\sum_{j=1}^{J}N_j^F(i)+Cost_T*\sum_{j=1}^{J}N_j^T(i) \tag{3.17}$$

因此，本章弹性云平台虚拟资源的自适应供给问题也就转化成一个受限优化问题，定义如下：

$$\underset{N_j^F(i),N_j^T(i)}{Min}\left(Cost(i)\right)$$

$$\begin{cases}\sum_{j=1}^{J}\left(\prod_{k=0}^{j-1}P_k\right)*\left(1-P_j\right)*\left(\prod_{l=1}^{j}\left(2*AST_l-AST_j\right)\right)<SLA\\ \sum_{j=1}^{J}N_j^F(i)\le\Theta_F\\ \sum_{j=1}^{J}N_j^T(i)\le\Theta_T\end{cases}$$

其中Θ_F和Θ_T分别代表可以使用的胖瘦虚拟机的数量。

3.3.3　主要参数估计

根据前面两章所描述的数学模型，为了从公式上直接求出响应时间的数值，我们有若干个物理参数需要估计其取值，下面详细探讨实验前对其估值技术方案：

（1）请求在连续两层之间跳跃的概率 P_j ($j\in[1, J\text{-}1]$)

任何请求在接受完成当前 j 层服务器的执行后，要么以概率 P_j 跳到后面第 j+1 层，要么以概率 1–P_j 返回到前面 j–1 层。而跳转概率的设置完全由用户应用请求的种类和层次服务器之间的设计方法来定。在测试系统中，我们可以通过在线观测一段时间内到达从第 j 层跳往第 j+1 层请求的数量和到达第 j 层请求的总数量继而获得其取值，而初始情况下则对于上述所有 P_j 均设置 0.5 的跳转概率。

以上方法是最简便可行的，当然亦有其他方案估计其取值。比如统计一段时间内所有用户请求的种类，然后将其根据访问云平台的层数数量进行分类，比如仅仅访问 j 层的请求为 $Num(R_j)$，此处($j\in[1, J]$)，则跳转概率可以表示为：

$$P_j=\frac{\sum_{k=1}^{J}Num(R_k)-\sum_{l=1}^{j}Num(R_l)}{R}\tag{3.18}$$

其中 R 代表请求总数。

对于本章后述实验部分，由于采用的基准测试程序的客户端可以自动控制和设置的 P_j 值以产生不同种类的请求，所以给实验工作提供了很大的便利。

（2）胖瘦虚拟机的请求服务能力 μ_F 和 μ_T

虚拟机的服务能力随着线程数多少而呈现出较大的差异，在少量线程的虚拟机资源池中，由于资源竞争 (Contention) 较少，使得虚拟机服务能力

相对较强。而随着请求数量的不断攀升，资源竞争现象变得严重，使得服务能力相对会有所降低。而其实际服务能力取值则是通过离线观测，即通过对于胖瘦虚拟机构建简单离线测试平台，计算不同工作负载到达的实际情况下，记录各项请求在虚拟机中的到达时刻和离开时刻，即可获得其的逗留时间，在获取大量逗留时间的情况下对其进行加权后倒数，即可获得虚拟机的请求服务能力。对于分配 2 个虚拟 CPU 的内核，2 GB 胖虚拟机实例和分配 1 个虚拟 CPU 的内核，1 GB 瘦虚拟机实例，其测试方法一致。

（3）胖瘦虚拟机的估计使用成本 $Cost_F$ and $Cost_T$

使用成本的估计则可以参照大型云计算平台服务提供商 Amazon EC2 在供应其虚拟云资源时候的价格，见其参考网站[16]。而其对于虚拟机实例所提供的价格均是以小时为单位进行收费，而在我们后述实验中，时间粒度的选择为一分钟。可以针对我们胖瘦虚拟机实例的具体配置，选择与其配置相似的虚拟机类型的定价来确定使用成本。在本次试验中，我们设置的胖瘦虚拟机使用一分钟的成本分别为$0.0287 和$0.0127。这个数量是综合了虚拟 CPU 数量、内存和存储大小综合获得的数据，即采用和本实验的虚拟机采用各项资源相近虚拟机的定价来估算本实验中的胖瘦虚拟机的使用成本。

3.4　实验设置和主要结果分析

本实验设计需要两部分，第一部分采用仿真软件 Simulink 中的工具包 SimEvents 来模拟基于事件仿真的排队论模型，然后基于提出的问题模型用 Matlab 来计算和预测多请求服务的响应时间。而层间转移概率 P_j 则是直接由基于仿真实验结果，我们运行真实的 RUBiS 系统来获取真实的响应时间，已比较预测结果和真实结果的差异性，本部分已经在第一章中介绍过。

Web 服务器采用的是 Apache 2.0.55，其功能是实现请求获取辨识、静态页面的组织和负载均衡功能；应用程序服务器使用的是 Tomcat 5.5，其主要作用是部署与交易相关的应用程序的 Servlet 并且配置与前层 Web 服务器和后层数据库服务器连接的纽带；而数据库服务器采用的是 MySQL，主要存放与应用相关的所有表格，交易历史，产品信息与客户信息等。

所有实验在 IBM 中国开发中心实施，实验环境和设置在第一章中已经详细介绍过。试验中，我们总共采用了一个胖虚拟机实例和一个瘦虚拟机实例作为 Web 服务器，同时采用了两个胖服务器和四个瘦服务器作为应用程序服务器。由于数据库服务器在集群方面的实验存在技术难题，我们采用了

一个胖虚拟机实例作为数据库服务器。所采用的虚拟机实例均为按需供给和调度，初始情况下 Web 服务器和应用程序服务器均采用一个瘦虚拟机实例。

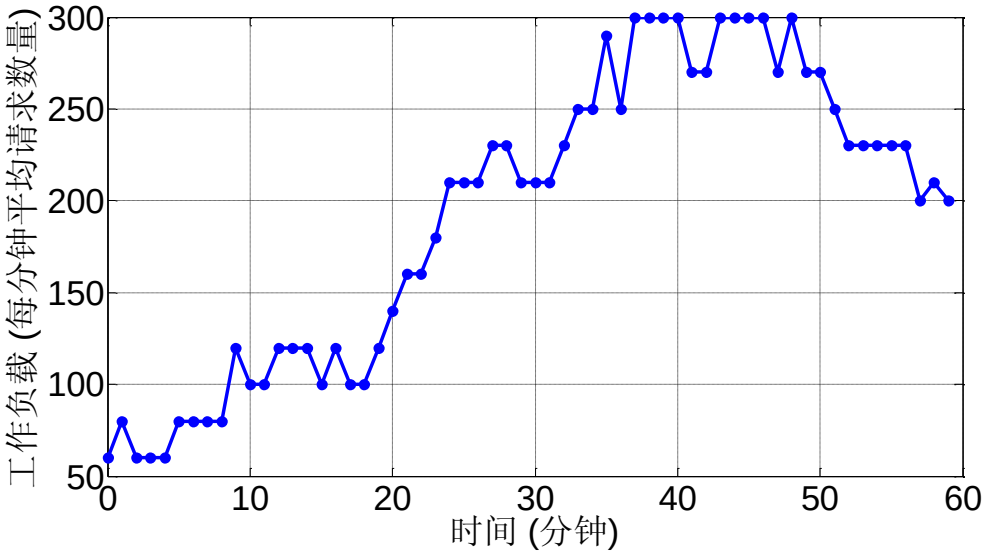


图3.5 2008年世界杯官方网站一个小时以内的负载强度

上图 3.5 展示了一个小时的实验时间里工作负载的强度变化，其数据采用的是 1998 年世界杯时官方网站上公布的网站请求访问强度的分析统计结果[117]。我们采用真实的工作负载来测试本基准测试程序系统，希望测试结果能更加符合真实情况。负载强度在开始的前 20 分钟并不太大，随后进入一个快速增长期，在第 37 分钟时到达峰值，随后小幅度波动了一段时间以后进入负载强度下降阶段。

在图 3.6 中展示了随着工作负载强度的提升，基于本文的受限优化策略所提供虚拟机数量的变化规律。很明显我们可以看出，预测模型能够很好的捕获工作负载变化的整体性规律，提供与之相适应虚拟机变化的调度方案。在工作负载不断变大时，所提供虚拟机数量也随之不断变大；变小亦然，胖瘦虚拟机也在调度过程中交替使用。

另外，我们也能获得的结论是，对于这样的三层模型，瓶颈一般都发生在应用程序服务器端，因为在第 2 层需要最多的虚拟资源才能达到较优的调度方案。这一点是可以理解的，因为应用程序服务器担任“承上启下”的作用，大部分与应用请求本身相关的任务均在本层完成。另外由于数据库服务器做集群的难度，导致大多请求缓存在应用程序服务器端，需要更多的虚拟资源来管理。

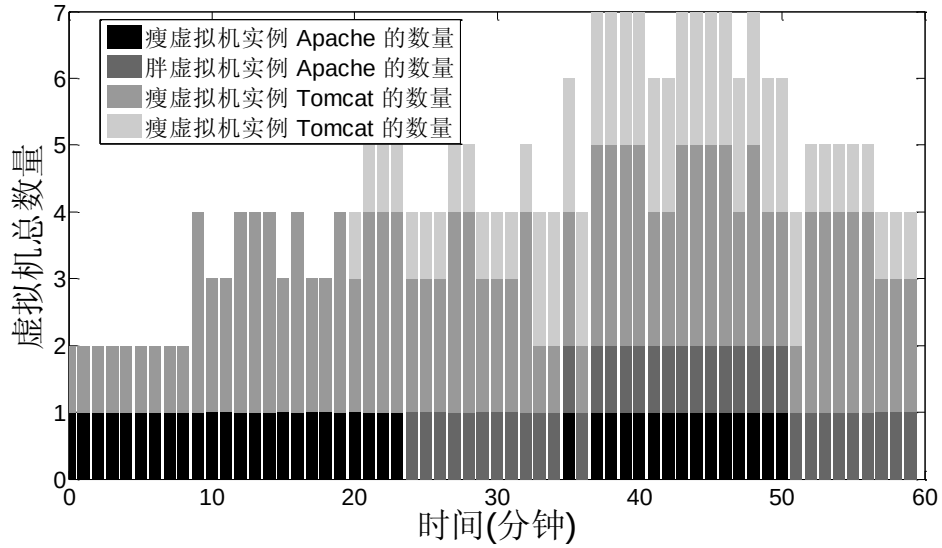


图3.6 胖瘦虚拟机实例在各层的分配方案

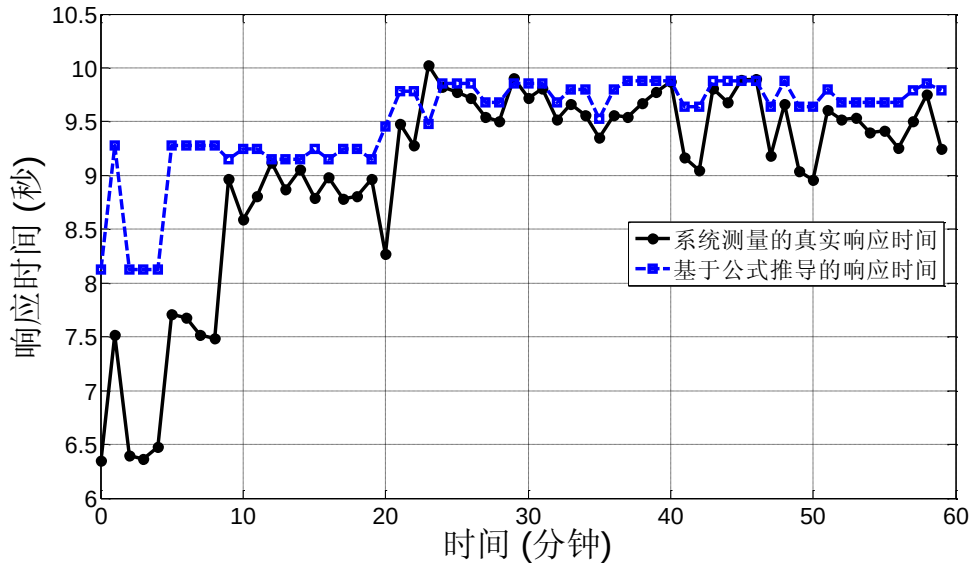


图3.7 预测响应时间和测试响应时间比较

SLA 为客户和云平台提供商制定的，为提供满意服务给客户的约定。本试验中，我们设置 SLA 为 95%以上的请求 10 秒得到响应，于是根据图 3.7 中的实验结果，我们很容易看到采用本文介绍的自适应供给方法，98%的请求都能满足约定，仅仅只有第 23 秒时超过了 10 秒的时间，而且超过的幅度并不太大。同样，上述两幅图示可以说明用本文的数学模型和求解方法能够较好的捕捉响应时间的数值，同样也可以较好的判断为了满足 SLA 请求所需要使用虚拟资源的数量。

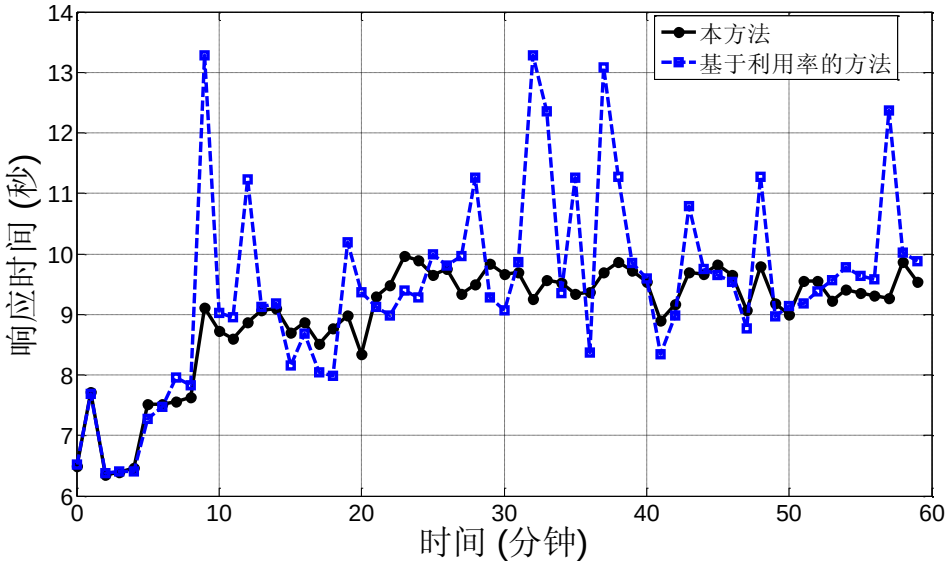


图3.8 本方法和利用率感知方法对于请求的响应时间比较

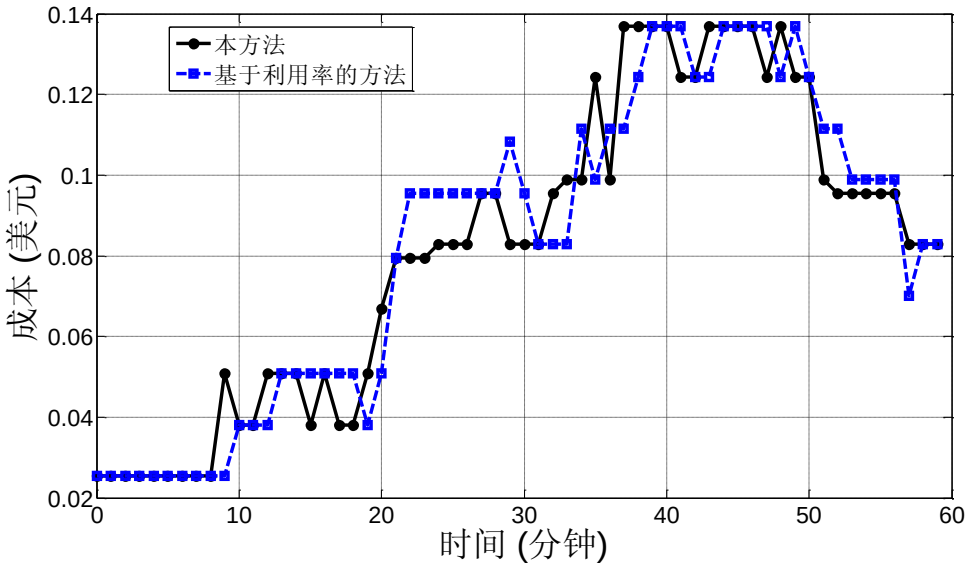


图3.9 本方法和利用率感知方法对于虚拟机的使用成本比较

然后，我们将本章使用的数学模型方法和一种广泛使用的，基于经验的方法进行比较，来验证本模型的优越性。基于经验的方法是以虚拟机 CPU 利用率作为调度策略来实行。由于负载是基本均衡的，所以每一层虚拟机的利用率基本相差不会太大。在调度过程中，如果某一层虚拟机利用率超过了 95%，在可以的情况下会增加一个胖客户虚拟机来分担请求；如果利用率在 85%和 95%之间则增加一个瘦客户机。同样的道理，如果利用率小于 45%，

则在存有该层减少一个胖虚拟机，如果在 45%到 60%之间，则减少一个瘦虚拟机。以上策略调整中，必须保证至少有一个瘦虚拟机在每一层运行以保证请求负载被执行的连续性。

很明显可以看出，以上方法是利用率感知的 (Utilization Aware)。由于实现方法简单易行，大多数传统网格计算任务调度和现有云平台请求调度方案均采用利用率感知的方法。

通过图 3.8 和 3.9 的比较，可以看出本方法的优越性所在。在图 3.8 中，本方法（黑色线）调度策略的实验结果能保证 98%的请求（只有第 24 秒响应时间略大于 10 秒）在 SLA 规定的时间得到响应，而且相应曲线比较平稳，不会出现比较大范围的起伏，这样保证了所有访问客户在获得请求响应时间时的公平性。而基于利用率的方法则相对起伏很大，而且多次违背 SLA 约定的 10 秒响应时间。

对于基于利用率调度方法违背 SLA 规定是显然的。由于在两种虚拟机调度策略基于利用率之间变化，需要其数值在一段时间之内进行调整，而在这一段时间内用户请求到达率可能会发生显著的变化从而导致调整期间并无法密切跟踪多任务负载的变化规律，而本文的数学模型具有实时性，即针对任何调度策略快速计算出最佳分配方案，从而实现虚拟资源的敏捷自适应管理。

在图 3.9 中，比较了两种方法对于调度成本的影响。依然可以看到，基于利用率调度的方法在反映到成本维度仍然有很大的波动性。比如，在基于利用率方法中具有较低响应的第 22 到 28 秒，其对于虚拟资源的成本使用过高。而本文方法在此段时间内也能满足 SLA 约定，使用成本相对低廉不少。

3.5　本章小结

本章是论文正文的开篇部分，其工作定位在弹性云平台的虚拟集群的逻辑规划，即在虚拟资源规划和调度层面进行了深入的分析和探讨。其通过排队论构建数学模型，并且给出了模型的求解思路从而获得任意请求负载到达时平均响应时间的大小，最终获得虚拟集群合理的分配方案，从而辅助服务提供商做合适的资源调度方案。

在基于 SimEvents 仿真工具和基于真实拍卖定价系统 RUBiS 基准测试程序两部分的实验结果中，证实了本章所述数学模型能比较好的获得弹性云平台的响应时间。同时将本方法和传统基于利用率调度的方法进行比较，证实

了本方法能够比较敏捷的供给虚拟资源，从而在保持较低成本的实际优势下，满足更多请求的响应时间要求。

但是本部分工作内容并未考虑到调度过程中虚拟机对于任务执行过程的动态性，即存在诸如虚拟机失效性，工作负载随机到达性等运行期状态（Runtime Condition）导致调度过程中存在的不确定性。引入以上不确定性后调度问题就会变得异常负责，后面几章分别从这里作为出发点，寻求问题的进一步求解思路。

第4章　弹性云平台资源低开销仿真优化供给

4.1　本章引论

第三章讲述的弹性云平台资源的自适应供给问题能够通过构建带约束的整数规划数学模型来满足虚拟资源供应要求，但是其主要缺点在于该方法不能将系统运行期的动态特性（诸如资源的随机可用性、工作负载的随机到达性）等纳入调度问题的考虑中，从而导致调度方法无法精准的捕获系统局部性、动态性和随机性。针对本问题所在，从本章开始考虑在引入动态性问题时的低开销仿真调度方法。

本章中我们将上述模型稍作修改，使其更能适应工作负载的动态变化和虚拟集群的按需和弹性分配。同时将引入动态性问题的弹性云平台用虚拟集群来执行多任务调度使得问题变得尤为困难。其困难点在于需要满足可能互相导致冲突的多任务目标。比如：需要同时最小化的多任务问题的总执行时间（Makespan）、和虚拟资源的使用量并且同时需要产生良好的服务质量。在这种应用场景下，我们利用基于仿真优化的现有成果，来产生次优或者足够好的解集以求得满意解。

低开销（Low-Overhead Scheduling，简称 LOS）调度算法是专门为弹性云平台下虚拟资源的优化调度设计的。我们将基于仿真优化的序优化方法进行延伸，研究如何敏捷的将部署在数据中心的虚拟资源按需隔离成虚拟集群，以动态和弹性方式提供多多任务负载的动态运算能力。

实验采用了第一章介绍的激光干涉引力波天文台项目的一组多任务计算基准测试程序，结果显示了 LOS 方法能够敏捷的产生一系列满意解集。我们在 16、32、64 和 128 个实验结点的虚拟集群中验证本方法的可扩展性，结论是本方法的调度开销为传统基于蒙特卡洛仿真的百分之一，同时能够产生足够好的决策方案。

本章的组织结构如下：4.2 节介绍了引入运行期随机因素弹性云平台多任务调度模型；4.3 节对此类随机调度问题的四类困难点进行了分析；4.4 节介绍基于向量序优化的低开销次优调度算法的基本概念和实现步骤；第 4.5 节对本章测试平台和实验环境进行介绍，并对实验结果进行了全面的分析；第 4.6 节是本章工作的总结。

4.2　低开销仿真优化调度问题背景介绍

在第三章我们介绍了如何在弹性云计算平台下划分虚拟集群使得按需弹性的提供虚拟资源来满足不断变化的用户请求的数量和分布。事实上，虚拟集群的划分在一定程度上受制于请求的种类、数量等因素。同时，多队列模型中的任何一个节点都可能成为请求的初始接收端。

在文献[118]中作者将虚拟集群作为组织来划分从而按需提供动态供给方案，其划分规则是基于负载应用的不同；而文献[119]中则更详尽的介绍了虚拟集群的方法。基于此，我们更新了模型的设计思路，将多队列前后到达的设计方案改为并行处理请求的新方案来适应不同类型的多任务工作负载。

在沿袭以上虚拟集群构建方案的同时，我们设计了如图 4.1 的调度系统，每个虚拟集群前端部署着一个用于接纳任务的到达的队列，队列中的任务采用先到达先服务方式（First-Come-First-Serve，简称 FCFS）。由于处理不同应用请求的虚拟机处理能力有差异，为了使得调度问题简化，我们所采用的虚拟机对于任务的处理能力相近。事实上，在我们后面设计的弹性云计算平台环境下，所使用的虚拟机类型完全一致来满足这一项设计。假设本问题一共划分了 I 个虚拟集群，虚拟集群 i（$i \in [1, I]$）拥有 m_i 个虚拟机，共有 I 个负载调度器用于根据工作负载到达情况和队列任务的数量均衡的提交任务请求，同时全局的控制和接受工作负载到达的频率。这里的工作负载在商业基准测试程序中是用户产生的请求，而在科学计算系统中则为每类应用需要处理的数据量大小。

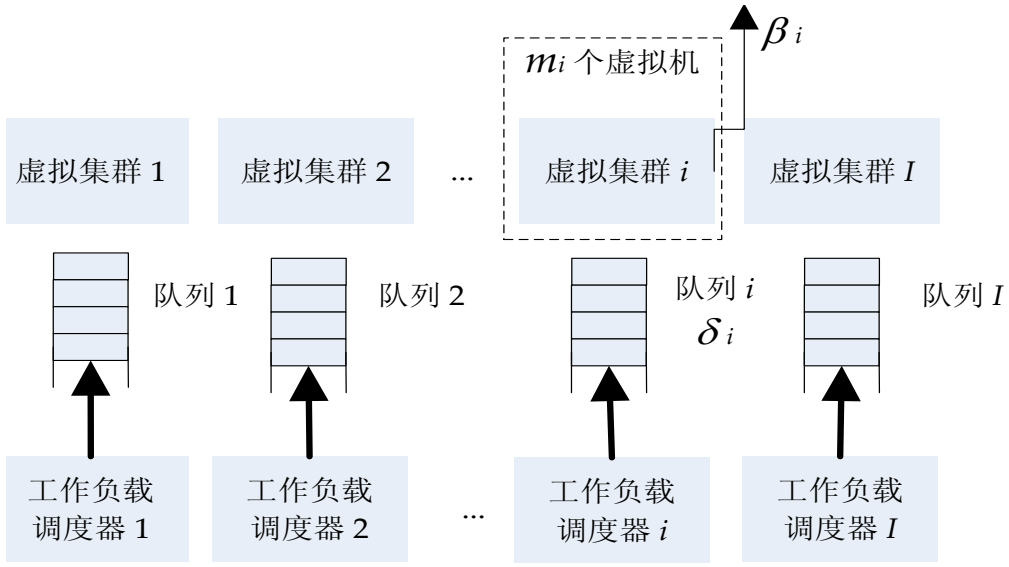


图 4.1 弹性云平台下工作负载调度问题的虚拟集群资源分配模型

表 4.1 工作负载调度问题使用的符号

符号	意义
$\theta_i^{(c)}$	虚拟集群 VC_i 中分配给任务类 c 的虚拟机数量
$\delta_i^{(c)}$	队列 i 中任务类 c 的数量
$p_i^{(c)}$	队列 i 中任务类 c 的期望执行时间
$\beta_i^{(c)}$	虚拟集群 VC_i 中任务类 c 的单位时间处理量
$t_i^{(c)}$	虚拟集群 VC_i 中任务类 c 的剩余执行时间
t^c	任务类 c 的剩余执行时间
$d_i^{(c)}$	虚拟集群 VC_i 中任务类 c 使用一个虚拟机的内存需求量
$D_i^{(c)}$	虚拟集群 VC_i 中任务类 c 使用虚拟机的内存总需求量

在本模型中，我们定义了任务类的概念。任务类为一系列具有相同类型，并且可以并行执行的任务。之所以定义任务类的目的是为了使得具有相同任务类的工作负载能够最大限度的串行化分配到某一个虚拟集群中，降低跨虚拟集群之间消息传递的开销，使得仿真策略更加有效。同时，不同任务类的任务分配到不同的虚拟集群中，使得并行化最大的同时能最大限度的提升虚拟资源的利用率。

假设我们一共有 C 个任务类，每个任务类的下标用 c 表示。表 4.1 用于总结我们在本问题中用到的所有概念。

符号 $\delta_i^{(c)}$ 代表队列 i 中任务类 c 的数量，$p_i^{(c)}$ 代表队列 i 中任务类 c 的期望执行时间，$\theta_i^{(c)}$ 代表虚拟集群 VC_i 中分配给任务类 c 的虚拟机数量。于是我们可以得到公式（4.1）：

$$\beta_i^{(c)} = \theta_i^{(c)} \Big/ p_i^{(c)} \tag{4.1}$$

其代表虚拟集群 VC_i 中任务类 c 的单位时间处理量，即处理效率。于是我们亦可以得到公式(4.2)中虚拟集群 VC_i 中任务类 c 的剩余执行时间：

$$t_i^{(c)} = \delta_i^{(c)} \Big/ \beta_i^{(c)} \tag{4.2}$$

基于此，从全部 I 个虚拟集群来考虑，我们可以获得任务类 c 的剩余总执行时间，用公式（4.3）表示：

$$t^c = \max\left\{t_1^{(c)}, t_2^{(c)}, ..., t_I^{(c)}\right\} \tag{4.3}$$

在所有的 C 个任务类上最小化最大的执行时间为我们调度问题需要处理的第一个目标。

同时，我们用 $d_i^{(c)}$ 来表示虚拟集群 VC_i 中任务类 c 使用一个虚拟机的内存需求量，则虚拟集群 VC_i 中任务类 c 使用虚拟机的内存总需求量可以用公式（4.4）表达为：

$$D_i^{(c)} = d_i^{(c)}\theta_i^{(c)} \tag{4.4}$$

基于此，我们得到本优化问题的第二个调度目标，既在给定任务类数量和虚拟集群的方案中，目标是使得总内存消耗量最小。用公式（4.5）来表达两个调度目标为：

$$\begin{aligned} J_1(\theta_i^{(c)}) &= \max_{c\in[1,C]}\max_{i\in[1,I]}\left(\delta_i^{(c)} * p_i^{(c)}\right)\Big/\theta_i^{(c)} \\ J_2(\theta_i^{(c)}) &= \sum_{i=1}^{I}\sum_{c=1}^{C} D_i^{(c)}\theta_i^{(c)} \end{aligned} \tag{4.5}$$

假设第 i 个虚拟集群采用 m_i 个虚拟机，则我们的目标是选择一种虚拟机分配策略 $\theta_i^{(c)}$，使得所有任务的完成总时间和内存总消耗量都能尽可能的小，其表达为公式（4.6）。

$$\begin{aligned} &\min_{\theta_i^{(c)}} J_1 \quad \min_{\theta_i^{(c)}} J_2 \\ &s.t. \sum_{c=1}^{C}\theta_i^{(c)} \le m_i \end{aligned} \tag{4.6}$$

对于以上问题的求解，我们的最优化问题实际是带有随机参数的仿真优化问题，其中的期望执行时间 $p_i^{(c)}$ 和单位虚拟机的内存需求量 $d_i^{(c)}$ 是需要提前仿真的不确定性参数，于是随机仿真优化问题既转化为如下公式（4.7）基于均值优化问题：

$$\min_{\theta_i^{(c)}\in\Theta}\left\{J_l\left(\theta_i^{(c)}\right)\right\} = \min_{\theta_i^{(c)}\in\Theta}\left\{E_{p_i^{(c)}}\left[E_{d_i^{(c)}}\left[J_l\left(\theta_i^{(c)}; p_i^{(c)}, d_i^{(c)}; T\right)\right]\right]\right\} \quad l=1,2 \tag{4.7}$$

上式中 $J_l\left(\theta_i^{(c)}; p_i^{(c)}, d_i^{(c)}; T\right)$ 代表一次采样仿真轨迹后获取的性能取值，经过多次仿真后，我们在所有的 $p_i^{(c)}$ 和 $d_i^{(c)}$ 维度求的平均值来评估当前的 $\theta_i^{(c)}$ 性能。鉴于此，我们需要引入两种估计模型来确定仿真性能。

定义一：（理想测试性能）。由于存在大量的随机因素，我们对每个 $\theta_i^{(c)}$ 的随机参数采样 N 次来获取平均性能。通常 N 预设一个比较大的数值，后面实验中设置 N 为 1000，然后将 1000 次的任务总执行时间和内存总消耗量两个性能结果进行数值平均来获取真实性能。本部分的测试结果成为理想测试性能，我们这里也称之蒙特卡洛（Monte Carlo）仿真采样性能。

定义二：（真实测试性能）。真实系统测试方案无法采用 N 次试验获取结果，取而代之使用 n 次试验的平均值来获取。通常 n 是远小于 N 的数值，本评估也称之为粗糙模型评价。粗糙模型和真实模型之间的差异性则通过一系列数学分析方法，比如Ω类型和噪声等级来刻画，从而获取到本应用的问题类型。最终通过对此类数学分析方法的应用来获取最终的决策策略。Ω类型和噪声等级在 4.4.2 节中会详细介绍。

4.3　虚拟集群仿真优化调度问题困难点

弹性云平台的调度问题从形式上来看并不复杂，但是鉴于本问题的虚拟机的性能随着工作负载的变化差异很大，无法用统一的处理能力来模拟运行时状态求解，于是我们选择基于仿真优化的方法，即给定工作负载和虚拟机当前的运行状况，来仿真当前情况下虚拟机的分配策略。针对于此类仿真优化问题，我们依然面对着如下四点困难：

（1）巨大的搜索空间问题：以虚拟集群 VC_i 为例，其分配了 m_i 个虚拟机，则对应于在其上运行的 C 个任务类，有调度策略的数量用公式（4.8）表示为：

$$|\Theta_i| = H(C, m_i - C) = (m_i - 1)!/((m_i - C)!(C - 1)!) \quad (4.8)$$

基于此，则整个调度系统的搜索空间的数值用为公式（4.9）表示为：

$$|\Theta| = \sum_{i=1}^{I}\left[(m_i - 1)!\left((m_i - C)!(C - 1)!\right)\right] \quad (4.9)$$

这个搜索空间的取值随着问题规模的扩大会变得非常大。我们仅以一种非常小规模的虚拟集群（I=1）为例，假设当前有 20 台虚拟机，并且有 7 个任务类，则带入上式得到需要仿真的策略数量是 27132 个。假设一个策略仿真时间是 2 秒，则仿真完成整个问题需要 15 个小时以上，这在实际调度问题是无法接受的；

（2）大规模不确定性参数：由于诸如有限网络带宽的限制、共享 CPU、内存和 I/O 读写的实际处理能力的动态性，本问题的期望执行时间 $p_i^{(c)}$ 和单位虚拟机的内存需求量 $d_i^{(c)}$ 是需要提前仿真的不确定性参数，一般需要采用蒙特卡洛仿真在每次试验前确定其取值大小，以上参数的规模数量是 $I \times C$。如此大规模的随机参数的仿真优化问题也使得求解尤为困难；

（3）多目标评价：本问题使用的最小化总执行时间和最小化总内存消耗两个互相冲突的优化目标，需要建立 Pareto 最优概念在两个方向上寻找非劣前沿，其难度远大于传统单目标可用数值计算直接比较的问题；

（4）结构信息的或缺：由于弹性云平台任务调度问题对于计算资源的依赖性和敏感性问题在以往的相关工作中并未有过多的涉及，所以对于此类结构信息的缺乏也加剧了此类问题的求解难度，我们构建的调度模型只是对于本调度问题的一个试探性的尝试。

4.4　低开销调度（LOS）仿真优化和其他方法

4.4.1　蒙特卡洛法和盲选法

为了便于比较，本文中采用的基于蒙特卡洛方法实际上是一种暴力破解的手段，其对每个候选决策都仿真 N 次，获取其关于 N 次的平均性能，然后计算出其非劣最优策略集，最后随机选择最优策略集中的一个作为当前策略来使用。本方法能相对简单的执行，并且每次产生的策略在基于当前工作负载情况下是优化的设计方案，不足之处在于其高昂的时间开销导致其无法适应工作负载的快速变化。

其算法步骤如下：

对于当前未评估决策 $\theta_i^{(c)} \in \Theta$

（1）随机产生 $p_i^{(c)}$ 和 $d_i^{(c)}$ 的取值；

（2）利用 $(p_i^{(c)}, d_i^{(c)})$ 对仿真 $\theta_i^{(c)}$；

（3）计算(2)中 $\theta_i^{(c)}$ 仿真产生的任务总执行时间和内存消耗量；

（4）返回(1)步重复执行 N 次；

（5）计算 N 次仿真的评价总执行时间和评价内存消耗量。

盲选方法（Blind Pick）又称为随机游走方法[21]，也可以作为一种候选方案。其特点在于设定以较大概率（比如 98%）的成功率，根据样本空间的大小随机选择一个集合 S_{bp}，然后对于 S_{bp} 中的任意一项决策进行 N 次蒙特卡洛仿真获取其理想性能特性，最终从 S_{bp} 中的非劣最优策略中随机选择出一个方案作为当前决策。盲选方法产生的 S_{bp} 小于暴力破解蒙特卡洛的全局解空间Θ，所以其开销比蒙特卡洛方法小很多。但是本方法的成功率毕竟是以概率保证，所以其实质是一种牺牲确定性换来时间开销降低的方法。

4.4.2　LOS 的基本概念

LOS 方法的基础是向量序优化方法[121]。介绍本方法的实施步骤之前，我们先探讨几个涉及到的核心概念。

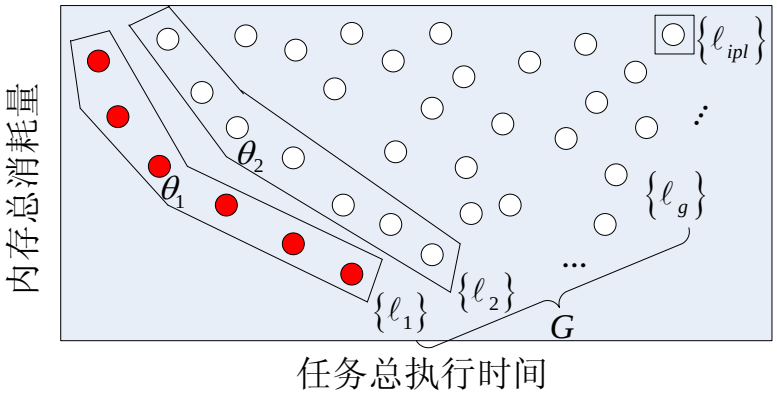
定义三：（不可比性）。θ_x 和 θ_y 具有不可比性，当且仅当 $J_l(\theta_x) < J_l(\theta_y)$

同时 $J_l(\theta_x) > J_l(\theta_y)$ 或者 $J_l(\theta_x) > J_l(\theta_y)$ 同时 $J_l(\theta_x) < J_l(\theta_y)$，不可比性概念的构建导致多目标评价具有困难，因此我们引入下述各类概念来构建策略之间的相互关系。

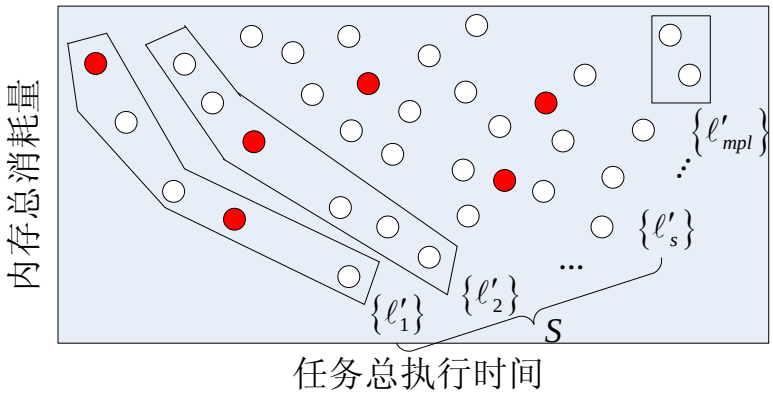
定义四：（优超）。θ_x 优超 θ_y ($\theta_x \succ \theta_y$)，如果$\forall\, l \in \{1, 2\}$，$J_l(\theta_x) \leq J_l(\theta_y)$，同时 $\exists\, l \in \{1,2\}$，$J_l(\theta_x) < J_l(\theta_y)$，根据定义，我们可以得到图 4.2 中$\theta_1$ 优超 θ_2。优超概念使得策略类之间具有相对的优先关系，这样在二维坐标下建立了相应的“序”关系，使得调度方案变得可比较。

定义五：（非劣最优策略集合）。对于策略集 l 中的任何一个θ_x，在全局策略空间Θ中如果不存在任何一个θ_y，使得$\theta_y \succ \theta_x$，则称 l 中的解均为非劣最优策略，也称非劣前沿策略。图 4.2（a）中的 l_1 构成非劣最优策略集合。非劣最优策略的引入源于在后续定义六种构建的满意策略的概念。

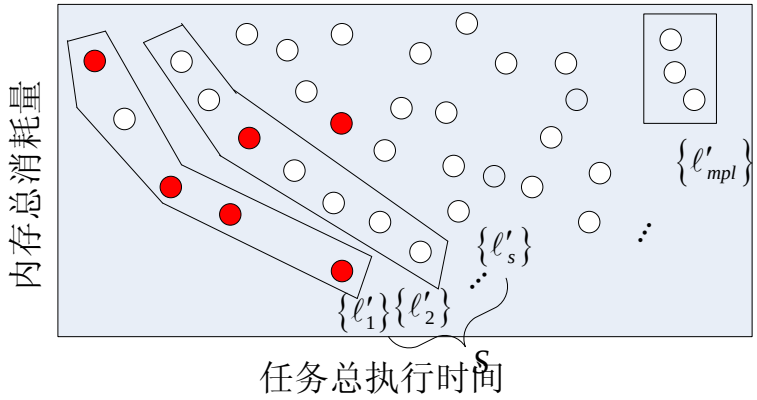
定义六：（满意策略集合 *G*）。全局策略集空间Θ中的非劣最优策略定义为满意策略。在图 4.2（a）中，所有的深色点集构成了满意策略集合。一般情况下，仅仅选择非劣前沿策略的数量并不足以构成满意策略的集合，于是可以选择前 g 层的策略构成满意策略集 G：G={{l_1},{l_2},…,{l_g}}。在本调度问题中，我们设置 g=1 构成满意策略集，其原因是我们面对的搜索空间巨大导致这种情况下非劣最优策略集合中包含有足够多的调度策略可供选择。



(a) 基于理想测试性能（蒙特卡洛方法）的评估模型



(b) 基于较小开销的真实测试性能



(c) 基于较大开销的真实测试性能

图 4.2 理想测试性能和真实测试性能示意

定义七：（选择集合 *S*）。选择在真实测试性能的前 *s* 层的所有策略构成了选择集合 *S*。在图 4.2（b）和 4.2（c）中，$S=\{\{l'_1\},\{l'_2\},\ldots,\{l'_s\}\}$。

对于 4.2（b）中仿真结果示意，由于其采用了较小次数的仿真结果，则对于真实非劣最优性策略集中的决策（红色的点）稀疏的散布在平面中，于是需要采用大的集合 *S* 来覆盖；而对于 4.2（c）中的仿真结果示意，则采用了较大的仿真开销导致真实非劣最优性策略集中的决策可以聚集在比较小的 *S* 集合中。此处约定，选择集合 *S* 加上调度方法的下标代表本方法选择出的 *S* 集合。例如 LOS 方法得到的 *S* 集合为 S_{LOS}，以此类推。

定义八：（Ω 类型）。在向量序优化中，本概念称之为排序性能曲线，既描述测试性能的决策在解空间的分布形状。Ω 类型通常被归纳成如下三种：平坦型、中间型和陡峭型。有关Ω类型的详细介绍和描述参见文献[121]。

定义九：（噪声等级）。噪声等级衡量理想测试性能和真实测试性能之间的差异性，其在公式（4.10）中定义如下：

$$NL\left(J_l\left(\theta_i^{(c)}\right)\right)=\frac{\max\limits_{\theta_i^{(c)}\in\Theta}\left(\left|J_l\left(\theta_i^{(c)}\right)-\hat{J}_l\left(\theta_i^{(c)}\right)\right|\right)}{\max\limits_{\theta_i^{(c)}\in\Theta}J_l\left(\theta_i^{(c)}\right)-\min\limits_{\theta_i^{(c)}\in\Theta}J_l\left(\theta_i^{(c)}\right)}\qquad l\in\{1,\ \}\qquad（4.10）$$

上式中分母代表真实观测性能和理想观测性能的最大差异值，分母代表真实观测性能的最大和最小值之差。其实际意义在于表达决策 $\theta_i^{(c)}$ 在本次真实观测的中所产生的最大误差和决策 $\theta_i^{(c)}$ 在所有的理想观测中的比率。在我们的实际应用中，观测性能在实验之前是不可获知的。于是我们采用了真实观测性能的均方差来计算，即按照公式（4.11）求解如下：

$$NL\left(J_l\left(\theta_i^{(c)}\right)\right)=\sqrt{\frac{\sum_{\theta_i^{(c)}\in\Theta}\left(\hat{J}_l\left(\theta_i^{(c)}\right)-E\left[\hat{J}_l\left(\theta_i^{(c)}\right)\right]\right)^2}{|\Theta|-1}}\qquad l\in\{1,\ \}\qquad（4.11）$$

获取了以上各项参数的取值以后，文献[121]中定义了选择集合 S 的大小 s 和以上各参数之间的关系，从而我们选择在观测性能中排名前 s 层的策略构成选择集合 S，如图 4.2(c)所示。在获取了选择集合 S 以后，需要用蒙特卡洛仿真的方法对其做理想观测性能评估，从而选出最后使用的调度方案。

4.4.3　LOS 的实现步骤

如果将上述两种方法比作一段频谱，暴力破解的蒙特卡洛方法位于频谱的一侧极点，则盲选方法位于频谱的另外一侧极点。LOS 方法则是这两种方法的折衷，位于两极点之间：其既不是对全局策略空间中的每一个决策Θ做 N 次暴力蒙特卡洛仿真获取其性能，亦不是完全以带有随机性的方法去盲选。相反，其采用 n 次仿真的粗糙模型进行一次评估，筛选出性能较优的决策集合 S，然后在 S 中进行 N 次蒙特卡洛仿真获取相应决策。

其实现步骤主要由如下三个算法分别组成：

算法 4.1：策略排序算法

输入：

策略集合 $=\{\theta_1,\theta_2,\ \ldots,\theta_{|\Theta|}\}$

策略粗糙评估性能集合 $=\left\{\left(\hat{J}_1(\theta_1),\hat{J}_2(\theta_1)\right),\ldots,\left(\hat{J}_1\left(\theta_{|\Theta|}\right),\hat{J}_2\left(\theta_{|\Theta|}\right)\right)\right\}$

输出：排序以后的策略集合 $=EqualPerfList_i\ \{i=1,2,\ \ldots,|\Theta|\}$

算法步骤：

1. Forall i = 1 to |Θ|

2.　　Forall j = 1 to |Θ|, $j\neq i$,

3. **Order** *PolicyList* based on $\hat{J}_1$

4. **If** $\hat{J}_1(\theta_i)=\hat{J}_1(\theta_j)$

5. **Order** θ_i and θ_j based on $\hat{J}_2$ in *PolicyList*

6. **If** $\hat{J}_2(\theta_i)=\hat{J}_2(\theta_j)$ //第二指标性能相同则删除

7. **DeList**(*PolicyList*, θ_j)

8. **Enlist**(*EqualPerfList_i*, θ_j)

9. **Endfor**

10. **Endfor**

11. **Output**(*PolicyList*)

12. **Output**(*EqualPerfList_i*)

算法 4.2：策略分层算法

输入: 排序以后的策略集合: *EqualPerfList_i {i = 1, 2, ... ,|Θ|}*

输出:分层策略集合={l'_1}, {l'_2},…, {l'_{mpl}}

算法步骤:

1. *mpl*←1

2. **While** *PolicyList* ! = Null

3. **Output:** {l'_{mpl}} <— *PolicyList*(1) //排序第一策略分配第一层

4. **Forall** ***i* = 1 to** |l'_{mpl}|

5. **Forall** *j* = 1 to |*PolicyList*|, $j \neq i$,

6. **If** θ_i doesn't dominate θ_j

7. {l'_{mpl}} <— θ_j //将θ_i置于不能优超

8. **DeList**(*PolicyList*, θ_j) //所有策略的后续集合

9. **Endfor**

10. **Endfor**

11. *mpl*←*mpl* +1

12. **EndWhile**

13. **Forall** *i* = 1 to |Θ|

14. **If** *EqualPerfList_i* != Null

15. **If** θ_i belongs to {l'_j}

16. **Insert** *EqualPerfList_i* into {l'_j}

17. **Endfor**

18. **Output** {l'_1}, {l'_2},…, {l'_{mpl}}

算法 4.3：LOS 核心算法
输入:
决策空间：$\Theta = \{\theta_1, \theta_2, \ldots, \theta_{|\Theta|}\}$；
足够好解决的数量：g
需要获取足够好策略的个数：k
需要获取 k 个足够好策略的概率：　α
粗糙评价实验次数：n
输出:
可供当前使用的良好决策策略
算法步骤:

1. **Forall**　$i = 1 : |\Theta|$
2. **Test** each schedule θ_i n times　　//粗糙评估 n 次实验结果
3. **Output** time and cost as　$\{\hat{J}_1(\theta_i), \hat{J}_2(\theta_i)\}$　　//粗糙评估性能
4. **Endfor**
5. **Plot**　$\left\{\hat{J}_1(\theta_i), \hat{J}_2(\theta_i)\right\}$ $\{\hat{J}_1(\theta_i), \hat{J}_2(\theta_i)\}$　($i \in [1 : |\Theta|]$) in x-y axis
6. **Plot** the measured performance as Fig 4.3(b)　　//策略排序和分层
7. **Calculate** the Ω and the noise level //计算表征策略分别的两参数
8. **Adjust** g, k to get g', k' //步骤 8 到 11 获取和回归实验一致参数
9. **Look up** the coefficient table to get (Z_0, ρ, β, η)　//获取回归参数
10. **Calculate** the initial s'
11. **Adjust** s' to get the related s
12. **Test** the policies at front s layers, $\{\theta_{[1]}, \theta_{[2]}, \ldots, \theta_{[s]}\}$ to select the policies we need.

以上比较可以得出，三种方法的调度策略都可以划分两个阶段：第一阶段为采用粗糙模型进行评估的阶段；第二阶段为采用理想模型进行评估的阶段。对于暴力破解的蒙特卡洛方法，其第一阶段的粗糙模型即为真实模型，而第二阶段则无任何评价；盲选方法第一阶段事实上无任何评价，第二阶段将选出的 S_{bp} 进行逐一理想性能评价；而 LOS 方法在两阶段上均进行评价。

4.5　实验设置与结果分析

4.5.1　测试平台实验方案设置

基准测试程序采用的是激光干涉引力波天文台中实现引力波验证逻辑

的并行工作负载，每一项任务类需要验证与之相关的业务逻辑关系，包含许多可执行的小任务可以进行调度，不同的任务完成时间和资源的占用情况各不相同。在本实验中一共有七项业务逻辑需要验证，对应于七种不同的任务类。业务逻辑之间可以视之为独立运行的工作负载，由于具体的业务逻辑内容本身并不是本章关注内容，这里不再赘述，具体每项业务的内容和需要验证的逻辑可以参见文献[25]中的工作。

对于不同的任务类，有不同数量的并行任务。任务为调度和执行的基本单元，即将所有并行任务合理的分配到虚拟集群中执行。在表 4.2 中，总结了七种任务类并行任务的数量特征。

表 4.2 虚拟物理集群配置

任务类编号	主要功能	并行任务数量
1	模板库生成的后续操作验证	3,576
2	干涉仪的工作状态约束验证	2,755
3	偶然性分析的完整性验证	5,114
4	偶然性分析的必然性验证	1,026
5	服务可达性约束验证	4,962
6	服务可终止性逻辑验证	792
7	服务变量垃圾回收验证	226

4.5.2　实验结果分析

首先我们分析本问题的Ω类型，即排序性能曲线。在向量序优化中将Ω类型分为三种，既陡峭型（Steep）、中和型（Neutral）和平坦型（Flat），通过对观测性能进行分析和评估，在二维平面上画出Ω类型的曲线。

在图 4.3 中，我们作出了Ω类型曲线。通过图 4.2 中的分层思想，将观测性能在空间中的策略分布按照 Pareto 最优性逻辑关系分成了 152 层，层的标号在图中用横轴 x 表示。而纵轴 y 的数量则代表观测性能中的前 x 层总共含有的策略的个数，与横轴 x 为 152 层对应的 y 值显然是全空间策略的总数 27,132（以 20 个虚拟机实例作为初次实验分析）。

从Ω类型曲线的结果中，可以很明显看出本应用属于陡峭型（Steep）问题。其实质是大多数策略都更多的倾向于集中在 x 比较小的层中，而在 x 比较大的取值时，y 相对变化不太明显，这一点可以从图上明显的看出。对于

本类问题的好处在于，更多的策略集中在 Pareto 非劣前沿层次附近，使得我们从 Pareto 非劣前沿逐层往后搜索时，能够将更多（前面较少数量的层集中了较多的策略）更好（集中在前面若干层的策略均具有很好的性能：较低的任务总执行时间和总内存消耗量）的策略选择进选择集合 S 中，使得后期针对于 S 中的策略仿真能够以更大的概率获得综合性能比较好的策略，这一点是盲选方法无法企及的。

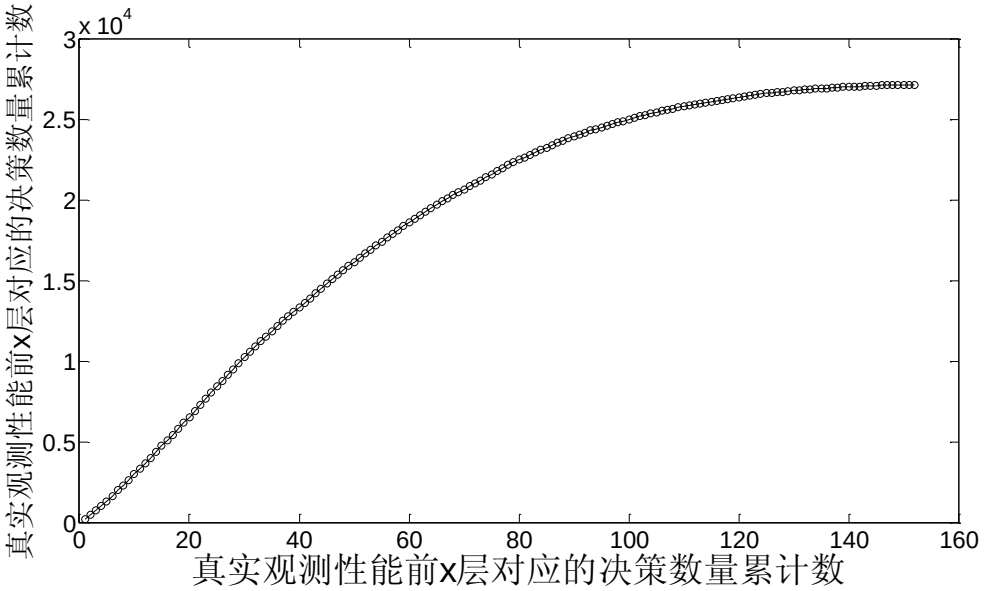


图4.3　Ω 类型

关于噪声等级，我们在任务的总执行时间和任务的总内存消耗两个维度进行分析，采用公式（5.11）可以很方便的进行计算。首先将任务总执行时间维度的所有时间归一化到[-1,1]区间中，然后计算出标准差为 0.4831；同样将内存总消耗量也计算出其观测性能的标准差为 0.3772，由此我们可以推断最大的标准差取值为 0.4831 在[0,0.5]之间属于小噪声类型。

事实上，通过前述分析我们已经能够获取本类应用采用向量序优化方法求解的优越性所在，即具有陡峭型Ω类型和较小的噪声等级的情况下，本问题类型能够通过较少的粗糙计算量换来良好的策略。

为了验证 LOS 方法是否真能搜索到部分非劣最优策略集中的决策，我们做出了如下实验来验证。首先我们对于每个决策 $\theta_i^{(c)}$，利用 1000 次实验评价其真实性能（图是中的黑点代表策略），然后将其画在二维坐标下，同时标注出其非劣最优策略集中的策略（图是中的蓝色圈代表理想测试性能中的非劣最优策略集 G）。由于标注范围较大，于是在图 4.4 中，我们截取了其具有代表性的一小部分。同时我们也将 10 次真实测验的性能结果画出来，

根据 LOS 算法求解出其 S 集合（图是中红色大圈标记的部分），然后将两幅图进行对照，可看出既有蓝色小圈标注又有红色大圈标注的决策是 $G \cap S$ 具有重叠部分的策略。可以看出，利用 LOS 方法，能够挑选出很多理想测试性能的非劣最优策略。

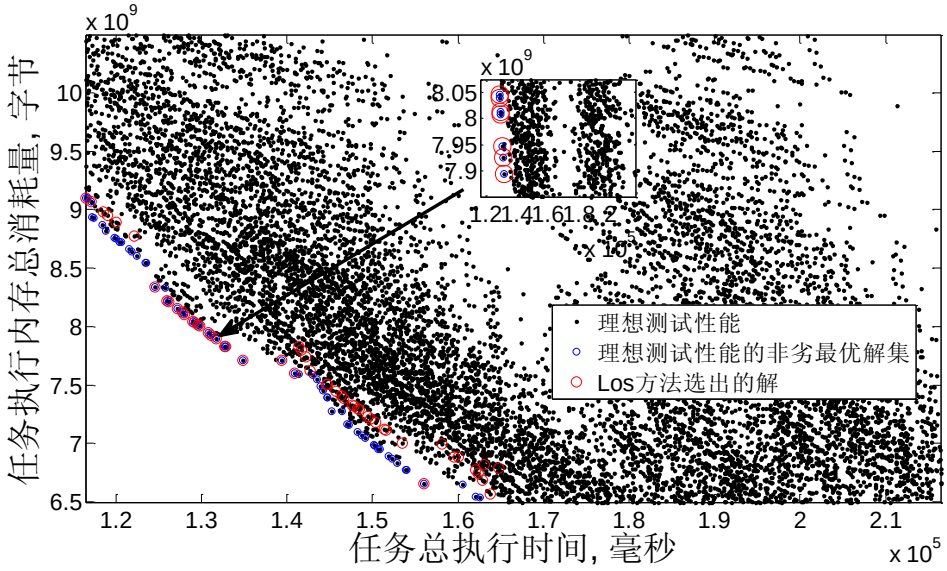
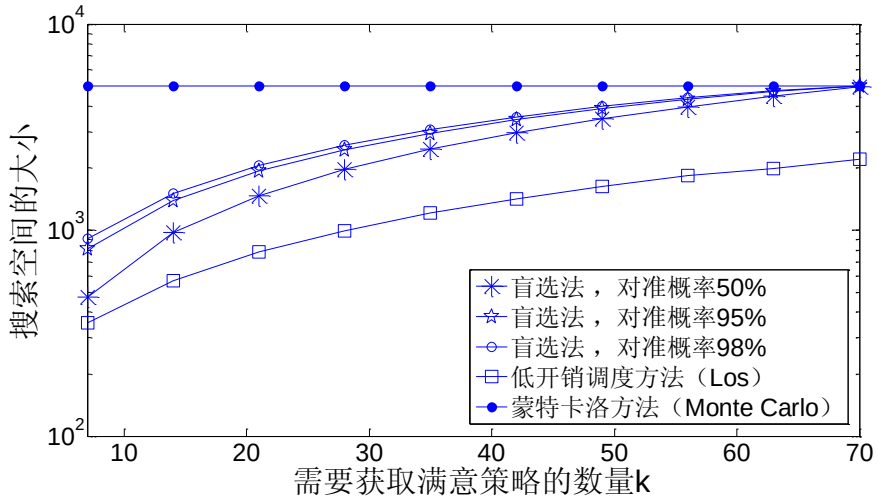
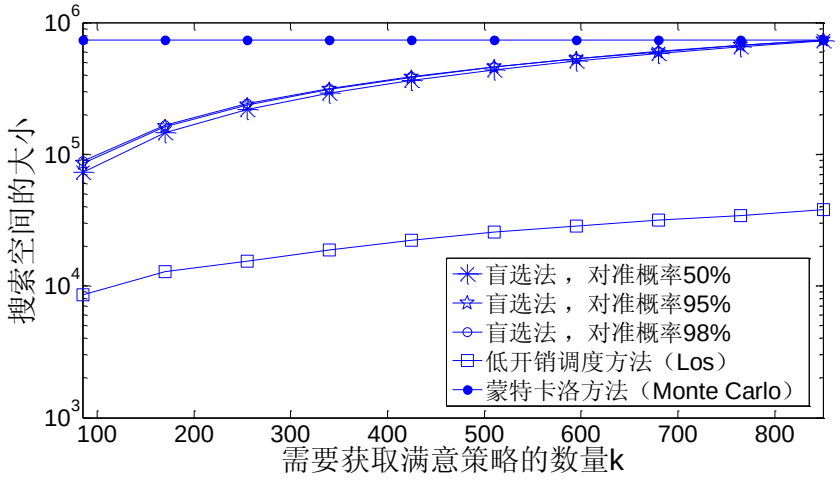


图4.4　LOS方法挑选出理想测试性能中具有非劣最优解策略中的方案

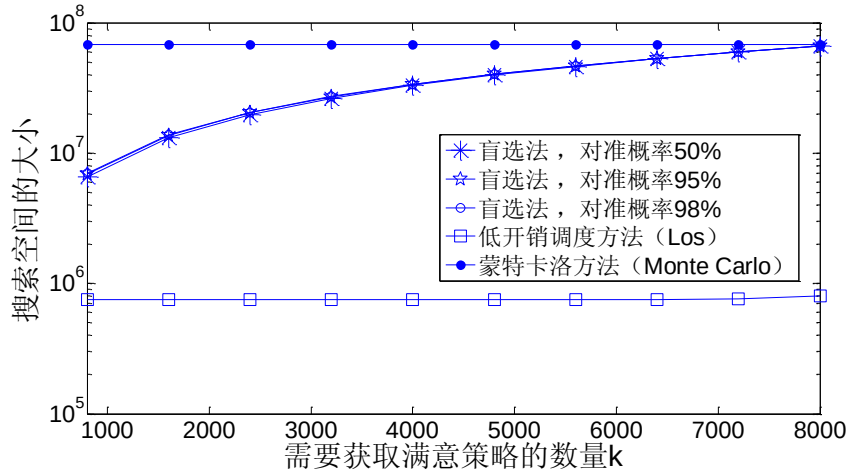
因为本类型调度问题中我们定义的满意策略集合包含的策略数量不止一个，所以为了体现各种方法的异同，可以设置在三种方法实施后，获取满意策略集中策略数量的百分比作为性能评价指标并评估其调度时间开销。此百分比数值越大，则对应的搜索难度越大，所需要的评估时间也就相应越长。



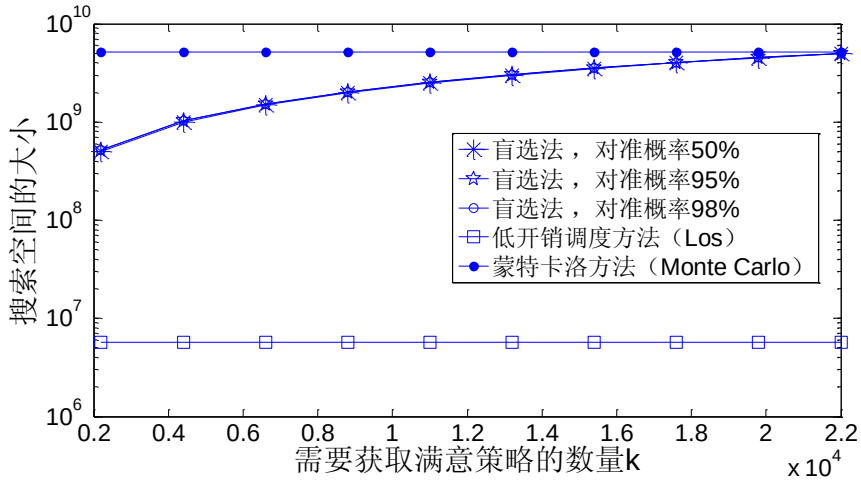
(a)　16个虚拟机



(b)　32个虚拟机



(c)　64个虚拟机



(d)128个虚拟机

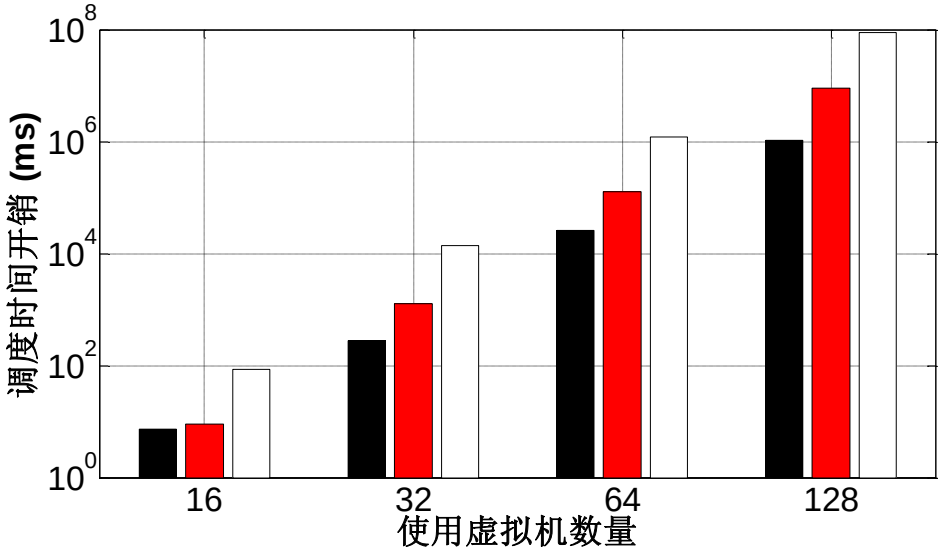
图4.5　在不同数量虚拟机下搜索空间大小随着需要获取满意策略大小的变化

图 4.5 比较了三种方法在对于目标策略空间的降低随着需要获取满意解数量下的变化趋势。盲选法中，我们分别比较了 50%、95%和 98%三种情况，并和低开销调度方法 LOS 以及蒙特卡洛方法 Monte Carlo 进行了对比。（a）到（d）图则用于测试上述比较在不同虚拟机数量下的可扩展性分析。从图中可以看出，低开销仿真优化方法相比其他各种方法明显小于其他方法的调度开销，且此优势随着需要对准的满意策略数量的增多而变得更大。

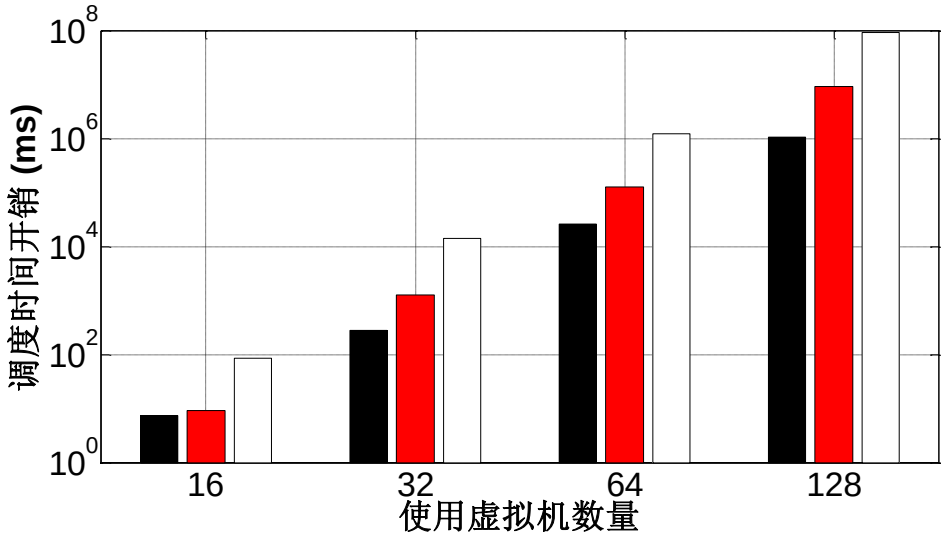
图 4.6 分别显示需要获取满意策略集合中 10%、50%和 90%的策略时，对应于总数量分别为 16、32、64、128 的虚拟机，利用三种仿真优化方法所需要的调度时间开销。黑色柱状性能代表利用 LOS 方法的开销，其性能为三种方法中最优的，而代表暴力破解的蒙特卡洛方法的时间开销是三种方法中最大的。同时，从（b）和（c）图中可以看出，LOS 方法具有较好的可扩展性。在虚拟机数量为 16 的情况下，其调度开销和其他方法相差不大，而当虚拟机数量成指数规模扩大时，其时间开销明显比其他方法增长的慢。

事实上，蒙特卡洛和盲选法的时间开销均可通过公式近似计算，LOS 方法的时间开销则取决于Ω类型和噪声等级。利用 LOS 选择出的 S 集合较小时，相应后阶段评价 S 中策略的时间也会相应少，从而开销更小。本问题良好陡峭的Ω类型和较小的噪声等级使得 LOS 方法相比之下优势明显。

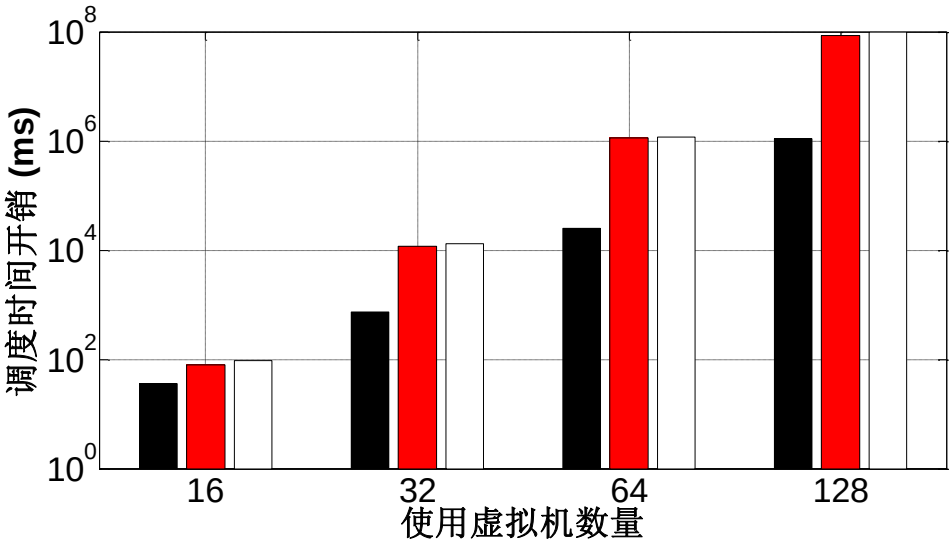
对于其他Ω类型或不同的噪声等级情况下，LOS 方法不一定能够发挥出良好的调度性能，这一点是显而易见的。因为在噪声非常大的极端情况，各种方法都退化成盲选法，而无任何仿真优化可言，所以对于任何问题，我们需要首先研究其问题类型，然后来判断是否适合低开销仿真优化方法来优化。



(a)需要找到10%足够好的解



(b) 需要找到50%足够好的解



(c) 需要找到90%足够好的解

图4.6　低开销仿真优化方法性能比较

4.6　本章小结

由于存在第三章模型未考虑随机性因素的实际问题，在本章的弹性云平台多任务调度优化问题中我们将其引入。面对大规模的决策搜索空间、诸多随机变量的存在、多目标评估以及有限的先验知识导致问题的求解非常繁琐

耗时。针对以上问题，本章首先提出了弹性云平台下仿真随机优化模型，刻画了调度策略和和任务总执行时间以及任务总内存消耗之间的联系，通过引入粗糙模型的思想来处理带有随机因素的仿真优化问题，从而整体降低仿真开销，缩短调度时间。在很短的决策阶段，能产生一个良好的策略集，本方法能以比较大的概率保证策略集中含有真实测试性能非劣最优策略集中的决策，从而能够保证调度整体性能。

本方法通过判定Ω类型和噪声等级两个指标来确定是否在本类问题中利用 LOS 方法可以发挥优势。事实上，在面对平坦型Ω和大噪声问题时，粗糙模型的优势发挥的并不明显，其性能最终和盲选方法差别不大，调度开销甚至会超过盲选法。而本应用实例中的工作负载是典型的陡峭型的小噪声问题，实验结果将 LOS 的低开销优势彰显无遗。

诚然本方法具有低开销性的好处，但是如何将其合理的进行延拓使之适应动态变化的工作负载，从而达到其提升评价指标的性能则是体现其优势的最重要方面。后面几章致力于将其拓展到多阶段连续迭代的工作负载中，使其发挥作用。

第5章　弹性云平台资源低开销迭代仿真优化供给

5.1　本章引论

第四章论述的弹性云平台资源的低开销调度方法能够通过构建合理的仿真模型来将可行策略的搜索空间降低到一个小的满意集合之内，从而系统能够更加敏捷的获取足够好而非一定获取最优性策略。从本章开始考虑如何合理的利用本低开销性调度优势，序贯性、实时性、动态性的产生一系列决策策略，以提升系统的整体性能。

LOS 方法主要的好处在于其实质上是一种低开销的调度方法，能够将决策阶段缩得很短。虽然每个决策阶段产生的决策本身为非最优，但是由于其低开销性，本方法能够很好的适应和捕获工作负载的局部和动态特性，从而在整个调度阶段中提升了全局性能。

本章首先分析三种调度方法的调度开销，证明 LOS 方法相对于其他方法的低开销性并给出经验数值。然后采用前面章节的多任务验证试验在吞吐率和响应时间两个性能指标进行验证，实验结果显示了 LOS 方法能够敏捷快速的在决策的每个阶段产生一系列处于非劣前沿的满意解集，综合实验结果验证了方法在理论上量化证明了方法的低开销性并迭代的产生不断适应于多任务负载变化的策略。

本章的组织结构如下：5.2 节将三种调度方法的开销进行了量化比较；5.3 节阐述了低开销迭代调度的实现原理及其优势所在；5.4 节介绍低开销迭代调度的算法并和迭代蒙特卡洛仿真方法和迭代盲选法进行了分析；第 5.5 节对本章针对于吞吐率和响应时间两个衡量指标，将实验结果进行分析；第 5.6 节是本章工作的总结。

5.2　三种方法调度开销对比

第四章介绍了蒙特卡洛仿真方法、盲选法和低开销调度 LOS 方法。现在总结比较以上三种方法的调度时间开销（Overhead），其定义为从程序运行开始到产生调度方案位置的总时间间隔。为了使得各种方法在时间开销上具有可比性，我们做如下简化：假设单次评价一个策略所需要的时间开销为 o，于是在进行时间开销的比较之前，我们先证明如下定理：

定理一：采用盲选方法的调度时间开销平均意义下为：

$$O_{bp}=\frac{k\times|\Theta|\times N\times o}{|G|}$$

证明：对于盲选方法，其对准概率符合超几何分布，既：

$$P\left[\left|G\cap S_{bp}\right|=k\right]=\frac{C_g^k C_{|\Theta|-g}^{|S_{bp}|-i}}{C_{|\Theta|}^{|S_{bp}|}}$$

而当给定满意策略集 G 和所需要对准等级 k 以后，则其获取决策个数的期望为：

$$E\left[\left|G\cap S_{bp}\right|\right]=\frac{\left|S_{bp}\right|\times g}{|\Theta|}=k$$

求解上式便得证。 □

定理二：LOS 方法比暴力破解蒙特卡洛方法和盲选方法的调度开销小，需要如下条件得以满足：

$$\frac{n}{N}+\frac{|S_{Los}|}{|\Theta|}<\frac{k}{|G|}+1$$

分别列举出三种方法的调度开销：

对于暴力破解蒙特卡洛方法，无第二评价阶段，即：$o_1=|\Theta|\times N\times o \quad o_2=0$

对于盲选方法，无第一评价阶段，即：$o_1=0 \quad o_2=\left(k\times|\Theta|\times N\times o\right)/|G|$

对于 LOS 方法，其拥有两评价阶段，且：$o_1=|\Theta|\times n\times o \quad o_2=|S_{Los}|\times N\times o$

将上述算式联立使得 LOS 开销最小既得定理二结论。 □

事实上，定理二较容易满足。其理由在于一般情况下，$n<<N$ 同时 $S_{LOS}<<|\Theta|$，以上两式相加即可小于 1，故而 LOS 方法的开销一般会比另外两种方法开销小，这就是低开销调度的缘由。

根据以上分析，可以得出 LOS 方法比暴力破解蒙特卡洛仿真和盲选方法快的倍数分别为：

$$R_1=\frac{O_{Monte}}{O_{Los}}=\frac{N}{n+\frac{|S_{Los}|}{|\Theta|}N}$$

$$R_2=\frac{O_{BP}}{O_{Los}}=\frac{k|\Theta|N}{|G|\left(|\Theta|n+|S_{Los}|N\right)}=\frac{k|\Theta|}{|G|\left(|\Theta|\frac{n}{N}+|S_{Los}|\right)}$$

结论：当 $n<<N$ 时 R_1 和 R_2 分母左边项趋近于 0，于是得到加速比的估计值：

$$R_1 = \frac{|\Theta|N}{|\Theta|n+|S_{Los}|N} = \frac{|\Theta|}{|\Theta|\frac{n}{N}+|S_{Los}|} \approx \frac{|\Theta|}{|S_{Los}|}$$

$$R_2 = \frac{k|\Theta|N}{|G|\left(|\Theta|n+|S_{Los}|N\right)} = \frac{k|\Theta|}{|G|\left(|\Theta|\frac{n}{N}+|S_{Los}|\right)} \approx \frac{k|\Theta|}{|G||S_{Los}|}$$

$n<<N$ 是通过实验设计显然成立的，对准策略的个数 k 的取值一般情况下小于或者等于足够好策略集合 G 的大小，而利用 LOS 方法的选择集合也会将整个搜索空间降低所以$|\Theta|<<S_{LOS}$。所以此情形下，根据 LOS 求解问题类型，可以估计出搜索空间$|\Theta|$的大小，从而便于考虑是否采用 LOS 方法能够体现出调度优势。

在后续实验中，采用了 20 个虚拟机，7 个任务类的弹性云平台调度环境设计实验中，我们有$|\Theta|$ = 27,132，N=1,000，s =190，n = 10，g =12 以及 k = 1。这样通过 LOS 方法所降低的调度开销的比率是： R_1 = 27,132/190 = 142.8 和 R_2 = 11.9。

5.3　LOS 方法的调度优势

随机到达的动态任务导致的工作负载随着时间是不断变化的，这种变化特性一般是很难捕捉到的，即比较难以在较短时间内提供较优良的决策方法来适应变化的工作负载，尤其是当负载强度变化非常剧烈时情况更为严重。而实际情况是，对于工作负载局部特征的细粒度仿真能在整体上更有利于选择满意策略。云环境下的多任务负载由于其任务之间的低耦合性，使其属于一种可任意时间调度的应用类型，而多阶段迭代的敏捷调度方法对于本问题类型恰到好处。

在证实了方法的低开销性以后，一个亟待解决的问题是如何利用低开销性和系统工作负载的动态时变性达到整体优化的目的。LOS 方法的主要特点是用迭代的方式将本章设计的仿真序优化方法应用到了调度过程中，每次迭代都是一次根据局部工作负载情况进行优化的过程，迭代优化的周期越短，选择的策略性能越能更加细粒度的跟上工作负载变化的局部性从而获取更加有效的策略。我们用图 5.1 来阐述这个问题。

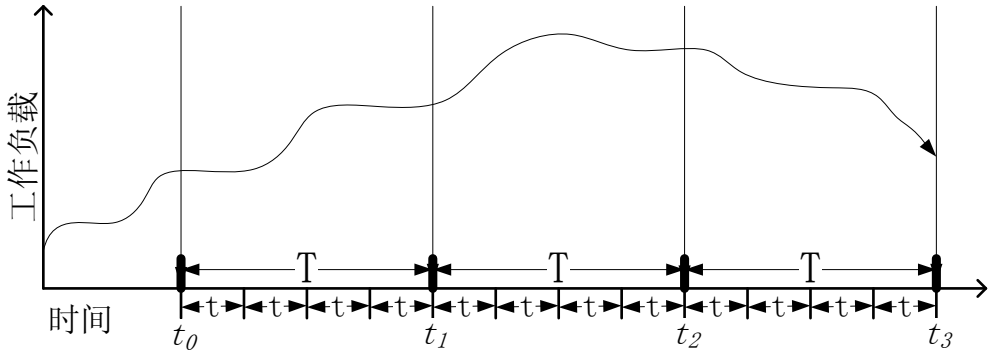


图 5.1 LOS 能够适应不断变化的工作负载

假设 T 是采用蒙特卡洛仿真方法单次调度所需要的时间开销，而 t 则是采用 LOS 方法单次调度的时间开销。在 t_0 时刻，当前蒙特卡洛方法被用于产生决策方案时，直到 t_1 时刻蒙特卡洛方法才能产生出其优化的调度方案，虽然本方法在 t_1 时刻是足够优化的（由于其仿真次数多导致受到噪声影响较小），但是由于其仿真代价巨大的缘故，t_1 到 t_2 时刻之间的任何时间点都无法再次利用蒙特卡洛方法产生中间决策输出。

但是 LOS 方法则不同，其能够在 t_1+t，t_1+2t, ….等时刻迭代的不断产生出仿真的决策。正如前面所说，LOS 方法对于局部工作负载强度的捕捉更加细腻，本方法快速的捕捉工作负载在较短时间间隔的变化规律，从而在整个时间轴上提升了系统的综合性能。

5.4　迭代仿真优化方法

5.4.1　迭代蒙特卡洛法和迭代盲选法

本节介绍介于多阶段的迭代蒙特卡洛仿真方法和迭代盲选法，这两种方法在仿真优化里很常见，也是如今我们处理云计算环境下虚拟资源自适应供给问题时使用的最多的方法。我们在算法 5.1 和 5.2 中介绍其具体的实施过程。

算法 5.1： 迭代蒙特卡洛仿真优化方法 iMonteCarlo

输入： $\Delta = \{\delta(t)=[\ \delta_1(t),\ \delta_2(t),\ldots,\ \delta_c(t),\ldots\ \delta_C(t)]\}$

输出： $U = \{\theta^*(t_i)=[\theta^*_1(t_i),\ \theta^*_2(t_i),\ldots,\ \theta^*_c(t_i),\ldots\theta^*_C(t_i)]\}$

算法步骤：

1. **Foreach** i in [1, I]　　//所有的调度阶段
2. **Load** workload $\boldsymbol{\delta_c(t_i)}$ between [t_i, t_{i+1}] for each VC c

3. **Foreach** $\theta(t_i)$ in U // 遍历每一个可行策略
4. **Foreach** VC c in [1,C] // 遍历每一个虚拟集群
5. **Simulate** $p_c(t_i)$ N times to get an average $P_c(t_i)$
6. **Simulate** $d_c(t_i)$ N times to get an average $D_c(t_i)$
7. $r_c(t_i) = \delta_c(t_i) \times P_c(t_i) / \theta_c(t_i)$ // 当前虚拟集群剩余执行时间
8. **if** (i+1< I) //不是最后一个调度阶段
9. **if** ($r_c(t_i) > (t_{i+1}-t_i)$) //部分任务未完成遗留到下一调度阶段
10. FinishedTask[c] = $(t_{i+1}-t_i) \times P_c(t_i)$
11. RemainedTask[c] = $\delta_c(t_i)$ - $(t_{i+1}-t_i) \times P_c(t_i)$
12. FinishedTime[c] = $(t_{i+1}-t_i)$
13. $\delta_c(t_{i+1})$ += RemainedTask[c]
14. **else** //所有当前阶段任务完成
15. FinishedTask[c] = $r_c(t_i) \times P_c(t_i)$
16. RemainedTask[c] = $\delta_c(t_i)$ - $r_c(t_i) \times P_c(t_i)$
17. FinishedTime[c] = $r_c(t_i)$
18. $\delta_c(t_{i+1})$ += RemainedTask[c]
19. **Endif**
20. **else** //最后一个仿真阶段
21. FinishedTask[c] = $r_c(t_i) \times P_c(t_i)$
22. FinishedTime[c] = $r_c(t_i)$
23. Makespan[i] = **max**(FinishedTime[1:C])
24. **Endif**
25. thruput($\theta(t_i)$) = sum(FinishedTask[c])/Makespan[i]
26. **Endfor**
27. **Endfor**
28. $\theta^*(t_i)$ <––Select one in Pareto(thruput($\theta(t_i)$), $\sum_{c=1}^{C} D_c(t_i)$)
29. **Go to** Step 1 for the next loop

算法 5.1 的执行过程表达了迭代蒙特卡洛仿真优化方法的使用过程。首先本方法遍历调度过程的所有阶段并载入当前阶段的工作负载，然后对于搜索空间中的每一个虚拟集群的分配决策方法在任务执行时间和内存消耗指标上仿真 N 次，如果算法执行不是最后一个调度阶段，则分析本阶段是否有工作任务遗留到下一个调度阶段并相应的做出调整。当所有分配决策方法遍

历以后，构建 Pareto 非劣策略空间并随机在非劣决策中间中选择一个作为最有策略输出并使用。

算法 5.2： 迭代盲选仿真优化方法 iBP

输入： $\Delta = \{\delta(t)=[\ \delta_1(t),\ \delta_2(t),\ldots,\ \delta_c(t),\ldots\ \delta_C(t)]\}$

输出： $U = \{\theta^*(t_i)=[\theta^*_1(t_i),\ \theta^*_2(t_i),\ldots,\ \theta^*_c(t_i),\ldots\theta^*_C(t_i)]\}$

算法步骤：

略

迭代盲选法的不同点在于根据对准概率的大小设置首先在整个可行决策搜索空间 U 中进行随机盲选获得一个 S_{bp}，然后以此为新的 U 在迭代的各个阶段执行蒙特卡洛仿真方法。算法和迭代蒙特卡洛很类似，这里略去。

5.4.2 迭代低开销仿真优化方法

算法 5.3 介绍了迭代低开销仿真优化方法的实施过程。其主要分为两个子过程，首先对于整个调度空间 U 中策略进行 n 次粗糙评估，选出满意策略集合 S 以后对其进行进行 N 次的精确评估获取使用策略。

算法 5.3：迭代低开销仿真优化方法 iLOS

输入： $\Delta = \{\delta(t)=[\ \delta_1(t),\ \delta_2(t),\ldots,\ \delta_c(t),\ldots\ \delta_C(t)]\}$

输出： $U = \{\theta^*(t_i)=[\theta^*_1(t_i),\ \theta^*_2(t_i),\ldots,\ \theta^*_c(t_i),\ldots\theta^*_C(t_i)]\}$

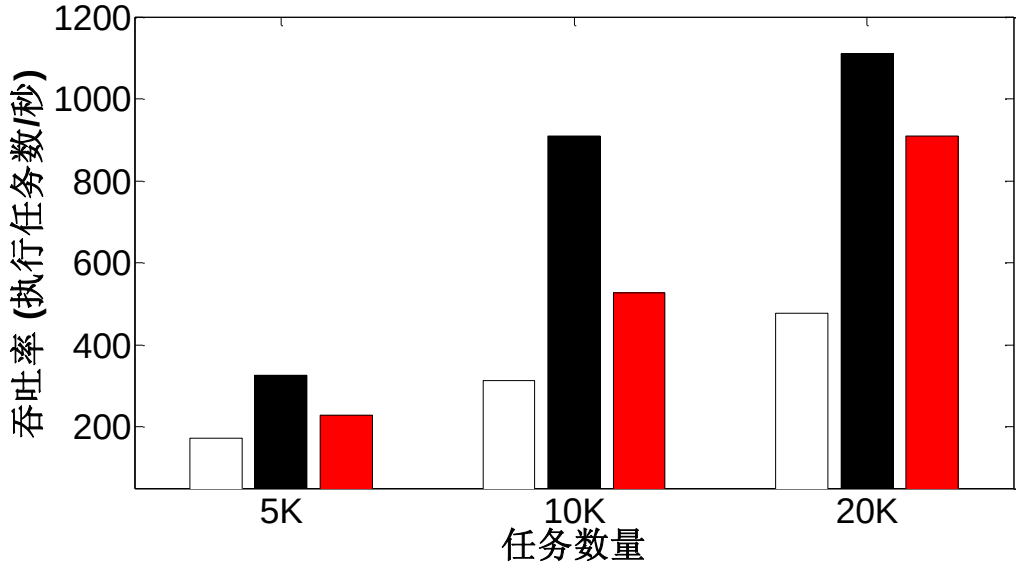
算法步骤：

1. **Foreach** i in [1, I'] //所有的调度阶段
2. **Load** workload $\delta_c(t_i)$ between [t_i, t_{i+1}] for each VC c
3. **Foreach** $\theta(t_i)$ in U // 遍历每一个可行策略
4. **Foreach** VC c in [1,C] // 遍历每一个虚拟集群
5. **Simulate** $p_c(t_i)$ ***n*** times to get an average $\boldsymbol{P_c(t_i)}$
6. **Simulate** $d_c(t_i)$ ***n*** times to get an average $\boldsymbol{D_c(t_i)}$
7. $r_c(t_i) = \delta_c(t_i) \times P_c(t_i) / \theta_c(t_i)$ //当前虚拟集群剩余执行时间
8. **if** (i+1< I') //不是最后一个调度阶段
9. **Go to** step 9 until step 23 in Algorithm 5.1 //次优仿真所有
10. **Endfor**
11. **Endfor**
12. $\{\theta^{(1)}(t_i)\ldots\theta^{(s)}(t_i)\}$<– Select all in Pareto(thruput($\theta(t_i)$), $\sum_{c=1}^{C} D_c(t_i)$)
13. **Go to** step 2 in Algorithm 5.1 until step 24 replacing U with $\{\theta^{(1)}(t_i)\ldots\theta^{(s)}(t_i)\}$
14. $\{\theta^*(t_i)\}$<– Select one in Pareto(thruput($\theta^{(s)}(t_i)$), $\sum_{c=1}^{C} D_c(t_i)$)

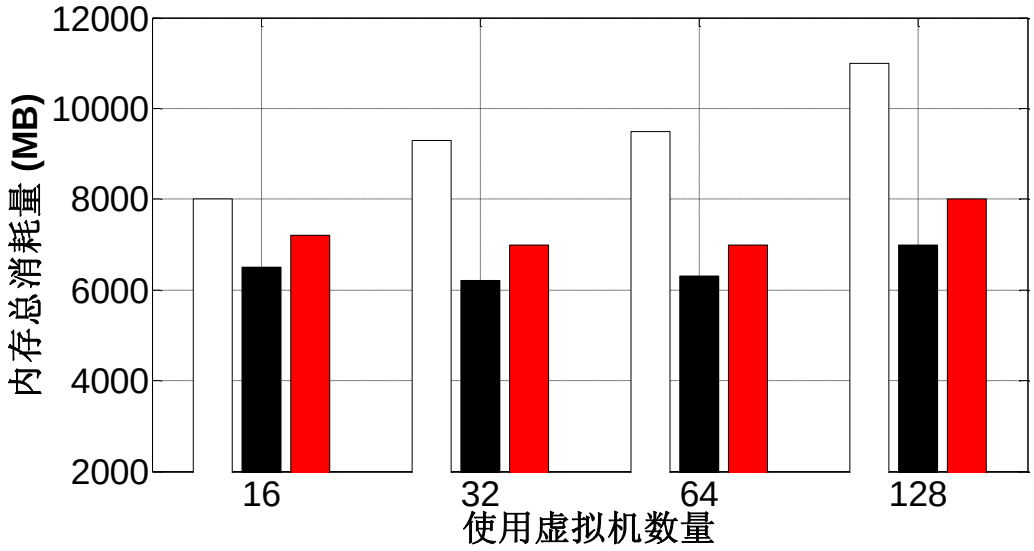
15. **Go to** Step 1 for the next loop
16. **Endfor**

5.5　实验结果分析

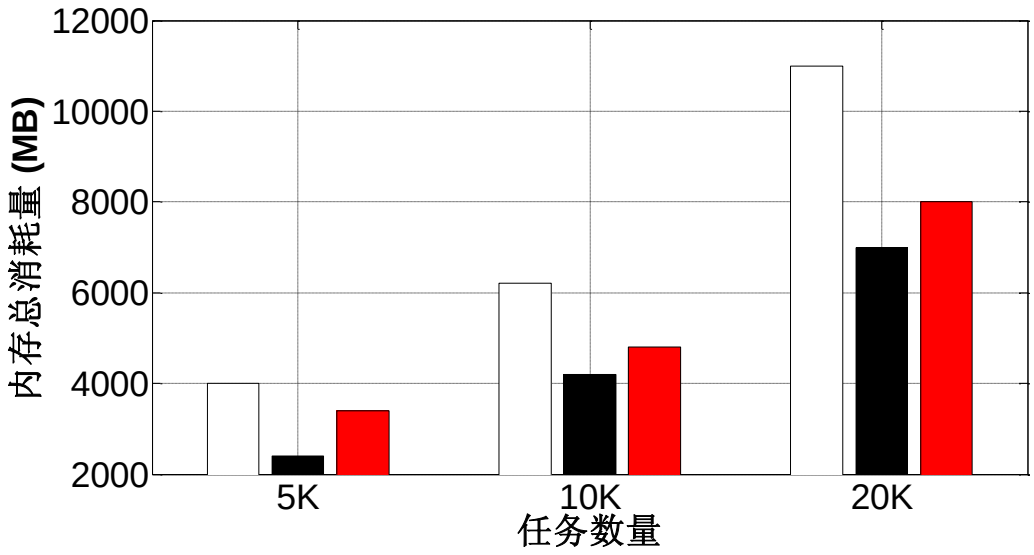
上一章的实验结果主要论述了低开销仿真优化方法在调度开销，即调度时间上的优势。本章的实验则在将其迭代的应用在仿真优化过程中，从吞吐率和响应时间两个指标考察其调度优势。



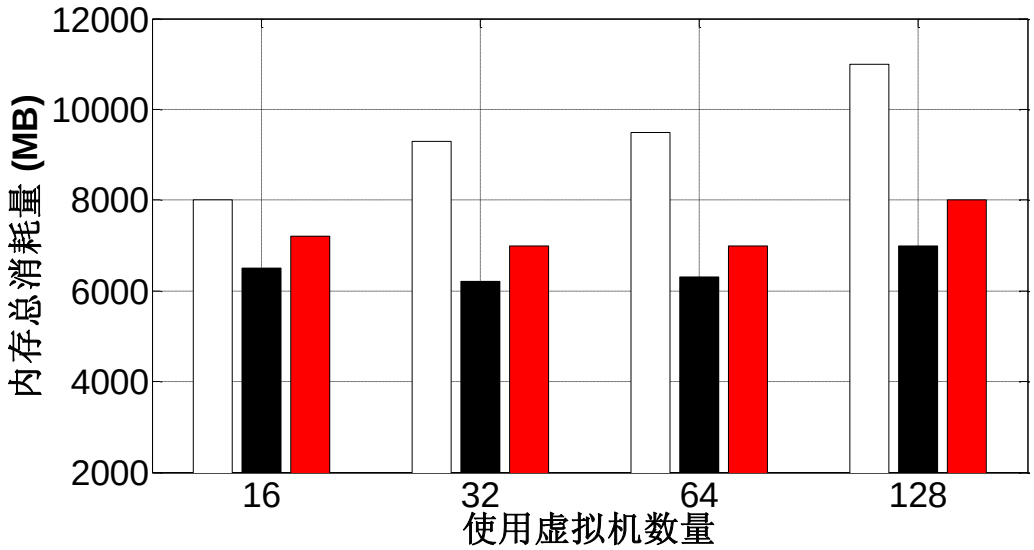
(a)三种方法分别在128个虚拟机实例，不同任务数量下吞吐率



(b) 三种方法分别在20K任务数量，不同数量虚拟机实例下的吞吐率



(c) 三种方法分别在128个虚拟机实例，不同任务数量下内存消耗



(d) 三种方法分别在20K任务数量，不同数量虚拟机实例下的内存消耗

图5.2　三种方法综合性能比较分析

图 5.2（a）和 5.2（b）对于吞吐率进行了综合比较，图 5.2（a）在给定 128 个虚拟机的情况下，对于任务总数量从 5K、10K 和 20K 三部分实验进行比较。而图 5.2（b）是在给定 20K 的任务数量中，对于虚拟机总数量从 16、32、64、128 进行了比较，我们试图从不同的维度观察三种调度方法在

吞吐率性能指标的量度下的可扩展性情况。黑色柱状、白色柱状和红色柱状为别为 LOS 方法、蒙特卡洛方法和盲选方法获取的实验结果。实验可以得出，LOS 方法在任务量较小，或者虚拟机数量不太大时，吞吐率性能提升的优势较小，而随着任务数量的加倍增长和虚拟机总数量的指数增长时，其吞吐率方面的优势体现得明显。

类似的结论可以在总内存消耗的实验结果中得以体现，见图 5.2（c）和 5.2（d）。不同的是，总内存消耗量随着总任务数量增长提升较快，而虚拟机数量的增长起伏并不太明显。其原因在于给定任务数量所需消耗内存的总量随调度策略的变化起伏并没有吞吐率那么明显。通过对实验结果的分析，总结起来可以得出如下结论：

（1）LOS 方法在负载变化比较剧烈，局部性差异比较大的多任务中，由于其低开销短时调度的特点，使得调度决策能更好适应工作负载变化；

（2）LOS 方法在陡峭的Ω类型和较小的噪声等级时表现性能良好，但是对于其他Ω类型，尤其噪声等级较大时，调度结果可能会导致开销较大，甚至性能比盲选法还要差，因为盲选法无粗糙评估阶段，综合开销更小。

5.6　本章小结

本章是前一章内容的延续和拓展。第四章我们介绍了低开销仿真优化方法的实现方式和原理，并用实验说明了其相对于蒙特卡洛仿真优化方法和盲选优化方法在降低仿真优化空间大小的主要不同，其优势主要体现在压缩调度空间到一个以上的数量级，从而缩短了调度时间开销。

本章我们首先理论上分析了低开销调度方法的开销调度优势，从而让前一章实验结果成为有根之木，有源之水。同时对低开销在最终决策性能上所能产生的优势进行了详细说明，即其低开销调度优势主要体现在迭代性的多阶段调度上，于是我们随之引入了迭代蒙特卡洛方法和迭代盲选法，并同时对比性的引入了迭代低开销调度方法的实现方法。最后我们从实验结果上分析了迭代低开销调度方法能够在调度性能上得到明显提升。

虽然本方法能够在一定程度上提升性能，但是其决策是完全基于控制调度时间的长短来进行的。换句话说，我们在决策过程中并未过多考虑影响决策的重要因素—工作负载的关联性。在下一章我们介绍低开销进化仿真优化方法，其通过引入工作负载相似性的概念来自适应的对其进行低开销和“高开销”的调度，从而再次整体提升调度性能。

第6章　弹性云平台资源低开销进化仿真优化供给

6.1　本章引论

在第三章我们对工作负载进行了分析，刻画了响应时间随其变化的数学模型，但是其不足有二。首先其并未考虑负载的随机到达性导致虚拟机的计算性能的运行期变化，这一问题在第四和第五章中引入仿真优化方法得以考虑并解决；其二本模型并未对相邻调度阶段的负载进行关联刻画，使得每阶段均完全独立和隔离，彼此之间无信息共享，后面调度阶段无法利用前面阶段的调度结果进一步提升调度性能。

第五章详细介绍了低开销迭代调度方法的优势以及分析了其对于系统性能提升的贡献。然而，该方法在负载变化不太大的情况下，其低开销调度优势无从体现，相反由于追求其更短调度区间的低开销性而故意引入的噪声的本质则会使得调度劣势体现出来。

针对上述问题，本章介绍弹性云平台资源低开销进化仿真优化供给方法，其优点在于在连续调度的各个阶段分析负载的变化规律，从而自适应的实现仿真时间长短的自由灵活选择。在负载变化剧烈的阶段，本方法自适应的切割第五章低开销迭代仿真优化的所产生的调度区间，从而对于突变负载进行更细粒度分析与调度，使其更好的适应负载的剧烈变化；而对于负载变化比较微弱的阶段，本方法自适应的将调度区间进行合并，从而为后阶段换取更多的仿真时间以克服噪声并全面提升调度性能。

本章的组织结构如下：6.2 节介绍了本章使用的多任务负载并对其相似度进行定义，同时讲述了如何根据相邻负载的相似性对调度区间进行合理的分割和合并；6.3 节详细介绍了切割和合并调度区间的算法，以及低开销进化仿真优化方法的实现思路；在 6.4 节中，我们给出了在广泛使用的亚马逊公司的弹性云计算平台 Amazon EC2 上对两个实验测试场景的设置，并获取了实验结果验证了低开销进化仿真优化方法对于调度性能的提升，第 6.5 节总结了本章的全部工作。

6.2　多任务负载类型

6.2.1　多任务负载简述

在弹性云计算平台中，多任务负载（Multitasking Workload）是随着时间不断变化的，对于不同的多任务负载类型提供不同的仿真时间，从而自适应的调整优化调度开销是必不可少的。对于弹性云计算平台虚拟资源的动态管理问题，情况更加复杂。其复杂性在于调度之前需将构建在物理集群上的虚拟资源按需切割成若干个虚拟集群，而每个虚拟集群均有各自多任务负载的变化规律，所以低开销调度的关键问题是如何兼顾各虚拟集群之间多任务负载的动态关系从而综合提升系统性能。我们首先用图 6.1 来对多任务负载变化进行分析。

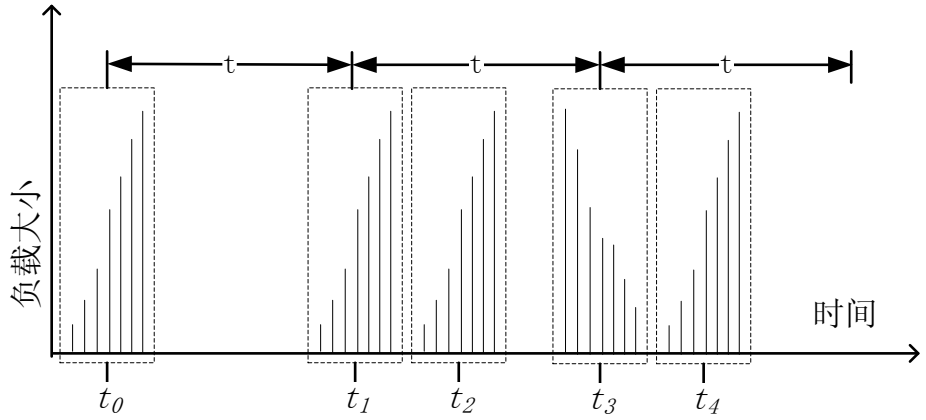


图6.1 多任务负载变化

上图给出了多任务负载变化的一种情况。在演示时间轴上共计五批任务陆续到达，每批任务中包含在虚线矩形框中的七条直线代表每一个虚拟集群中任务到达的数量，我们沿用前面两章的七个虚拟集群作为案例进行分析。多任务负载的变化程度则体现在相邻两批之间的相似程度。例如在 t_0，t_1 和 t_2 三个时刻到达的多任务负载在分布上很相似，而 t_2 和 t_3 则在分布上具有很大的差异。

分析多任务负载的差异性的主要目的是为了更好的降低调度决策开销。对于相似的多任务负载，可以使用相近甚至同一决策策略来进行调度，而对于差异较大的多任务负载，则需要将决策区间进行自适应按需切割，从而独立分析以获得好的决策策略。

6.2.2　多任务负载相似度分析

多任务负载相似度用于定义两批任务之间的相似性。假设在 t'时刻和 t''时刻到达了两批任务，分别是 $\delta(t') = [\delta_1(t'),\ldots,\delta_c(t'),\ldots,\delta_C(t')]$ 和 $\delta(t') = [\delta_1(t''),\ldots,\delta_c(t''),\ldots,\delta_C(t'')]$，其中 C 的定义和第四章一致，代表虚拟集群的数量，c 代表虚拟集群的下标。

$$\mathrm{Sim}(\delta(t'),\delta(t''))=\frac{\sum_{c=1}^{C}\delta_c(t')*\delta_c(t'')}{\sqrt{\sum_{c=1}^{C}\delta_c(t')^2*\sum_{c=1}^{C}\delta_c(t')^2}}$$

事实上，上述定义可以理解为两批多任务负载的强度相关性。负载强度均为正数取值，所以上述相似性范围为[0,1]。相似度为 1 时两批多任务负载分布完全一样，而相似度为 0 时表现在坐标系上为负载具有正交性，其实质是$\delta_c(t')$和$\delta_c(t'')$中总有至少一者为 0。

定义了相关性以后我们需要刻画相似度阈值，即在相邻多任务负载到达何种程度以后自动进行合并，或者单一调度区间多任务负载相似性差异到达何种程度以后自动进行区间分割的问题。我们采用的分割合并策略在随后分成三小节做分别介绍：

（1）调度区间分割策略

将同一个调度区间（指通过低开销迭代仿真优化方法已经形成的区间）中所有批次的多任务负载进行两两比较，如果差异最小的两者大于给定阈值 α，则认为本调度区间的多任务负载具有显著差异性，需要将调度区间进行分割；然后递归的利用低开销仿真优化方法分别分析两个被分割的子区间，直到区间宽度小于或等于给定的阈值 w 为止。此时认为如果再度划分调度区间则引入的噪声将会大于低开销本身对于性能的提升优势，从而对调度无益。

分割策略的前提是所采用的多任务负载均匀的分布在调度区间中，本方法取中作为分割点，可以保证每次分割以后差异最大的两批多任务负载以极大的概率分布在分割线的两端，从而实现分开调度。下图 6.2 展示了此调度区间分割策略。

由于 t_1 时刻和 t_2 时刻多任务负载极其相似而和 t_3 具有较大差异，基于此我们将调度区间平均划分为两个独立的调度阶段分开处理从而针对相似性较强的负载利用同一调度策略，相似性较差的负载采用不同调度策略。但是原本 t 的仿真时间缩短一半从而引入了噪声，这里就是存在如何折衷噪声和压缩调度阶段的问题。

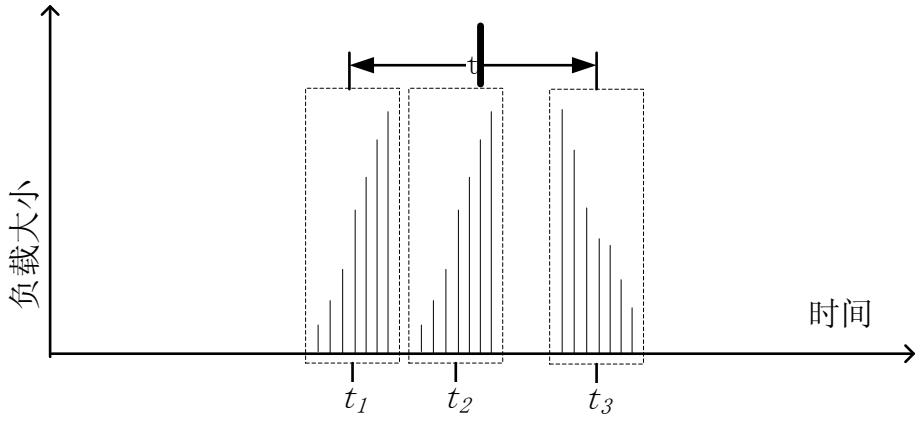


图6.2 调度区间分割图示

在概率论中，经典的大数定理说明了仿真时间提高两个数量级才能导致仿真精度提升一个数量级的结论。在虚拟集群的多任务负载剧烈变化时，如仿真精度到 10%以下（7 次切割区间）则基本退化为盲选，基于此我们启发式的选择最多三次对于低开销迭代仿真优化区间进行切割，从而确保实验对于多任务负载强度的把握控制在一定精度以内。

（2）调度区间合并策略

调度区间合并策略的原则基于（1）中已经分割好了的调度区间，或者说已经验证过的所有调度区间中多任务负载两两之间具有阈值高于α的相似性，即不可再分性。合并策略实施步骤如下：

将当前区间中的每一个负载和其相邻后一调度区间中的每一个负载进行两两比较，如果差异最大的两者小于给定阈值β，则认为本相邻调度区间的所有多任务负载的到达强度具有显著相似性，可以将以上两个调度区间进行合并；然后将合并以后的调度区间作为当前区间，采用同样的方法分析其与相邻后续调度区间多任务负载的差异性，直到区间之间具有显著差异性或者应用程序停止时结束。

下页图 6.3 展示了此调度区间合并策略。在图中可以看出，从 t_1 到 t_3 两个不同的调度阶段，t_1 和 t_2 阶段的多任务负载极其相似，而在两个调度区间均无其他类型的多任务负载到达，根据算法描述可以将其进行合并成一个独立的调度区间。而再考虑后一个调度区间从 t_3 开始时，由于新的多任务负载和前阶段的具有很大的差异性，故本轮合并阶段结束，从 t_3 开始进行新的一轮分析。

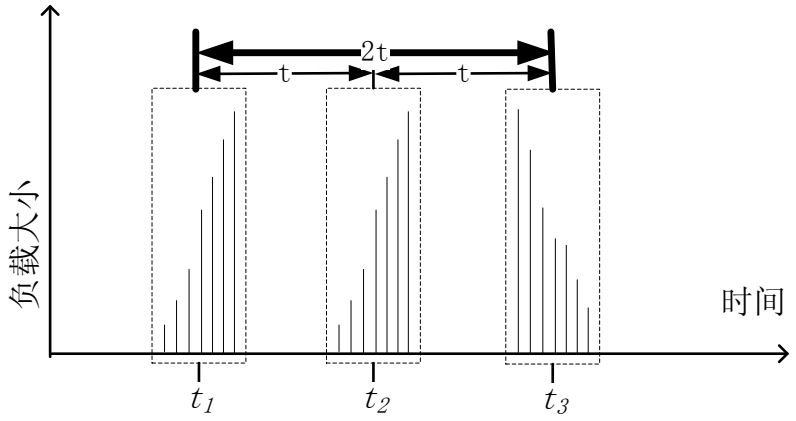


图6.3 调度区间合并图示

由于具有相似多任务负载在最佳调度策略上的相似性，合并了调度区间以后可以采用一次仿真的统一调度策略。其主要优势在于能够为后阶段的仿真争取到更多的仿真时间，可以提升仿真性能取得噪声更小的调度策略。本策略的产生弥补了低开销迭代仿真优化相对于迭代蒙特卡洛仿真方法的空白，使其能够自适应调度多任务负载变化比较平缓的场景。

基于以上分析，我们可以认为低开销进化仿真优化方法的实质是自适应的将迭代蒙特卡洛仿真方法和低开销迭代仿真优化方法进行整合，并将其合理配置到各自适应的多任务负载变化阶段，从而综合提升系统整体的调度性能。

（3）阈值设定分析

首先我们分析研究调度区间分割算法中的阈值。在分析了后续实验多任务负载的到达强度后我们做如下合理假设：首先，多任务负载的到达具有成批性，即不存在时间轴上零散分布于各个虚拟集群多任务负载到达的情况；其次，多任务负载到达的非空性，即每批到达的多任务负载在各个虚拟集群中均具有一定的强度。

根据以上假设，我们调查了现有多任务负载的强度，并根据分布随机生成了一万对组合数据以后，计算各种情况的相似性并得出统计结果如图 6.4 所示。

经过以上分析，我们利用启发式规则，统计出多任务负载相似性的中位数的下 33%分位数即为我们设定区间分割的阈值α，而中位数的上 33%分位数即为我们设定区间合并的阈值β。不同的应用场景下α和β的取值不尽相同，所以需要根据具体问题类型在实验之前进行设计。

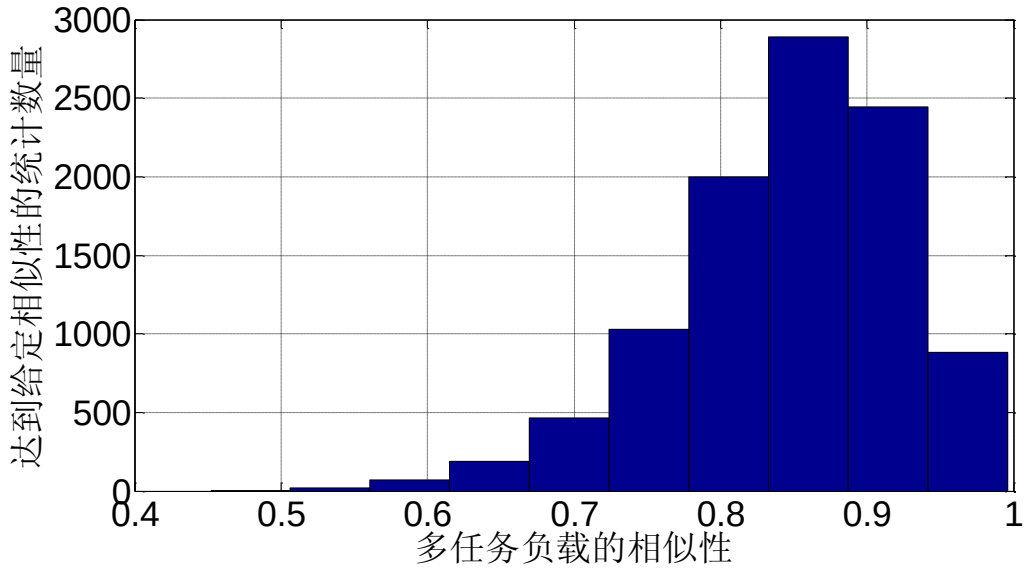


图6.4 多任务负载相似性的统计性分布

6.3　进化仿真优化方法

6.3.1　进化仿真优化方法算法分析

基于前一节的分析，首先我们用算法 5.1 和算法 5.2 来实现调度区间分割和合并的实现方式，然后我们在算法 5.3 中利用其给出了低开销进化仿真优化的实施步骤。

算法 5.1：多任务负载的分割算法 Partition(t_i, t_{i+1})

输入：位于 t_i 和 t_{i+1} 之间的所有多任务负载

（假设$\delta(t_l)$为在 t_l 时刻产生的多任务负载，其中 $t_i \le t_l < t_{i+1}$）

输出：位于 t_i 和 t_{i+1} 之间决策点

算法步骤：

1. DecisionList ← Null;
2. **If** (t_{i+1} - t_i) < **w**
3. **Add** t_i into DecisionList
4. **Return**　　//决策区间无法再度划分，取当前起点为决策点并返回
5. **Foreach** workload $\delta(t_l)$ in [t_i, t_{i+1}]
6. **Foreach** workload $\delta(t_k)$!= $\delta(t_l)$ in [t_i, t_{i+1}] && $k > l$
7. **if**(Max(Sim($\delta(t_l)$, $\delta(t_k)$)) < α)　　//最大相似度小于给定阈值
8. **Partition**(t_i, t_i + (t_{i+1}- t_i)/2);　　//递归计算分割的前半阶段决策点

9. **Partition**(t_i + (t_{i+1}- t_i)/2, t_{i+1});　//递归计算分割的后半阶段决策点
10. **EndIf**
11. **EndFor**
12. **EndFor**
13. **Sort** and **Return** DecisionList　//按时间排序返回决策点

算法 5.2：多任务负载的合并算法 Merge(*DP*)

输入：当前决策点时刻集合 *DP*

输出：合并算法执行后的决策点 *DPM*

算法步骤：

1. Foreach *dp* in *DP*
2. **Find** *dps* associated with *dp*　//寻找和 *dp* 对应的决策阶段 *dps*
3. **If (***dp* is not the last stage**)**　//最后一个阶段无法进行区间合并
4. **Find** *dpn* after *dp*　//*dpn* 是紧接着 *dp* 的后一个决策点
5. **Find** *dpns* associated with *dpn*　//寻找和 *dpn* 对应的决策阶段 *dpns*
6. **Foreach** workload $\delta(t_i)$ in *ds*
7. **Foreach** workload $\delta(t_k)$ in *dpns*
8. **If**(Min(Sim($\delta(t_i)$, $\delta(t_j)$)) > β)　//最小相似度大于给定阈值β
9. **Remove** *dpn* from *DP*;　//删除 *dpn*，即合并 *dps* 和 *dpsn* 合并
10. **Associate** *dp* with (*dps*+*dpns*)　//将 *dp* 和合并后区间关联
11. **Goto** Step 4
12. **Else Goto** Step 1
13. **EndIf**
14. **EndFor**
15. **EndFor**
16. **EndIf**
17. **EndFor**
18. **Return (***DPM*←*DP***)**

算法 5.3：低开销进化仿真优化方法 eLOS

输入：$\Delta = \{\delta(t)=[\delta_1(t), \delta_2(t),\ldots, \delta_c(t),\ldots \delta_C(t)]\}$

输出：$U = \{\theta^*(t_i)=[\theta^*_1(t_i), \theta^*_2(t_i),\ldots, \theta^*_c(t_i),\ldots \theta^*_C(t_i)]\}$

算法步骤：

1. **Foreach** i in [1, I']
2. DecisionList ← Partition(t_i, t_{i+1})
3. **EndFor**
4. DecisionList ← Merge(DecisionList)
5. **for each** i in [1, I'] in DecisionList　　//决策列表所有的调度阶段
6. **Load** workload $\delta_c(t_i)$ between [t_i, t_{i+1}] for each VC c
7. **for each** $\theta(t_i)$ in U　　// 遍历每一个可行策略
8. **for each** VC c in [1,C]　　// 遍历每一个虚拟集群
9. **Simulate** $p_c(t_i)$ $\underline{\boldsymbol{n}}$ times to get an average $\boldsymbol{P_c(t_i)}$
10. $r_c(t_i) = \delta_c(t_i) \times P_c(t_i) / \theta_c(t_i)$　　//当前虚拟集群剩余执行时间
11. **if** (i+1< I')　　//不是最后一个调度阶段
12. **if** ($r_c(t_i) > (t_{i+1}-t_i)$)　　//部分任务未完成遗留到下一调度阶段
13. FinishedTask[c] = $(t_{i+1}-t_i) \times P_c(t_i)$
14. RemainedTask[c] = $\delta_c(t_i)$ - $(t_{i+1}-t_i) \times P_c(t_i)$
15. FinishedTime[c] = $(t_{i+1}-t_i)$
16. $\delta_c(t_{i+1})$ += RemainedTask[c]
17. **else**　　//所有当前阶段任务完成
18. FinishedTask[c] = $r_c(t_i) \times P_c(t_i)$
19. RemainedTask[c] = $\delta_c(t_i)$ - $r_c(t_i) \times P_c(t_i)$
20. FinishedTime[c] = $r_c(t_i)$
21. $\delta_c(t_{i+1})$ += RemainedTask[c]
22. **Endif**
23. **else**　　//最后一个仿真阶段
24. FinishedTask[c] = $r_c(t_i) \times P_c(t_i)$
25. FinishedTime[c] = $r_c(t_i)$
26. Makespan[i] = **max**(FinishedTime[1:C])
27. **Endif**
28. thruput($\theta(t_i)$) = sum(FinishedTask[c])/Makespan[i]
29. **Endfor**
30. **Endfor**
31. $\theta^*(t_i)$ <––Select the largest thruput($\theta(t_i)$)
32. **Go to** Step 1 for the next loop

与 iLOS 方法的不同之处主要在于：和蒙特卡洛仿真方法类似，iLOS 对于每个调度阶段时间长度的选择完全依赖于所评价的两个阶段所仿真策略的数量来

决定，而 eLOS 的第 9 行采用 $\underline{\boldsymbol{n}}$ 代替 iLOS 的 $\boldsymbol{n}$，其数量大小是由于其对区间进行自适应切割和合并以后带来仿真次数响应的改变。

结合第五章对于各种调度方法时间开销的综合比较分析，此处可以得到估计数值为 $\underline{\boldsymbol{n}} \approx (t_i - t_{i-1})/o$，即后一阶段仿真次数取决于前一阶段仿真时间长度，在第一仿真阶段我们采用相对较大的仿真次数来获得。正是因为有了依赖于多任务负载特性的可变仿真时间长度，“进化仿真优化”因此而得名。

6.3.2　各种仿真优化方法的比较

在图 6.5 中，我们比较了三种仿真优化方法。横轴代表多任务负载类型，其在第 6.2.2 节中已经做过详细介绍。这里为了表达的方便，我们将相似性转化为两线条之间的高度差，即高度差小的多任务负载具有更高的相似性，此处同样分析 t_0 到 t_3 时刻的所有情况。

假设蒙特卡洛仿真优化的时间开销是 T = 4t，而利用迭代仿真优化方法则需要 t 的时间开销，也就是在蒙特卡洛仿真的一个调度周期，迭代仿真优化能够连续不间断执行 4 次调度。虽然较少的仿真时间导致每次调度从整体精度而言相比蒙特卡洛仿真优化差，但是其决策能够将更少批次多任务负载控制在一个调度区域，而非在一个更长的区间上根据多任务负载的平均到达进行调度，从而能够更加细粒度的执行决策。

而进化式仿真优化方法则充分学习多任务负载之间的相关性，将具有相似多任务负载的区间进行合并，比如在(t_0+2t, t_1)和(t_2, t_2+2t)将两个两阶段的调度过程合并以一次性决策策略优化，从而为后阶段仿真提供更长的时间和精度。而在区间(t_1+t, t_1+2t)中，由于多任务负载差异性较大，故自适应的将其分割为两个独立的调度阶段分别进行，以此分别获取各自调度策略。

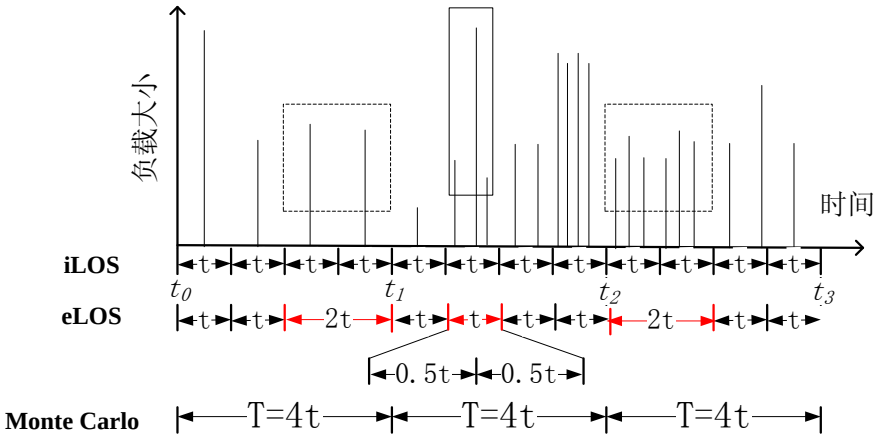


图6.5 三种仿真优化方法比较分析

显而易见，蒙特卡洛方法和迭代仿真优化方法此时位于调度优化频谱的两个端点处，即蒙特卡洛仿真利用较长的仿真时间获取足够好的决策，其高精度调度优势更容易体现在多任务负载变化不太剧烈，观测噪声（概念已经在第四章中定义）不太大的场景下，各个阶段具有较大相似性时具有最佳调度优势。而迭代仿真优化方法位于调度优化频谱的另外一个端点处，即利用较短的调度时间获取一个足够好而非最佳的决策，其低开销调度优势更容易体现在多任务负载比较剧烈，观测噪声不太大的场景下适用。进化仿真优化则综合了两种方法的调度优势，在多任务负载变化剧烈情况下自适应的切割仿真优化区间从而实现分别调度，而在多任务负载变化相对较弱时自适应合并仿真阶段从而实现集合调度，其调度性能整体而言会更佳。

在观测噪声较大的情况下，上述三种方法均具有较弱的调度性能，极端情况下则会演化为多阶段盲选法，其有别于迭代盲选法在于迭代盲选法具有 o_2 评价阶段（在第五章中已经介绍）而多阶段方法均无 o_2 阶段。

6.4　实验结果分析

本章的实验部分在 Amazon EC2 环境下实现。测试实验架构如下页图 6.6 所示。其中实验环境我们和前面两章中保持一致，设置了 7 个虚拟集群，每个虚拟集群由若干个虚拟机组成，其中有一个特殊的虚拟机命名为 Level-2 Seed Server，其作用是负责管理本虚拟集群中的设备，包括建立 HTTP 连接，分发应用程序计算任务，统计任务完成时间以及最终向最前端的 Level-1 Seed Server 进行汇报。Level-1 Seed Server 的主要任务是作为仿真服务器用于计算虚拟集群的分配策略，也就是执行进化低开销调度算法，将计算的分配策略分配给 Level-2 Seed Server。在运行结束时，统计每个 Level-2 Seed Server 完成的多任务负载任务的总数量，从而计算任务执行时间，吞吐量等指标。

除此以外，我们将本部分实验在两类场景下展开，具体见下页图 6.7。场景一为先仿真后执行方式，即每次调度前有充分的时间进行仿真从而计算出执行结果，然后产生与仿真阶段相对应的多任务负载获取系统性能；场景二为仿真和执行同时进行过程，即第 i 阶段仿真并产生第 i+1 阶段的调度。为保证各种方法的公平性，第零阶段调度策略通过在实验前以足够长的仿真时间来获取产生。

以上两种场景在 Amazon EC2 的实验环境中都很常见。场景一出现在 Spot Instance 的使用中，即用户有足够长的时间进行实验前仿真，获取好的结果，在 Spot Instance 的定价回归合理再执行；场景二出现在常见的实时调度场景中。

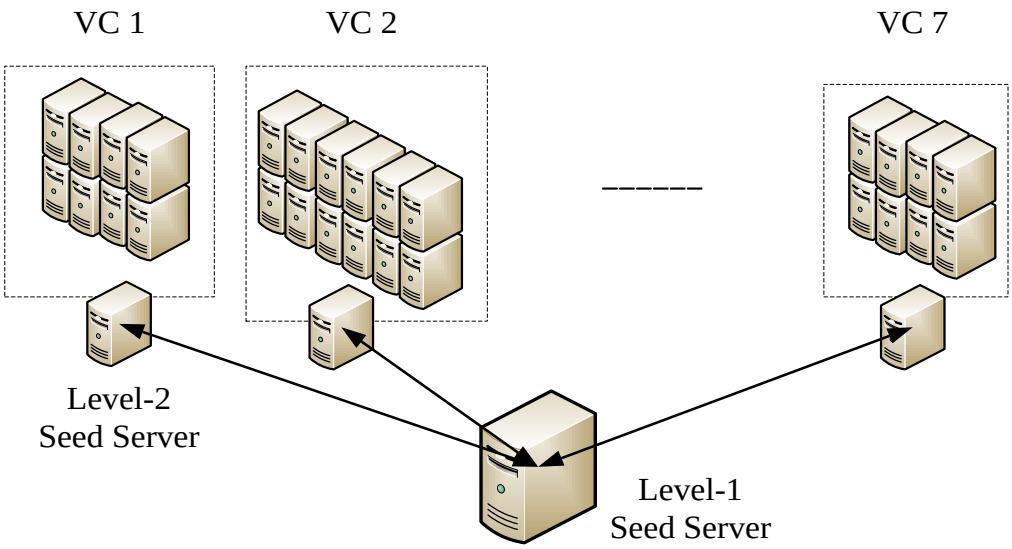


图6.6 测试实验架构

在以上两种场景下，我们比较了如下几种实验方法。在场景一中，由于不涉及到多任务负载的分析问题，所以我们比较了迭代蒙特卡洛仿真优化方法、迭代盲选法和低开销迭代仿真调度方法；在场景二中，我们除了采用了以上三种方法以外，增加了本章提出的低开销进化仿真优化方法。统一采用的性能指标是吞吐率，即我们测量出所有多任务负载的总执行时间以后，以其除总任务数量来获得。

除此以外，我们构建了实验所需要的四种测试用例来对多任务负载的强度进行分类，以期获得各种各样类型的多任务负载下四种方法的性能。这四种强度类别分别是低到达频率低差异负载（Low Frequency Low Covariance，简称 LFLC），低到达频率高差异负载（LFHC），高到达频率低差异负载（HFLC）和高到达频率高差异负载（HFHC）。测试的用例参见 1.5.2 节介绍的 7 类应用程序。

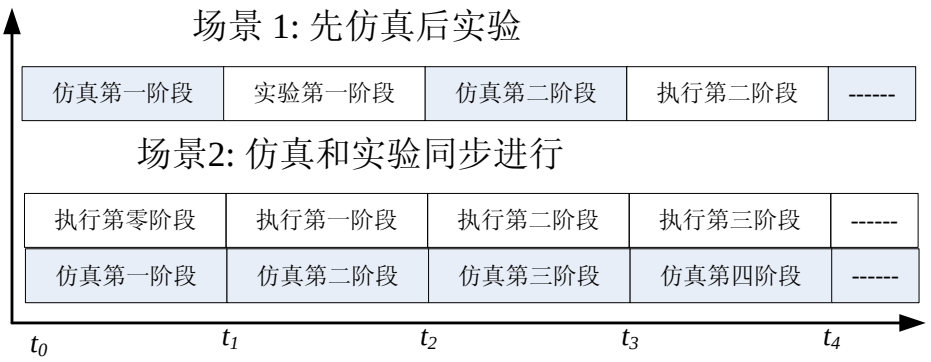
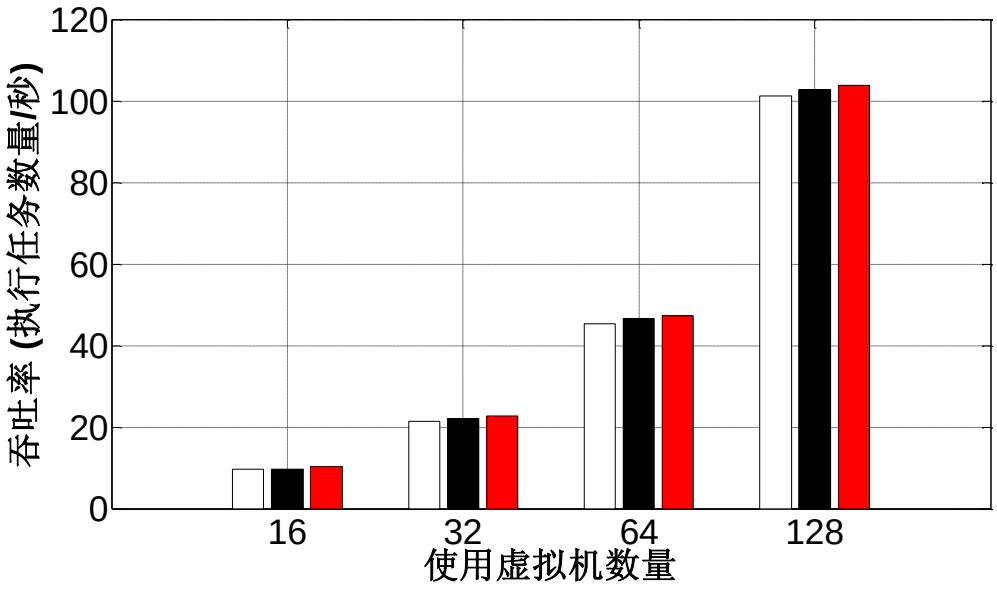
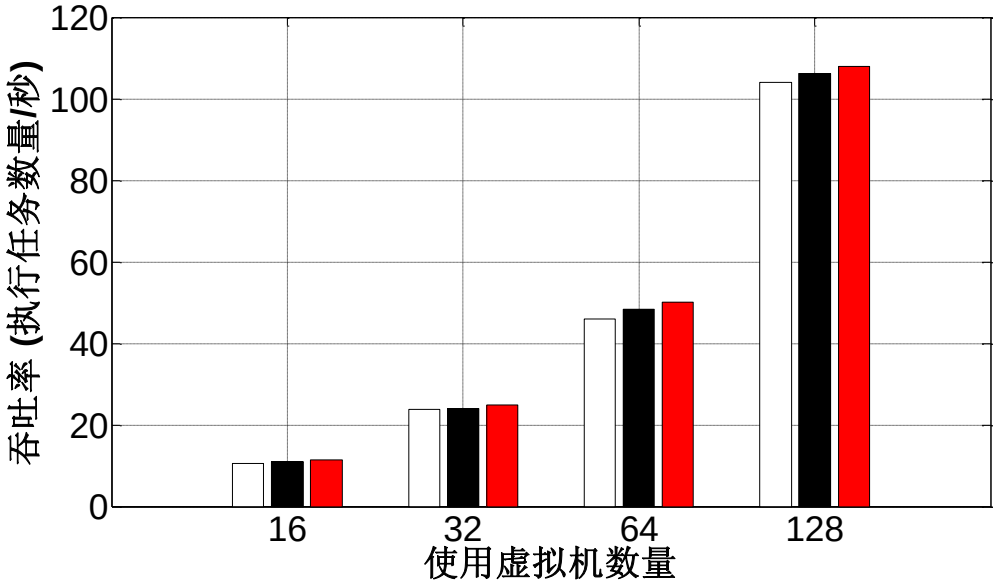


图6.7 两类测试实验场景

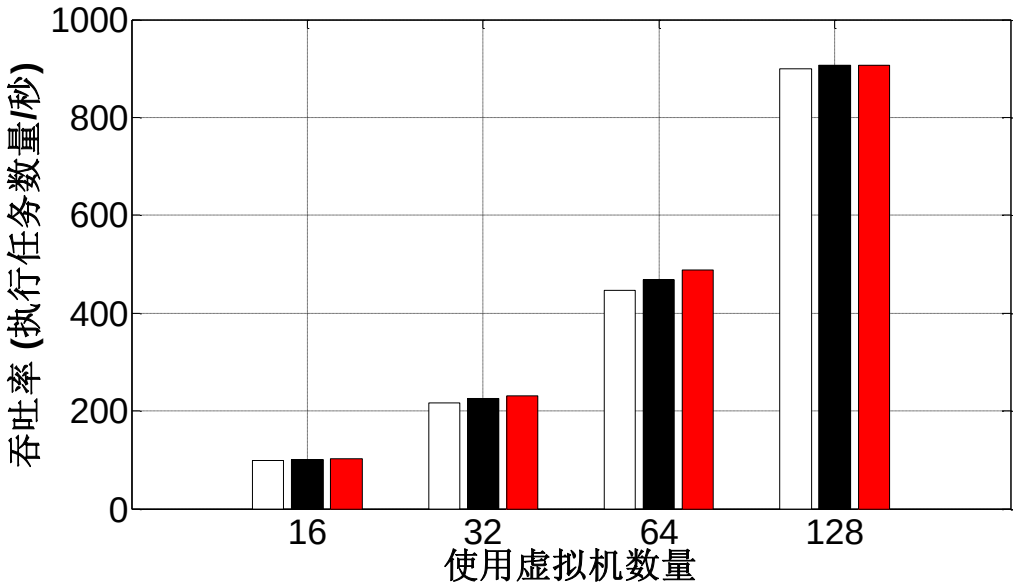
在图 6.8 中对场景一的试验结果进行了分析，在（a）（b）（c）（d）四种实验环境下，虽然 iLOS 的表现要略好于另外两种方法，但是实际上三种方法的吞吐率差异并不太大。这一点比较容易解释，因为各种方法都有充足的仿真时间，所以各种方法都能寻找到较优的虚拟资源调度策略分配给七类基准测试程序。相对而言各自能找到真实情况的比较好的策略。



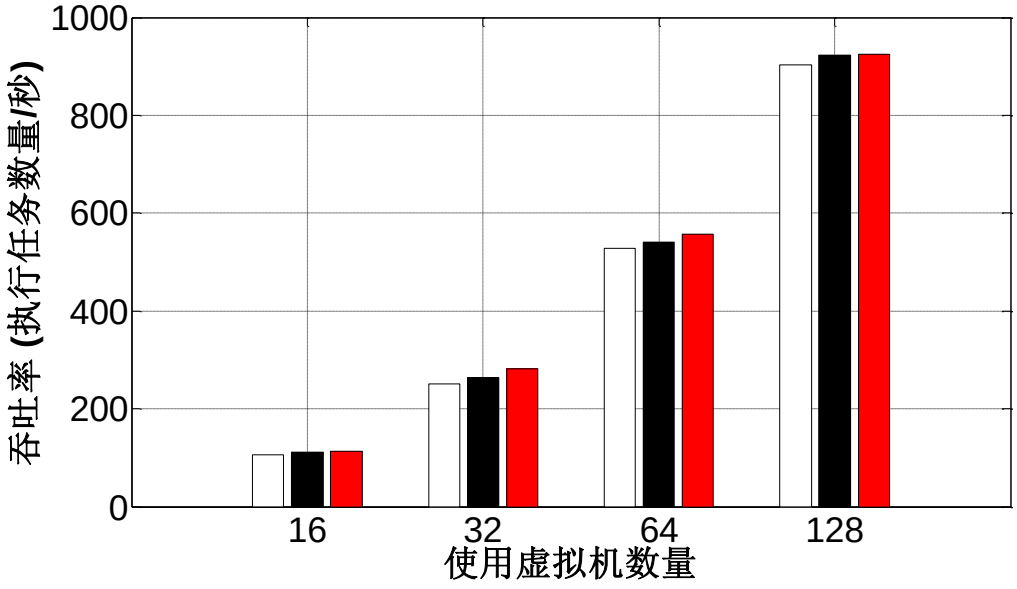
(a) 三种方法分别在场景一的LFLC多任务负载强度下的实验结果



(b) 三种方法分别在场景一的LFHC多任务负载强度下的实验结果



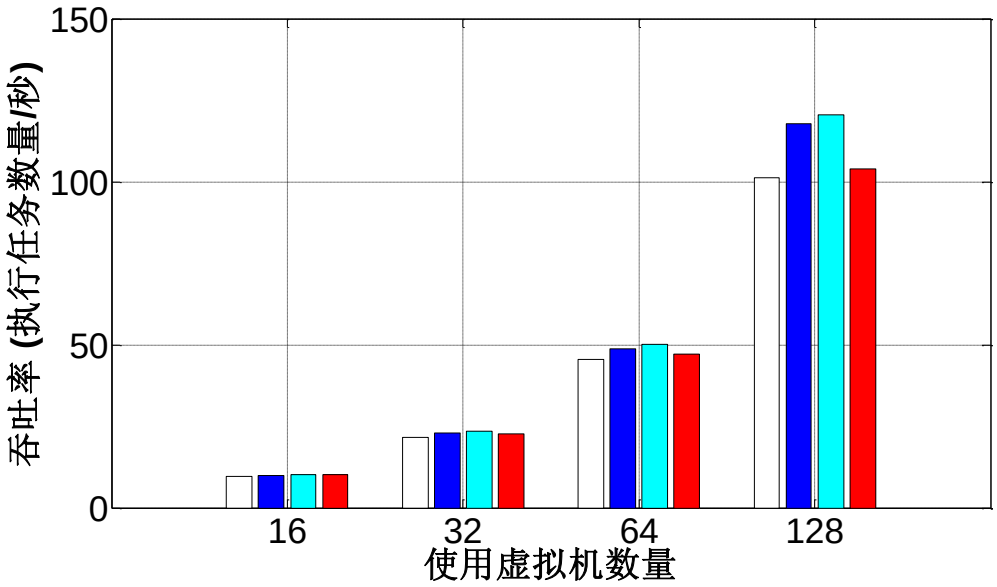
(c) 三种方法分别在场景一的HFLC多任务负载强度下的实验结果



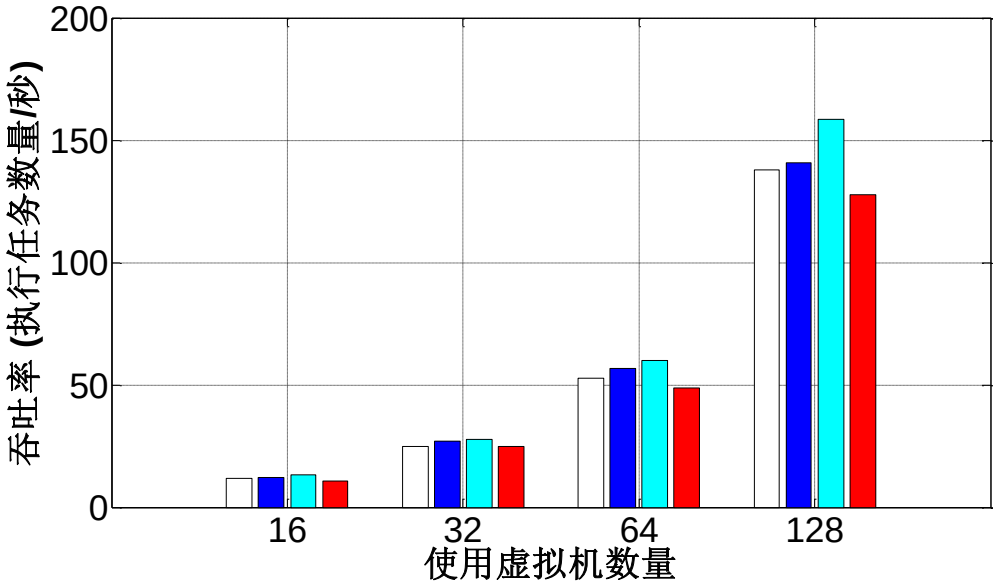
(d) 三种方法分别在场景一的HFHC多任务负载强度下的实验结果

图 6.8 各种方法分别在场景一的四种多任务负载强度下的实验结果

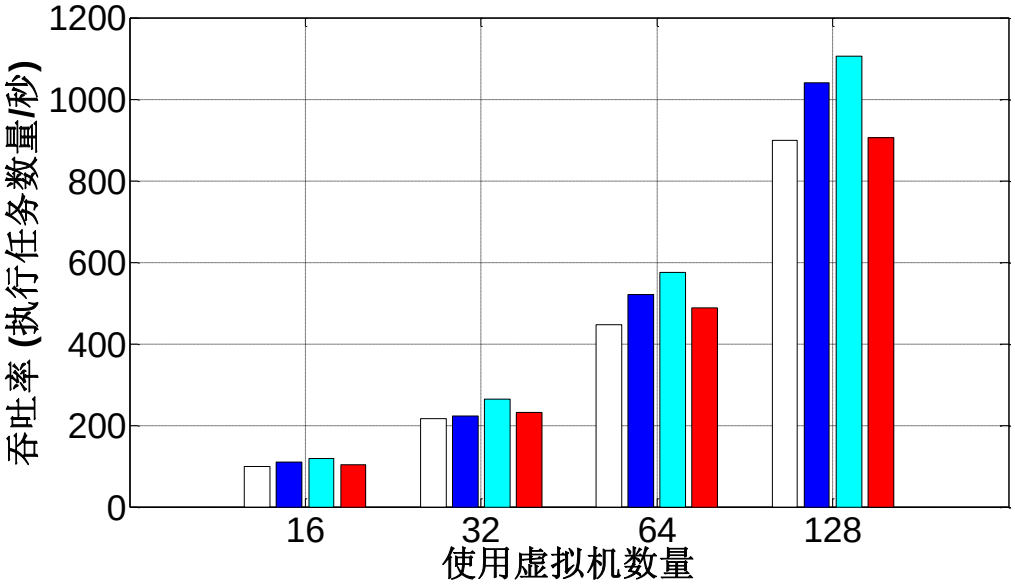
场景二的四种实验环境下的实验结果在图 6.9（a）（b）（c）（d）中呈现。和场景一一致，我们通过改变虚拟机数量来改变问题求解空间的大小从而增加获取最优策略的难度。



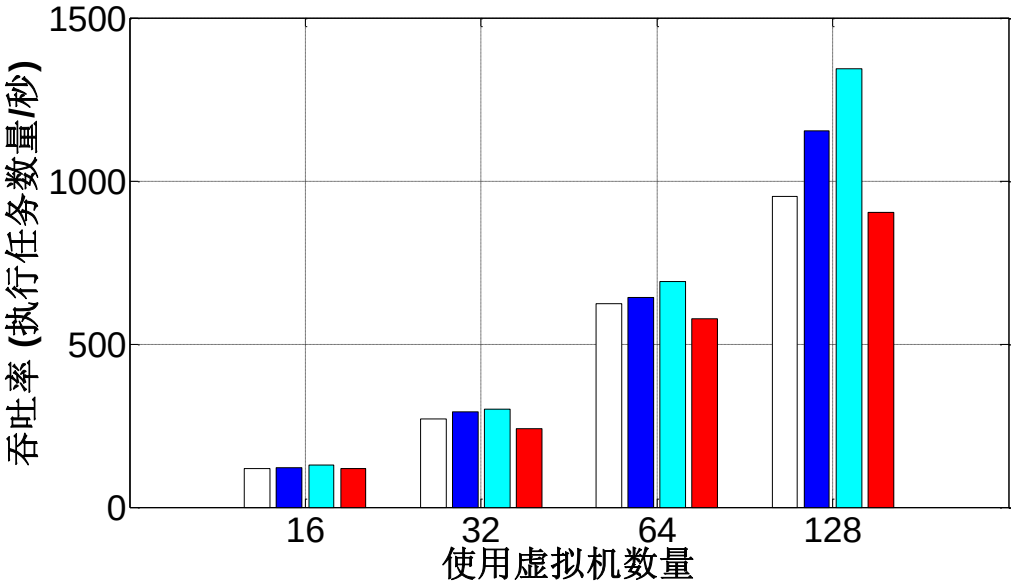
(a) 四种方法分别在场景二的 LFLC 多任务负载强度下的实验结果



(b) 四种方法分别在场景二的 LFHC 多任务负载强度下的实验结果



(c) 四种方法分别在场景二的 HFLC 多任务负载强度下的实验结果



(d) 四种方法分别在场景二的 HFHC 多任务负载强度下的实验结果

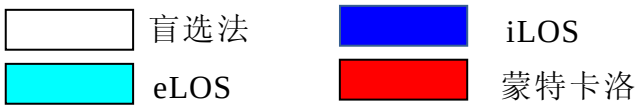


图 6.9 各种方法分别在场景二的四种多任务负载强度下的实验结果

通过实验结果，可以看出 iLOS 相比蒙特卡洛仿真和盲选法，以及 eLOS 相比于其他调度方法的优势就能得到明显体现。在多任务负载到达的频率较弱，虚拟资源数量较少的情况下，各种方法依然表现不出太大的差异性，其原因在于每

一个阶段都有极少的任务能完成，大部分到达的多任务负载都不断累积到最后一个阶段，所以整体而言在少量阶段上做粗粒度调度和多阶段做细粒度调度的性能差异性不大。但是当虚拟机数量到达一定程度以后，各个阶段的多任务负载都能及时得到执行，eLOS 通过学习多任务负载的动态变化规律对其进行自适应分析，以“高开销”和低开销相结合适应多任务负载动态变化，使其最大能提升 30%左右的调度性能。

6.5　本章小结

本章分析了低开销迭代仿真优化方法的性能提升方式，既通过对于调度区间内部和相邻调度区间之间多任务负载的相似性进行分析和比较来实现。低开销进化仿真优化方法通过分析多任务负载的相似性来自适应分割和合并调度阶段从而灵活的实现了在多任务负载变化平缓阶段以高开销调度来获取更大的仿真精度，在变化剧烈时以低开销调度方案来自适应切割仿真区间，引入适当的噪声从而获取调度性能的优化实现。

通过 Amazon EC2 上搭建的实验环境我们验证了在两种实验场景，四种多任务负载类型下，各种仿真优化方法的性能分析。我们先通过对于多任务负载进行仿真分析获取每种方法在各个阶段的最佳调度方案，然后通过施加相应的多任务负载强度，在真实基准测试程序环境下对于仿真所获得的调度方案进行测试，验证了 eLOS 方法相比其他方法的调度优势。

第7章　弹性云平台多指标综合评价模型

7.1　本章引论

前述章节从构建在大规模数据中心虚拟集群的低开销进化式仿真优化方法的调度方面探讨了如何规划和提升弹性云计算平台的调度性能，本章首先以弹性云计算平台后台数据中心的综合评价体系的有效构建中这个具体应用为背景展开，然后将其建模为多指标仿真优化的一般性问题，利用低开销进化仿真优化的核心思想，探讨如何实现合理的指标约简以构建多指标的最佳评价子集模型。

在目前有关数据中心评价指标的相关工作中，主要的难点有二：首先现有对于数据中心合理并公认的综合评价体系并没有完整的构建起来；其二，已有评价指标多且杂，而且对于每个指标的评价耗时耗力。在毫无先验知识的前提下，如何利用仿真优化方法选择一个较小的评价指标集合，其中包含最核心合理的评价指标的同时，对其进行评价所需要的仿真时间也能较小，是需要解决此类问题的主要难点。在本章的工作中，我们侧重于在全面收集各项评价指标的基础之上，利用低开销仿真优化的核心思想解决以上两个核心难题。同时本部分工作也是基于 OCBA 仿真优化方法在指标维度的拓展和应用。

本章借鉴第六章多任务负载切割和合并的主要思想，将低开销进化仿真优化思路进行重新建模，将其应用在数据中心多指标评价体系的进化式构建中。其核心思想是：类似于多任务负载相似性，将多目标评价体系中的具有正相关相似的指标进行合并，负相关相似的指标进行约简，从而构建出最具有代表系统性能特性的评价指标。

本章的组织结构如下：7.2 节建立了数据中心综合性能优化模型；7.3 节求解该模型的目标约简算法；在 7.4 节中，我们给出一个实例说明算法的可行性，7.5 节总结了本章的全部工作。

7.2　数据中心综合性能优化模型

7.2.1　数据中心综合性能评价指标体系

文献[122]研究了弹性云计算平台企业级数据中心性能评价的指标体系，见如表 7.1。

表 7.1 企业级数据中心性能评价指标体系

一级指标	二级指标	三级指标
建筑体评估	建筑选址	
	水资源综合利用	
	建筑节能方案	
	资源消耗	
	室内空气质量评价	
设备评估	基础物理设施	安全保护综合评价
		电器连接与布线综合评价
		空气质量与噪声综合评价
		电磁干扰综合评价
		温湿度与热分布综合评价
		电网质量与谐波综合评价
	IT 设备	性能指标综合评价
		仿真负载离线综合评价
		极限性能测试综合评价
		设备/系统综合评价
		温升/红外热成像
		能耗估计评价
	软环境	单线图审核与更新
		机房整体设计布局评估
		机房应急救援措施评估
		监控系统整体评估
		可维护性与可维护能力
		可维护方案与维护成本
设计评估	是否符合中国标准	主要指国内数据中心的构建、设计、验收等相关标准。
	是否符合可持续发展	主要是指是否考虑了数据中心的部署是否符合企业未来 5-10 年的短期，10~20 年的中期目标以及未来 20~30 年的长期发展规划。
	是否符合绿色设计原则	绿色设计的主要原则是通过对机房进行合理的布局和设计，在数据中心稳定按需运行的前提下，实现机房设备能效大和污染气体排放量小等实际问题。
业务评估	当前业务的整体满足程度	业务的整体可靠性
		业务的容错恢复性
		业务的评价响应时间
	未来业务的可扩展性	是否能满足访问量的急剧增长
		是否能满足数据量的急剧增长
	业务的操作难度	便于软、硬件人员操作业务
		便于软、硬件人员维护业务

该指标体系从三个层级，总共 33 个方面进行了较为全面的概括和总结，几乎所有的指标依然可以再次分解为更加细致的子指标，比如安全保护综合评价、空气质量与噪声综合评价、机房应急救援措施评估等。对于一个如此庞大的评估系统，其细粒度可量化的指标可能达到上百甚至上千。上述指标体系中的指标大部分存在一定的随机性，单次观测存有很大的误差，为了降低误差，往往需要多次观察后作处理，则总的计算量无法接受，本章致力于引入低开销进化仿真优化思想，将大量评价指标体系进行合理约简。

本章工作的目标是从所有可选指标中选择出一个最佳的指标子集，来取代评价上述所有指标造成的巨大时间开销，从而对不同数据中心服务商进行有效的综合比较，指导用户进行选择。由于更细粒度指标集的确定非常专业且暂未形成标准，本章工作以提出进化仿真方法和思路为主，并辅以实验验证大规模多目标评价进化约简方法的好处。

7.2.2　综合评价模型符号说明

首先我们在表 7.2 中列举出本章综合评价模型需要使用的符号并简要的介绍其意义，后面对其进行解释。

表 7.2 综合评价使用模型符号及其释义

符号	意义
S	可行策略集合
F	评价指标集合
X_{ik}^{j}	第i个策略，第k个评价指标，第j次测试值
μ_{ik}	随机变量 X_{ik}^{j} 的期望
σ_{ik}^{2}	随机变量 X_{ik}^{j} 的方差

- S：可行策略集合，总计包括 I 个可行策略，即$|S|=I$，$S=\{s_1,s_2,\ldots,s_I\}$；可行策略可以理解为不同的数据中心服务提供商；
- F：评价指标集合，总计包括 K 个评价指标，假设这里仅考虑最后一层指标集合，即$|F|=K$，$F=\{f_1, f_2,\ldots, f_K\}$；
- X_{ik}^{j}：第 i 个策略，第 k 个评价指标，第 j 次仿真测试值，X_{ik}^{j}是个随机变量；
- μ_{ik}：随机变量X_{ik}^{j}的期望，即第 i 个策略，第 k 个评价指标的真实值；

- σ_{ik}^2：随机变量X_{ik}^j的方差，即第 i 个策略，第 k 个评价指标的单次仿真方差；

7.2.3　综合评价模型基本假设

- 所有测试值均为极大型指标，即 X_{ik}^j 越大表示测试效果越好；
- 随机变量 X_{ik}^j 服从期望为 μ_{ik}，方差为 σ_{ik}^2 的正态分布，即 $X_{ik}^j \sim N(\mu_{ik}, \sigma_{ik}^2)$；
- 策略 i，评价指标 k 在不同的仿真测试 u 和 v 是独立的，即 $P(X_{ik}^u | X_{ik}^v) = P(X_{ik}^u)$；
- 不同的评价指标 s，t 之间可能存在相关性，即 $|\rho(\mu_s, \mu_t)| \geq 0$，其中 $\mu_s = (\mu_{1s}, \mu_{2s}, \cdots, \mu_{Is})^T$，$\mu_t = (\mu_{1t}, \mu_{2t}, \cdots, \mu_{It})^T$。

7.2.4　数据中心优化模型

数据中心优化的目标是从所有可行的方案集中，找出一个或者几个各项评估指标数值都比较小（最大化问题可以相应转化）的方案：

$$\max_{i=1,2,\cdots,I} M = (\mu_{i1}, \mu_{i2}, \cdots, \mu_{iK}) \tag{7.1}$$

上述模型是一个多目标规划模型，为了说明方法的可行性，本文就多目标评价函数进行合理的目标约简，以期降低计算复杂度。

7.3 多指标约简方法

多目标规划（7.1）式可以简化为：

$$\max_{i=1,2,\cdots,I} M' = \sum_{k=1}^{K} \omega_k \, \mu_{ik} \tag{7.2}$$

此时将多目标规划模型转化为单目标规划模型，问题复杂度大大降低。式中的加权系数 ω_k 满足 $\sum_{k=1}^{K} \omega_k = 1$，表示第 k 个评价指标在总体评价中的地位和作用，对于重要的指标应赋予大的权重。有关权重的确定方法可以参考 AHP[123] 中比较矩阵的方法，在此不再赘述。

虽然上述模型以是单目标规划模型，但是实际中μ_{ik}是事先不可知的，只能通过大量的仿真实验，取其平均值 $\overline{X_{ik}} = \frac{1}{b_i}\sum_{j=1}^{b_i} X_{ik}^j$ 近似 μ_{ik}。近似的程度随着仿真次数的增加而增大，$\overline{X_{ik}} \sim N(\mu_{ik}, \frac{\sigma_{ik}^2}{b_i})$。

若每个方案每个目标的仿真次数为 b，则总计需要仿真 $B=bIK$ 次。在数据中心优化问题中可行策略的数量 I 非常大，指标体系中的评价指标集的数量 K 也很

大，为了保证估计精度，策略 i 的评价指标 k 需要仿真的次数 b_i 也需要具有一定的规模，上述原因导致了总体仿真规模 B 将达到一个无法接受的程度。

多目标规划模型（7.2）式中的各个评价指标的权重ω_k不同，权重小的指标对综合评价的影响小，而权重大的指标对综合评价的影响大，为了提高精度，并降低仿真次数，就不能将计算资源均匀地分配给各个指标，对于影响大的指标应该配置更多的计算资源，而影响小的指标可以配置较少的计算资源。

此外，数据中心是一个复杂的系统，各个评价指标之间存在各种各样的依存关系，各个指标之间并不独立。指标之间的相关性导致一些指标的评价值中可能包含其他指标评价值的信息，此时再对这些指标进行评价就会导致计算资源的浪费。

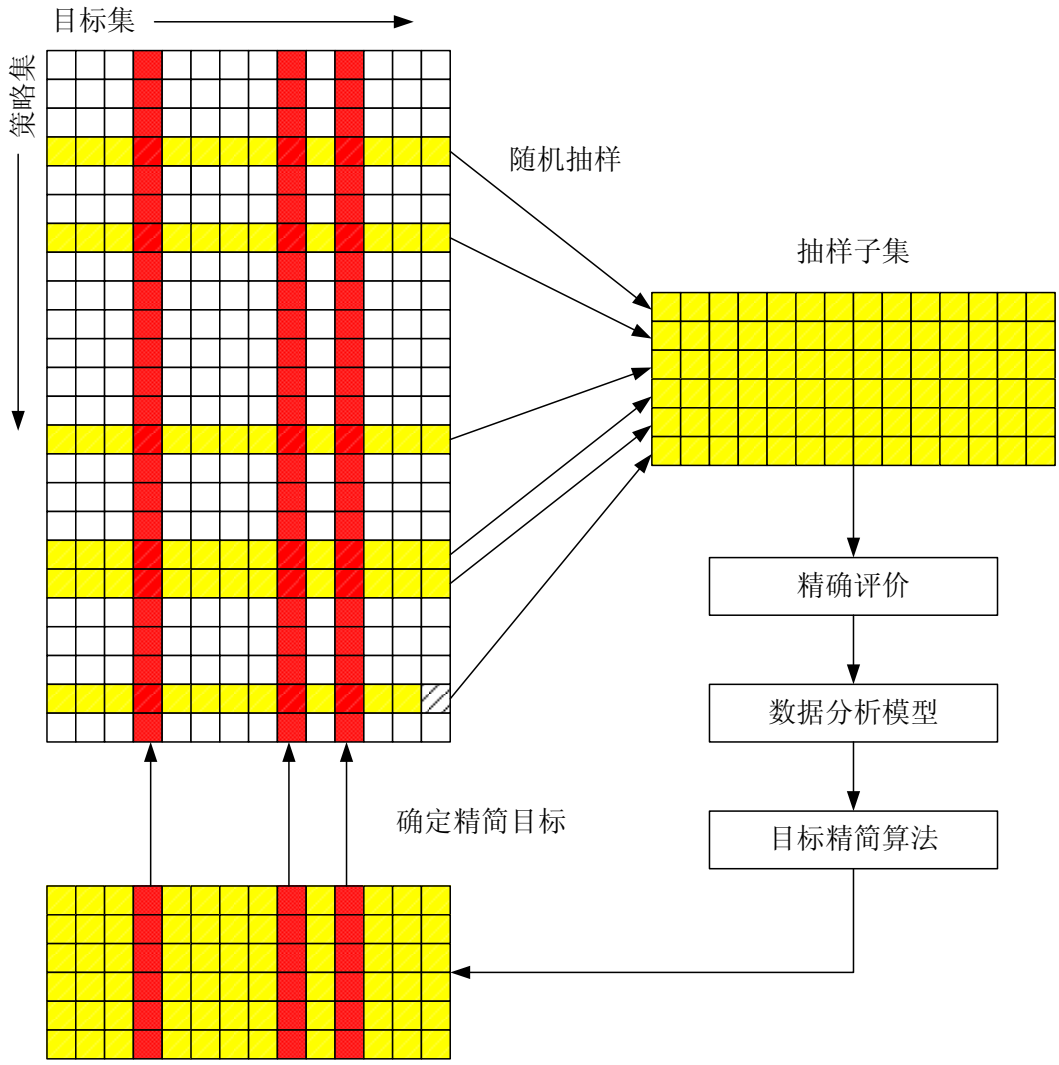


图 7.1　目标约简模型

要实现约简目标，就必须了解目标之间的相互关系。因此需要首先对策略空间进行抽样，并对抽样空间内的各个目标进行精确评价，通过研究抽样空间目标之间的关系，运用目标约简算法，降低评估目标数量，并推广到完整的策略空间中，用少量的目标去评价剩余的策略集，从而实现降低评价开销的目的，整体思路示意图如图 7.1。在 7.4 节中，我们将用实验进行验证。

在图 7.1，首先从策略集中随机抽取一定数量的策略，用以研究目标之间的相关性。随后对抽样策略集合中每个策略进行精确评价，分析观测值，并运用目标约简算法减少目标数量，确定约简后目标集，最后对策略集中未经评估的策略评估约简后的新目标集。从示意图中可以看出，运用目标约简算法，仅仅评估了图中阴影部分，图中白色方框部分是未评估的，降低了评估成本。

后面给出了多传感器信息融合模型和目标约简进化仿真优化模型求解方法。

7.3.1　多传感器信息融合模型

多传感器信息融合是指为完成决策和估计任务，对若干传感器的观测结果在一定准则下加以自动分析和处理过程。本模型中，将一个评估指标看成一个传感器，不同策略下导致的该指标的数值看成是传感器的观测信息。此处约定采用第 i 个传感器评价概念和采用第 i 个目标评价概念一致。

设传感器 A 能够带来的信息量为 $H(A)$，传感器 B 能够带来的信息量为 $H(B)$，由于传感器之间的相关性，两个传感器之间信息存在重复部分，记传感器 A，B 融合后产生的信息量为 $H(A,B)$，则有

$$I(A,B) = H(A,B) - [H(A) + H(B)] \geq 0$$

其中 $I(A,B)$表示传感器 A 与 B 之间的互信息。

若在传感器A的基础上，增加传感器B，此时增加的信息量为：

$$H(B|A) = H(A,B) - H(A) = H(B) - I(A,B)$$

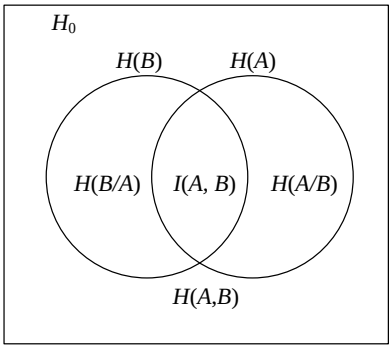


图 7.2 信息交叠示意图

图 7.2 中 H_0 表示真实评价所需要的信息量，初始时刻，传感器 A 加入，此时提供的信息量为 $H(A)$，之后传感器 B 加入，传感器 B 的加入并没有带来 $H(B)$ 的信息量，而是只增加了 $H(B|A)$ 的信息量。若 B 本身所具备的信息量越大，A，B 之间的互信息越小，则通过增加传感器 B 所产生的信息量就越大。

直观地看，一个传感器的信息量就等价与一个圆的面积，每增加一个传感器 K，都等价于在图中增加一个面积为 $H(K)$的圆，这个圆可能与已有的圆重复，若完全落在其他圆的交集里，则说明该传感器没有带来任何信息，完全可以由其他传感器表出。若传感器 K 的加入导致了所有圆的交集面积的增长，则说明 K 带来了信息量。

对于同一个融合模型，传感器的数量越多，融合带来的信息量就越大，但是传感器过多将带来新的问题。一方面，增加了探测成本，也增加了融合成本；另一方面，可能导致融合模型不稳定，尤其是传感器读数相关性较大时，可能出现传感器之间的相互干扰，影响融合模型的稳定性。

为了克服上述问题，一种可行的方法是从所有的传感器中按照一定的原则，选择部分传感器作为融合来源。这样做一方面可以有效降低探测和融合成本，另一方面可以提高融合模型的稳定性。

（1）单传感器信息量模型

下面研究评价指标 k 对于“策略 X_i 是最佳策略”这一结论提供的信息量，其中策略 X_i 是评价指标 k 认为的最优策略的编号。

针对评价指标 k，构建相应的传感器Xe_k，Xe_k针对各个策略的观测值为 $Y_k=(y_{k1},y_{k2},\cdots,y_{kI})$，$y_{ki}\sim N(\mu_k,\sigma_k)$，不妨设 $y_{k1}\geq y_{ki}\ i\neq 1$。假设这些策略的真实综合评价性能为 $Y=(y_1,y_2,\cdots,y_I)$，$y_i\sim N(\mu,\sigma)$，此处真实评价性能和第四章一致。由前面假设，策略 X_i 的综合性能是由各个传感器的读数加权得到的，因此各个传感器的读数与综合评价结果存在一定的线性相关性。

y_k是策略 X_i 的真实性能，现在用y_{ki}去估计 y_i 可能存在误差。以 Y 为应变量，Y_k 为自变量进行线性回归，建立起 Y 和 Y_k 的关系：

$$Y=Y_k\hat{\beta}+\varepsilon$$

$$Y_k=\begin{bmatrix}1 & y_{k1}\\ 1 & y_{k2}\\ \vdots & \vdots\\ 1 & y_{kI}\end{bmatrix}$$

采用线性最小二乘回归方程系数：

$$\hat{\beta} = \left(Y_k{}^T Y_k\right)^{-1} Y_k{}^T Y$$

$$\hat{Y}_k = Y_k\,\hat{\beta},\ \varepsilon = Y - Y_k\left(Y_k{}^T\,Y_k\right)^{-1} Y_k{}^T Y,\ S = \varepsilon\,\varepsilon^T\ ,\ \hat{\sigma}_k{}^2 = \frac{S}{I-1}$$

其中 $\hat{\beta} = \left(\beta_0, \beta_1\right)$ 是回归系数。

传感器 k 的观测值 Y_k 与真实性能 y 的联合分布为：

$$f(y_k, y) = f(y|y_k)f(y_k) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_k} e^{-\frac{(y-\beta_1 y_k-\beta_0)^2}{2\hat{\sigma}_k^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{(y_k-\mu_k)^2}{2\sigma_k^2}}$$

传感器 k 的修正评价值 $\hat{y}_k$ 和真实性能 y 的联合分布为：

$$f(\hat{y}_k, y) = \frac{1}{\sqrt{2\pi}\hat{\sigma}_k} e^{-\frac{(y-\hat{y}_k)^2}{2\hat{\sigma}_k^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma_k} e^{-\frac{((\hat{y}_k-\hat{\beta}_0)/\hat{\beta}_1-\mu_k)^2}{2\sigma_k^2}}$$

$$= \frac{1}{2\pi\hat{\sigma}_k\sigma_k\hat{\beta}_1} e^{-\frac{(y-\hat{y}_k)^2}{2\hat{\sigma}_k^2}+\frac{\left[(\hat{y}_k-(\hat{\beta}_0+\hat{\beta}_1\mu_k)\right]^2}{2\left(\sigma_k\hat{\beta}_1\right)^2}}$$

$f(\hat{y}_k, y)$关于 $\hat{y}_k$ 的边缘分布为：

$$f(\hat{y}_k) = \int_{-\infty}^{\infty} f(\hat{y}_k, y)\,dy = \frac{1}{\sqrt{2\pi}\hat{\beta}_1\sigma_k} e^{-\frac{\left[(\hat{y}_k-(\hat{\beta}_0+\hat{\beta}_1\mu_k)\right]^2}{2\left(\sigma_k\hat{\beta}_1\right)^2}}$$

对于任意的策略X_i，被传感器k评判为最优策略的概率为

$$P(\hat{y}_k \geq a) = \int_a^{\infty} f(\hat{y}_k)\,d\hat{y}_k = \frac{1}{I}$$

a为传感器k划定的最优策略阈值。设 b 满足

$$P(y \geq b) = \int_b^{\infty} f(y)\,dy = \frac{1}{I}$$

b为真实性能划定的最优策略阈值。

若策略 X_1被传感器k选为最优策略。记：事件 A 为传感器 k 认为策略 X_1 是最优策略，事件 B是策略X_1是真正的最优策略。则 $P(A) = P(\hat{y}_k \geq a)$，$P(B) = P(y \geq b)$，

$$P(A,B) = P(\hat{y}_k \geq a, y \geq b) = \int_a^{\infty}\int_b^{\infty} f(\hat{y}_k, y)\,d\,y d\hat{y}_k$$

根据条件概率公式：

$$P(B|A)=\frac{P(A,B)}{P(A)}$$

其中 $P(B|A)$ 表示若传感器 k认为策略 X_1 是最优策略，则策略 X_1是真正的最优策略的概率。基于此分析，我们可以获得在传感器 k 引入后所能新提供的信息量为：

$$H_k=H_0\ -\ [-log_2\ P(B|A)]=log_2\ I+log_2\ P(B|A)$$

上式的获取和实际意义在信息论和熵的有关内容中均有涉及，由于论文篇幅有限，这里不再详细解释。

定理 1 传感器 k 提供的信息量 H_k 关于$\hat{\sigma}_k$严格单调减。

证明： $f(\hat{y}_k,y)=\frac{1}{\sqrt{2\pi}\hat{\sigma}_k}e^{-\frac{(y-\hat{y}_k)^2}{2\hat{\sigma}_k^2}}\cdot\frac{1}{\sqrt{2\pi}\sigma_k}e^{-\frac{((\hat{y}_k-\hat{\beta}_0)/\hat{\beta}_1-\mu_k)^2}{2\sigma_k^2}}$

$$=e^{\frac{1}{\hat{\sigma}_k^2}}\frac{1}{\sqrt{2\pi}\hat{\sigma}_k}e^{-\frac{(y-\hat{y}_k)^2}{2}}\cdot\frac{1}{\sqrt{2\pi}\sigma_k}e^{-\frac{((\hat{y}_k-\hat{\beta}_0)/\hat{\beta}_1-\mu_k)^2}{2\sigma_k^2}}$$

当$\hat{\sigma}_k=1$时，

$$P(\hat{y}_k\geq a,y\geq b)=\frac{1}{2\pi\sigma_k}\int_a^\infty\int_b^\infty e^{-\left[\frac{(y-\hat{y}_k)^2}{2}+\frac{\left(\frac{\hat{y}_k-\hat{\beta}_0}{\hat{\beta}_1}-\mu_k\right)^2}{2\sigma_k^2}\right]}\mathrm{d}\,y\mathrm{d}\hat{y}_k=C>0$$

$$P(A,B)=\int_a^\infty\int_b^\infty f(\hat{y}_k,y)\,\mathrm{d}\,y\mathrm{d}\hat{y}_k=e^{\frac{1}{\hat{\sigma}_k^2}}C$$

$$P(B|A)=\frac{P(A,B)}{P(A)}=Ie^{\frac{1}{\hat{\sigma}_k^2}}C$$

$$H_k=\log_2 I+\log_2 P(B|A)=\log_2 I+Ie^{\frac{1}{\hat{\sigma}_k^2}}C$$

$$\frac{\partial H_k}{\partial\hat{\sigma}_k}=-2\hat{\sigma}_k Ie^{\frac{1}{\hat{\sigma}_k^2}}C<0$$

因此，传感器 k 提供的信息量H_k关于$\hat{\sigma}_k$严格单调减。

由定理 1 得到：传感器 k 提供的信息量与回归误差的大小呈单调减的关系，即误差越小，提供的信息量越大。若仅需要一个传感器去估计原问题时，应该选择回归误差最小的传感器。下面我们分析对于多传感器融合的优化模型。

（2）多传感器融合信息量模型

研究用 P 个传感器融合的情况，P 个传感器对应 P 个评估指标。设传感器 p 对策略 X_i 的读数为 x_{ki}，令

$$X=\begin{bmatrix}1 & x_{11} & x_{12} & \cdots & x_{1I}\\ 1 & x_{21} & x_{22} & \cdots & x_{2I}\\ \vdots & \vdots & \vdots & \ddots & \vdots\\ 1 & x_{P1} & x_{P2} & \cdots & x_{PI}\end{bmatrix}$$

假设融合模型为线性最小二乘回归融合，构造回归方程$Y=\hat{\beta}X+\varepsilon$，采用线性最小二乘回归方程系数：

$$\hat{\beta}=(X^TX)^{-1}X^TY$$

$$\hat{Y}=\hat{\beta}X,\ \varepsilon_i=Y_i-X(X^TX)^{-1}X^TY_i,\ S=\sum_{i=1}^{I}\varepsilon_i^2\ ,\ {\hat{\sigma}_p}^2=\frac{S}{I-P}$$

研究目标函数的真实值与估计值的联合分布 $P(\hat{y},y)$，设样本的综合评价真实值 y 的分布为$P(y)$，真实综合评价为Y时，观测值$\hat{y}$的条件概率$P(\hat{y}|y)\sim N(y,\hat{\sigma}_p)$，则$P(\hat{y},y)=P(\hat{y}|y)P(y)$。

通过融合，可以认为融合后构成了一个新的传感器Xe_v，如图 7.3。

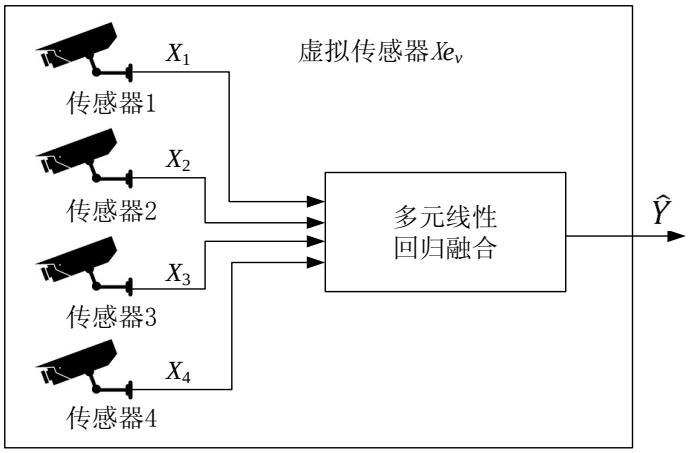


图 7.3 多传感器融合示意图

该传感器与真实值之间的联合概率分布为$P(\hat{y},y)$，类似单传感器模型，得到融合结果$\hat{Y}$的熵为：

$$H_V=log_2I+log_2P(C|V)$$

$$P(C|V)=\int_a^{\infty}\int_b^{\infty}\frac{1}{2\pi\hat{\sigma}_p\sigma_p}e^{-\frac{(y-\hat{y})^2}{2\hat{\sigma}_p^2}+\frac{[(\hat{y}_p-\mu\hat{\beta}]^2}{2\sigma_p^2}}\,dy\,d\hat{y}$$

其中：

$$\sigma_p = \sqrt{\sum_{l=1}^{P}\left(\sigma_p \hat{\beta}_p\right)^2}$$

表示回归估计值的均方差，σ_p为第 p 个传感器的均方差。

由定理 1，容易导出如下推论 1。

推论 1 多传感器融合模型中，多传感器融合后提供的信息量$P(C|V)$与融合结果的误差均方差呈单调减的关系。

证明：略。

上述推论再次给出了：回归偏差越大，则融合结果提供的信息量越少。因此最佳传感器融合问题实际上等价于如何从有限的传感器中选取适当的传感器，实现融合结果与真实值的偏差最小。当采用线性回归融合时，最佳传感器融合问题等价于多元线性回归的指标筛选问题。

7.3.2　进化仿真优化方法

表 7.3 进化仿真优化方法中的符号说明

符号	意义
Θ	策略集合
n	策略的个数，$n = \|\Theta\|$
Θ'	抽样策略集合
m	抽样策略个数，$m = \|\Theta'\|$
M	原问题目标个数
P	粗糙模型目标个数
G	真实性能排名前g的策略集合，构成满意策略集
S	采用粗糙模型得到的，排名前 s 的策略集合
$G \cap S$	S 集合中性能比较好的策略构成的集合
k	S 集合中和 G 集合重叠策略的度量
AP	对准概率，定义为$\mathrm{Prob}(G \cap S) > k$

在 7.3.1 中，给出了数据中心策略优化问题的多传感器融合模型。该模型针对寻求多目标规划问题的最优策略，给出了最佳传感器组合就是最

小线性回归误差回归变量的组合这一结论。通过 7.3.1 可以得到如下结论：若需要求解数据中心多目标规划问题的最优策略，可以通过线性最小二乘回归的方法求解。实际问题往往并不需要找到问题的最优策略，只要给出可以接受的若干满意策略就可以了。在表 7.3 中我们定义了常用变量。

为了了解不同目标之间的相互关系，首先从策略空间Θ中均匀分布随机采集 m 个样本，构成策略空间的一个子集$\Theta'=\{\theta_1,\theta_2,\cdots,\theta_m\}$，通过研究子集$\Theta'$各策略与综合评价函数的关系进行最佳评价目标的选择。对子集Θ'中每个策略所有目标进行真实的蒙特卡洛评价，得到子集Θ'的评价矩阵：

$$J=\begin{bmatrix} J_1(\theta_1) & J_2(\theta_1) & \cdots & J_M(\theta_1) \\ J_1(\theta_2) & J_2(\theta_2) & \cdots & J_M(\theta_2) \\ \vdots & \vdots & \ddots & \vdots \\ J_1(\theta_m) & J_2(\theta_m) & \cdots & J_M(\theta_m) \end{bmatrix}$$

记$J_i=[J_i(\theta_1),J_i(\theta_2),\cdots,J_i(\theta_m)]^T$，则$J=[J_1,J_2,\cdots,J_M]^T$。

设 $\forall\theta_i\in\Theta'$ 的精确评价为$f(J_i)=Y_i$，对综合评价 Y 和评价的各个目标 J 进行多元线性回归：

$$Y=\beta X+\varepsilon \tag{7.3}$$

其中$X=[1,J_1,J_2,\cdots,J_M]^T$，$\beta=(\beta_0,\beta_1,\cdots\beta_M)$。

若该多元线性回归无法通过线性回归 F 检验，则认为在Θ'中，综合评价函数 $f(J)$无法通过不同的目标 J_i 的线性组合得到，此时需要通过优化Θ'，使得评价函数与各目标之间在某一区间内近似呈现线性关系。

若（7.3）通过 F 检验，即综合评价 Y 与各个分目标 J_i之间的线性近似关系成立。从所有目标中，按照一定规则选择部分目标进行回归，一方面可以降低采样误差导致的回归系数的不稳定性，降低多重共线性，另一方面可以有效地降低目标的评价次数，降低仿真时间。

综上所述，本算法的基本思路是：通过对Θ'中数据特征的研究，从所有 M 个目标中选出个P个目标，$P\ll M$，构造用 P 个目标代替 M 个目标的粗糙多目标规划模型，使得粗糙模型的对准概率 AP 最大化。

设从 M 个目标中选取了 P 个目标，与常量 1 构成向量 $X^*=[1,J_1{}^*,J_2{}^*,\cdots,J_M{}^*]^T$，$J_i{}^*\in\{J_i|j=1,2,\cdots,M\}$，用该 P 个目标代替所有 M 个目标，构造回归方程$Y=\hat{\beta}X^*+\varepsilon^*$，采用线性最小二乘回归方程系数：

$$\hat{\beta}=\left(X^{*T}X^*\right)^{-1}X^{*T}Y$$

$$\hat{Y}=\hat{\beta}X^{*},\ \varepsilon_i=Y_i-X^{*}\left(X^{*T}X^{*}\right)^{-1}X^{*T}Y_i,\ S=\sum_{i=1}^{I}\varepsilon_i^2\ ,\ {\hat{\sigma}_2}^2=\frac{S}{n-P}$$

研究目标函数的真实值与估计值的联合分布$P(\hat{Y},Y)$：

设样本的综合评价真实值Y服从期望为E_y方差为σ_1的正态分布，即$P(Y)\sim N(E_Y,\sigma_1)$，真实综合评价为 Y 时，观测值$\hat{Y}$ 服从正态分布，即条件概率$P(\hat{Y}|Y)\sim N(Y,\sigma_2)$，则$P(\hat{Y},Y)=P(\hat{Y}|Y)P(Y)$，即：

$$P(\hat{Y},Y)=\frac{1}{\sqrt{2\pi}\sigma_1}e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}}\cdot\frac{1}{\sqrt{2\pi}\hat{\sigma}_2}e^{-\frac{(\hat{Y}-Y)^2}{2\hat{\sigma}_2^2}}$$

图 7.4 展示了联合概率分布示意图。

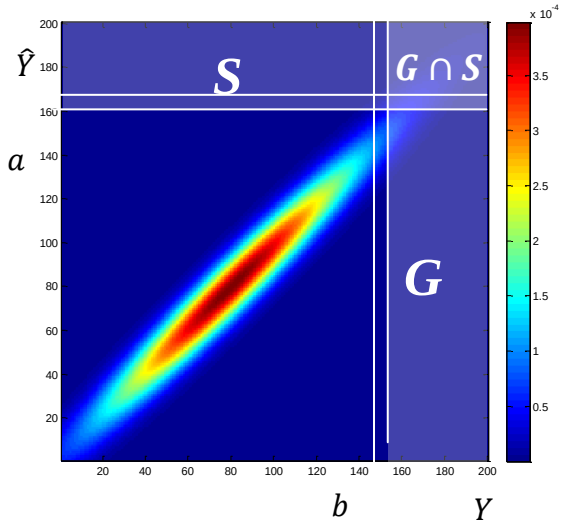


图 7.4 融合结果与真实值联合分布

S 区域为粗糙模型的综合评价超过 a 的解的集合，其概率为：

$$P(S)=\int_a^{\infty}\int_{-\infty}^{\infty}P(\hat{Y}_i,Y_i)\,dY\,d\hat{Y}\triangleq\frac{s}{n}$$

s 为粗糙模型综合评估值超过阈值 a 的策略的数量。

G 区域为精确模型的超过 b 的解的集合，其概率为：

$$P(G)=\int_{-\infty}^{\infty}\int_b^{\infty}P(\hat{Y}_i,Y_i)\,dY\,d\hat{Y}\triangleq\frac{g}{n}$$

g 为精确模型综合评估值超过 b 的策略的数量。

$S \cap G$为粗糙模型排名在前 s，且精确模型排名前 g 的策略的集合，其概率为：

$$P(S \cap G) = \int_a^{\infty} \int_b^{\infty} P(\hat{Y}, Y)\, dY\, d\hat{Y}$$

通过 $\hat{y}$选出前 s 名中，属于真实性能的前 g 名的概率为：

$$P(S \cap G|S) = \frac{P(S \cap G)}{P(S)}$$

$P(S \cap G) > k$表示集合$S \cap G$元素个数大于 k 的概率。$P(S \cap G) > k$ 的计算方法为：

$$P(|G \cap S| > k) = \sum_{i=k}^{n} C_n^i P(S \cap G|S)^i (1 - P(S \cap G|S))^{n-i}$$

我们需要搜索最佳的Q个目标来代替原来的M个目标，使得 $P(S \cap G) > k$ 最大：

$$\max: \mathrm{AP} = P(S \cap G) > k \qquad (7.4)$$

引理 1 若$P_1 < P_2$，则$\forall k \in [0, n]$ 都有：

$$\sum_{i=k}^{n} C_n^i P_1^i (1 - P_1)^{n-i} < \sum_{i=k}^{n} C_n^i P_2^i (1 - P_2)^{n-i}$$

证明：设：$P_1^j (1 - P_1)^{n-j} = P_2^j (1 - P_2)^{n-j}$

$$\left(\frac{P_1}{P_2}\right)^j \left(\frac{1 - P_1}{1 - P_2}\right)^{n-j} = 1$$

当$P_1 < P_2$，则对于 $\forall i < j$ 都有

$$\frac{P_1^i (1 - P_1)^{n-i}}{P_2^i (1 - P_2)^{n-i}}$$

$$= \left(\frac{P_1}{P_2}\right)^i \left(\frac{1 - P_1}{1 - P_2}\right)^{n-i}$$

$$= \left(\frac{P_1}{P_2}\right)^{j-(j-i)} \left(\frac{1-P_1}{1-P_2}\right)^{n-(j-(j-i))}$$

$$= \left(\frac{P_1}{P_2}\right)^{j} \left(\frac{1-P_1}{1-P_2}\right)^{n-j} \left(\frac{P_1}{P_2}\right)^{-(j-i)} \left(\frac{1-P_1}{1-P_2}\right)^{(j-i)}$$

$$= \left[\frac{(1-P_1)P_2}{(1-P_2)P_1}\right]^{j-i} > 1$$

即当：$i \le \lceil j \rceil$

$$C_n^i P_1^i (1-P_1)^{n-i} > C_n^i P_2^i (1-P_2)^{n-i}$$

同理，$i > \lceil j \rceil$

$$C_n^i P_1^i (1-P_1)^{n-i} < C_n^i P_2^i (1-P_2)^{n-i}$$

当$k-1 \le \lceil j \rceil$时，

$$\sum_{i=1}^{k-1} C_n^i P_1^i (1-P_1)^{n-i} > \sum_{i=1}^{k-1} C_n^i P_2^i (1-P_2)^{n-i}$$

$$1 - \sum_{i=1}^{k-1} C_n^i P_1^i (1-P_1)^{n-i} < 1 - \sum_{i=1}^{k-1} C_n^i P_2^i (1-P_2)^{n-i}$$

$$\sum_{i=k}^{n} C_n^i P_1^i (1-P_1)^{n-i} < \sum_{i=k}^{n} C_n^i P_2^i (1-P_2)^{n-i}$$

当$k-1 > \lceil j \rceil$时，

$$\sum_{i=k}^{n} C_n^i P_1^i (1-P_1)^{n-i} < \sum_{i=k}^{n} C_n^i P_2^i (1-P_2)^{n-i}$$

因此，$\forall k \in [0,n]$，都有

$$\sum_{i=k}^{n} C_n^i P_1^i (1-P_1)^{n-i} < \sum_{i=k}^{n} C_n^i P_2^i (1-P_2)^{n-i}$$

即，若$\mathrm{P}_1 < \mathrm{P}_2$，则$\forall k \in [0,n]$都有：

$$\sum_{i=k}^{n} C_n^i P_1^i (1-P_1)^{n-i} < \sum_{i=k}^{n} C_n^i P_2^i (1-P_2)^{n-i} \qquad \square$$

引理 2 $\forall k \in [0,n]$ $AP = \sum_{i=k}^{n} C_n^i P^i (1-P)^{n-i}$是关于 P 的单调增函数。

证明：由引理 1 得证。 □

定　理　2　：　$\forall k \in [0,n]$　，　$x_1^* = \arg\max\left(\mathrm{AP} = \sum_{i=k}^{n} C_n^i P^i (1-P)^{n-i}\right)$　，

$x_2^* = \arg\max P$，　则$x_1^* = x_2^*$

证明：由引理 2，$\mathrm{AP} = \sum_{i=k}^{n} C_n^i P^i (1-P)^{n-i}$是关于 P 的单调增函数，当 P 取得最大时，AP 可以取到最大值。 □

由定理 2，规划模型可以等价转化为：

$$\max P(S \cap G) = \int_a^{\infty} \int_b^{\infty} P(\hat{Y}, Y)\, \mathrm{d}Y\, \mathrm{d}\hat{Y}$$

$$\text{s.t.} \begin{cases} \int_a^{\infty} \int_{-\infty}^{\infty} P(\hat{Y}, Y)\, \mathrm{d}Y\, \mathrm{d}\hat{Y} = \frac{s}{n} \\ \int_{-\infty}^{\infty} \int_b^{\infty} P(\hat{Y}, Y)\, \mathrm{d}Y\, \mathrm{d}\hat{Y} = \frac{g}{n} \end{cases}$$

定理 2 当$a \le b$时，$\int_a^{\infty} \int_b^{\infty} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\hat{\sigma}_2} e^{-\frac{(\hat{Y}-Y)^2}{2\hat{\sigma}_2^2}}\, dY\, d\hat{Y}$关于$\hat{\sigma}_2$单调减。

证明：$\int_a^{\infty} \int_b^{\infty} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\hat{\sigma}_2} e^{-\frac{(\hat{Y}-Y)^2}{2\hat{\sigma}_2^2}}\, \mathrm{d}Y\, \mathrm{d}\hat{Y}$

$$= \frac{1}{\sqrt{2\pi}\sigma_1} \int_b^{\infty} e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}} \left[1 - F\left(\frac{a-Y}{\hat{\sigma}_2}\right)\right] \mathrm{d}Y$$

其中$\mathrm{F}(\cdot)$是标准正态分布的分布函数，当$\mathrm{a} \le b$，$Y \in [b, \infty)$，$a - Y \le 0$，设存在两个变量$\hat{\sigma}_{21}$和$\hat{\sigma}_{22}$，有$\hat{\sigma}_{21} > \hat{\sigma}_{22}$，则：

$$1 - F\left(\frac{a-Y}{\hat{\sigma}_{21}}\right) < 1 - F\left(\frac{a-Y}{\hat{\sigma}_{22}}\right)$$

将$\hat{\sigma}_2 = \hat{\sigma}_{21}$代入目标函数

$$Z_1 = \frac{1}{\sqrt{2\pi}\sigma_1} \int_b^{\infty} e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}} \left[1 - F\left(\frac{a-Y}{\hat{\sigma}_{21}}\right)\right] dY$$

将$\hat{\sigma}_2 = \hat{\sigma}_{22}$代入目标函数

$$Z_2 = \frac{1}{\sqrt{2\pi}\sigma_1} \int_b^{\infty} e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}} \left[1 - F\left(\frac{a-Y}{\hat{\sigma}_{22}}\right)\right] dY$$

因为$1 - F\left(\frac{a-Y}{\hat{\sigma}_{21}}\right) < 1 - F\left(\frac{a-Y}{\hat{\sigma}_{22}}\right)$，所以$Z_1 < Z_2$，得证。 □

由定理 2，当$a \le b$时，$\int_a^{\infty} \int_b^{\infty} \frac{1}{\sqrt{2\pi}\sigma_1} e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}} \frac{1}{\sqrt{2\pi}\hat{\sigma}_2} e^{-\frac{(\hat{Y}-Y)^2}{2\hat{\sigma}_2^2}}\, dY\, d\hat{Y}$关于$\hat{\sigma}_2$单调

减，所以

$$arg\,max\int_a^{\infty}\int_b^{\infty}\frac{1}{\sqrt{2\pi}\sigma_1}e^{-\frac{(Y-E_Y)^2}{2\sigma_1^2}}\frac{1}{\sqrt{2\pi}\hat{\sigma}_2}e^{-\frac{(\hat{Y}-Y)^2}{2\hat{\sigma}_2^2}}dY\,d\hat{Y}=arg\,min\,\hat{\sigma}_2$$

即规划模型可以简化为：

$$min\,\hat{\sigma}_2=\varepsilon\cdot\varepsilon^T$$

$$s.t.\begin{cases}\varepsilon=Y_i-X^*\left(X^{*T}X^*\right)^{-1}X^{*T}Y_i\\ X^*=(1,J_1^*,J_2^*,\cdots J_P^*)\\ J_i^*\in\{J_j\,|j=1,2,\cdots,M\}\end{cases}\qquad(7.5)$$

（7.5）给出了进化仿真优化方法目标约简模型，与多传感器融合模型中推论 1 的结论是一致的，即：要实现用约简目标后的粗糙模型估计原问题，估计误差越小，粗糙模型带来的信息量越大，AP 值也越高。因此目标约简问题可以转化为如何选取最佳的目标，使得回归误差最小的问题。下面给出两种求解目标优选的方法。

7.3.3　目标优选算法

（1）最优集合法。遍历所有可能的目标组合，之后对各种目标组合进行信息融合，计算融合结果所带来的信息量，得到在给定目标数量下的最佳目标组合，实现信息融合结果所带来的信息量最大化。最优集合法能够确保在给定目标数量的前提下提供最大信息量的目标组合，但是当目标的数量非常庞大时，最优集合法的计算规模将很大，这个也是我们需要避免的方向。

（2）逐步融合法。为了降低计算复杂度，可以采用贪心的算法。即逐步地引入新的目标，每次引入目标的原则为：能够在最大程度上带来信息量的增长。直观的解释就是每次加入能够带来传感器信息量的并集最大程度上增长的传感器；为了降低陷入局部最优的可能性，再加入去除目标的环节，在适当的时候去除那些对于总体信息量无关紧要的目标。这些目标在当初加入的时候可能能够带来信息量的增长，但是随着后续目标的添加，有些目标的信息量可能被后续目标覆盖，因此这些目标需要去除，以实现用最少的目标获取最大的信息量。而基于本思想又如下两种实现方法：

（2.a）基于 EcGA（Extended compact Genetic Algorithm）的最优集合法。为了获取最佳的目标组合，实现用最少的目标，获取最大的信息量，一种最简单的思路就是遍历所有可能的目标组合，从中选择一个最佳的集合，

实现目标的最佳选择。

本最佳组合问题的实质是：用最少的目标，获取最大的信息量。这是一个双目标规划问题：

$$max\, H(C \mid J_1, J_2, \cdots, J_P)$$

$$min\, P$$

求解这类多目标规划的方法主要有：

α. 使用价值函数，将多目标规划问题转化为单目标规划问题；

β. 将多目标规划问题认为是求解 Pareto 最优解集问题；

γ. 将一个目标作为主目标，其他目标选定一个可以接受的值作为约束条件，将多目标规划问题转化为单目标规划问题。

方法α常常用来求解大规模的多目标规划问题，但得到的最优解取决于价值函数的形式和参数。针对本问题，得到的最优解无法控制精度和计算复杂度。方法β能够得出所有的 Pareto 非劣前沿，不存在主观性，但是得到的是解的集合，具体决策时，还需要结合其他方法从中确定一个最佳的策略。方法γ常常用于对于某些指标有限制的多目标规划问题。例如本问题可以转化为：给定信息量限制的前提下，确定最少的传感器组合，满足信息量需求；或者：给定传感器数量（计算成本）的前提下，确定最佳传感器组合，实现获取的信息量最大化。

三种处理多目标规划方法的示意如图 7.5 所示。图中，A 解是给定传感器数量为 P 时，能够提供最大信息量的解，B 解是给定信息量比例为 95%，所用最少传感器数量的解。C 解是给定信息量和传感器数量的加权系数比为 M:1时的解。A，B，C 都是 Pareto 非劣前沿中的一个解。

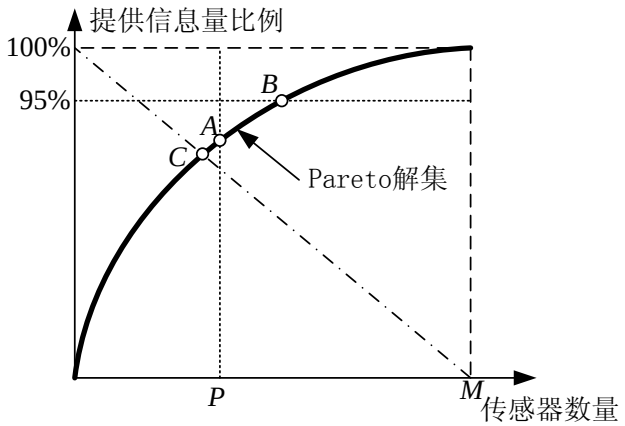


图 7.5 三种处理多目标的方法示意图

事实上，方法 γ 中的做法可以互相转化。当从 1~M 个传感器约束条件下，求解传感器的最佳组合实现信息量最大化时，等价于求解问题的 Pareto 非劣前沿。获取了 Pareto 非劣前沿后，即可得到给定信息量约束条件下，最少传感器组合。因此问题的本质可以认为是：在给定传感器数量为 P 的前提下，最佳的传感器组合，实现融合信息量最大化。即：

$$max\, H(C|J_1,J_2,\cdots,J_P)$$

$$s.t.\ \ J_i \in \{Xe_1,Xe_2,\cdots,Xe_n\}$$

由推论 1，上述优化问题可以转化为：

$$min\, \hat{\sigma}_P$$

$$s.t.\ \ J_i \in \{Xe_1,Xe_2,\cdots,Xe_n\}$$

其中$\hat{\sigma}_P$是P 个传感器融合结果与真实值的误差均方差。

上述规划问题是个大规模的组合优化问题，解空间的规模为C_M^P，当传感器的数量很庞大时，采用遍历的方法将不可解，尤其是针对数据中心性能评价问题。每种策略下，各个属性的观测值还存在一定的误差，为了降低观测误差而采取的重复观测方法将进一步增加计算量。

遗传算法是求解组合优化的有效方法，在本问题中，若采用 0-1 编码，设某个策略为 x，则

$$min\, \hat{\sigma}_P$$

$$s.t.\begin{cases} x = (x_1,x_2,\cdots,x_M) \\ x_i = 0,1 \\ \sum_{i=1}^{M} x_i = P \end{cases}$$

为了降低遗传算法的复杂度，对上式进行拉格朗日松弛：

$$min\, \hat{\sigma}_P + \lambda\left(\sum_{i=1}^{M} x_i - P\right)$$

$$s.t.\begin{cases} x = (x_1,x_2,\cdots,x_M) \\ x_i = 0,1 \end{cases}$$

对上述问题采用拉格朗日松弛算法即可将等式约束去掉。为了简化描述，后面都是针对无约束最优化问题。

从编码方式上来看，传感器的顺序决定了模式定理中优秀基因段长度。由于求解前，并不知道哪些传感器之间存在关联关系，因此可能出现相邻传感器之间无法形成优秀基因段。例如第一个传感器和最后一个传感器是问题的最佳组合，也就是凡是选中这两个传感器的染色体都具有较好的表现，但是由于这种优秀基因模式的长度跨过整个染色体，因此极易被交叉算子破坏，优秀模式无法得到遗传，此时遗传算法的效果受限。

为了克服编码方式对模式长度的影响，Harik 在 2000 年提出 EcGA[125] (Extended compact Genetic Algorithm)。EcGA 算法是 cGA(compress GA)的扩展，属于分布估计算法（Estimation of Distribution Algorithms）的一种，其基本思想是利用聚类分析的方法把所有变量划分为彼此独立的变量组，各组变量的边缘分布之积作为所有变量的联合分布，并在组的内部使用 cGA（compress GA）的算法。

在使用聚类方法划分基因组时，EcGA 采用的是最小描述长度准则（MDL）。为了降低计算复杂度，EcGA 采用一种贪心的做法，对基因进行划分。首先假设每个基因都独立成组，然后基因组两两合并，从中选出使得模型的组合复杂度最小的一种组合方式，构成一个新组。以此类推，不断地合并原有分类，直到无法合并为止。

经过分组后，组与组之间相对独立，可以确保大多数的优秀基因积木块都在分组内部，减少一个优秀基因积木块跨越多个组的可能性，降低交叉算子破坏已有优秀基因积木块的概率。

对于一个分组内部的基因是不采用交叉算子的，因此即使内部基因存在相关性，也不用担心交叉算子破坏了原有优秀基因积木块。在分组内部采用 cGA 的方法进行进化。cGA 的每一代只有两个个体，并且维护一个优胜个体基因概率表。cGA 根据优胜个体基因概率表中每个基因出现 1 的概率，随机产生两个个体，比较两个个体的适应度，若优胜个体的某个基因与淘汰个体的等位基因不一样，则修改优胜个体基因概率表，使得在下一代更有可能产生优胜个体。

将 EcGA 应用与传感器最佳组合筛选问题中的直观解释是：首先运用最小描述长度准则（MDL），根据传感器的观测值进行聚类，得到若干分组，使得组的内部的传感器存在一定的相关性，但是组与组之间的传感器相互独立。对于组内部的传感器运用 cGA 的算法进化得到组内部的最佳传感器代表；将一个组看成是一个基因，重新构造上层染色体，运用标准遗传算法求解最佳组合。具体算法如图 7.6：

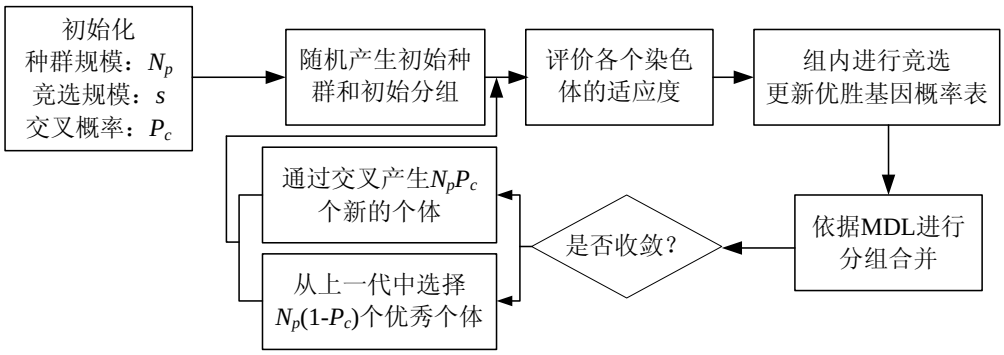


图 7.6 EcGA 算法流程图

通过上述算法即可得到最佳的传感器组合，使得传感器总的数量是 P，此时获取的信息量最大。

（2.b）基于逐步回归的逐步融合方法

上述方法是在全局范围内搜索传感器的最佳组合，如果传感器的数量过多，传感器筛选的计算量将达到一个不可接受的规模，即使采用 EcGA 也可能无法满足实际需求。为此给出基于逐步回归的逐步融合方法。

由前面分析，最佳传感器组合问题，本质是如何用最少的传感器去估计真实性能的问题。本文所采用的融合方法利用线性回归融合的方法，原问题即可转化为如何用最少变量获取最高精度的线性回归问题。

前面提到，过多的传感器可能导致传感器成本增加，数据处理复杂度升高，另一方面过多的传感器可能导致回归系数的稳定。

若各个属性之间存在相关性，即某些变量可能被其他变量线性表出时，此时该变量的回归系数就很大程度上依赖于那些表出变量的回归系数，个别的训练数据的波动，都可能导致这些相关变量的回归系数出现很大的波动，增大了采样误差对模型的影响。在线性回归中，这种现象称之为线性回归的多重共线性问题。

当回归变量之间存在相关性时，就会出现多重共线性现象。在多重共线性发生时，回归数据的细微变化都会导致回归模型的剧烈变化。在消除多重共线性的过程中，指标数越少，多重共线性越弱，粗糙模型越稳定，但此时的回归精度越低。

消除多重共线性的方法主要有先验信息法，改变变量的定义形式法，主成份分析法[125]、岭估计法[126]和逐步回归法[127]等。

先验信息法和改变变量定义的方法是指利用问题各属性之间的先验关系和变量的定义形式调整指标，该方法依赖于问题本身，不具备一般性。

主成份分析方法通过线性变换将原本相关的属性变换到相互独立的正交空间中，再用独立变量进行回归。这种方法虽然消除了多重共线性，但无法降低指标的个数，不适用于本文所提减少目标的目的。

岭估计的方法虽然可以提高参数估计的稳定性，但该方法得到的是有偏估计，影响估计精度。

逐步回归方法是目前使用较为广泛的消除多重共线性的方法，逐步回归分析方法是综合了逐步剔除法和逐步引入法的特点产生的方法。其基本原理为：从一个自变量出发，视自变量对因变量的影响显著性大小，从大到小引入回归方程，同时，在逐个自变量选入回归方程中，如果发现以前被引入的自变量在其后由于某些自变量的引入而失去其重要性. 可以从回归方程中随时予以剔除。引入一个变量或剔除一个变量，为逐步回归的一步，每步都要进行显著性检验，以便保证每次引入变量前回归方程中只包括显著性变量，这个过程反复进行，直到既无不显著变量从方程中剔除，又无显著变量需要引入回归方程为止。

运用逐步回归方法进行传感器筛选，其步骤和传统的逐步回归方法是相同的，其流程如图 7.7。通过逐步回归算法，可以逐步引入传感器，直到满足传感器数量约束或者粗糙模型估计精度约束为止。

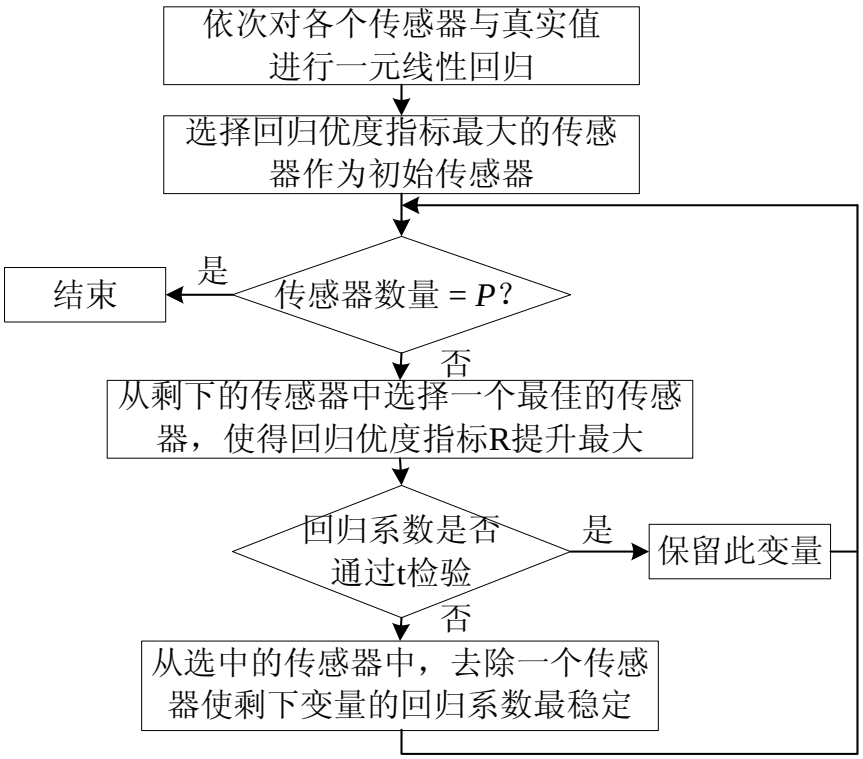


图 7.7 逐步回归算法

7.4 实例结果分析

由于弹性云计算平台的核心评价指标在目前并未取得权威机构的发布和认可，在本章的后述实验中，我们采用仿真和基于数学模型生成的随机数据上做性能测试和分析，其主要目的是评价本章中提及的基于传感器信息融合和进化仿真优化方法的有效性和合理性。

假设某弹性云计算平台数据中心的系统性能评价指标需要综合考虑 200 个性能指标，总计有 100 个不同的云计算平台组成候选策略集合，各个策略的各个目标的真实评价（多次重复的蒙特卡洛仿真求平均）如图 7.8 所示，数值按照目标之间一定的相关性生成。从任意一横行来看，性能取值之间具有一定的相关性，既两个目标的评价可能同时具有一致（同时较好或者同时较差，正相关）的评价性能趋势或者相反（一方较好另一方较差，负相关）的评价性能趋势，而策略之间没有相关性。

上述假设是合理的，比如在弹性云计算平台数据中心多评价指标构建体系中，虚拟资源的可扩展性指标与虚拟云计算平台的吞吐率是完全正相关的；而不同厂商的云计算产品由各自的核心技术来实现，彼此之间基本相关度不大。

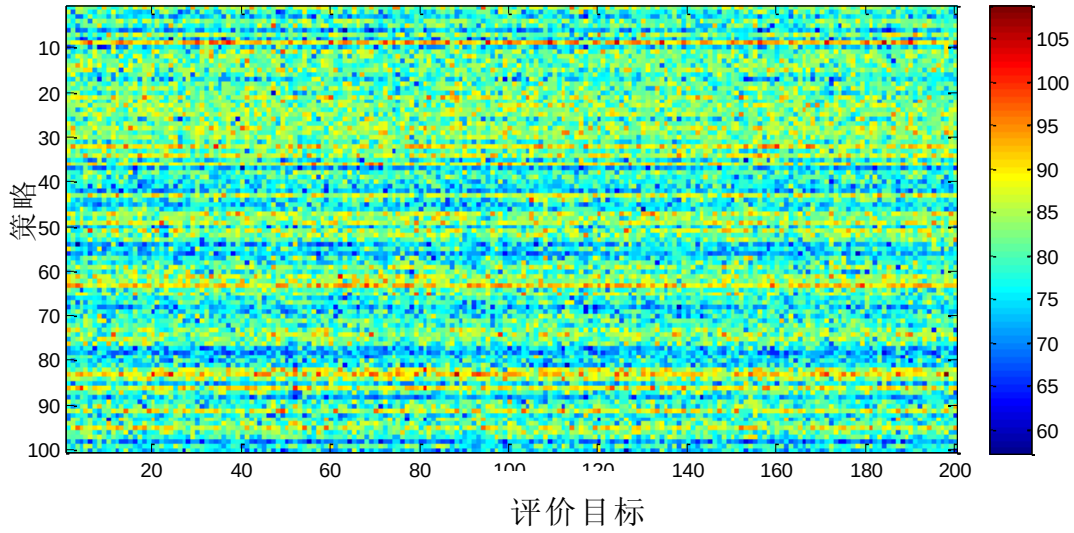


图 7.8 100 个策略，200 个评价目标的真实性能

取各个策略在所有目标下真实评价值的平均值作为本策略的综合真实性能，如图 7.9 所示。排名前 15（g=15）的策略编号为[61，28，82，47，49，43，34，95，91，36，86，32，83，63，9]构成足够好的策略集。

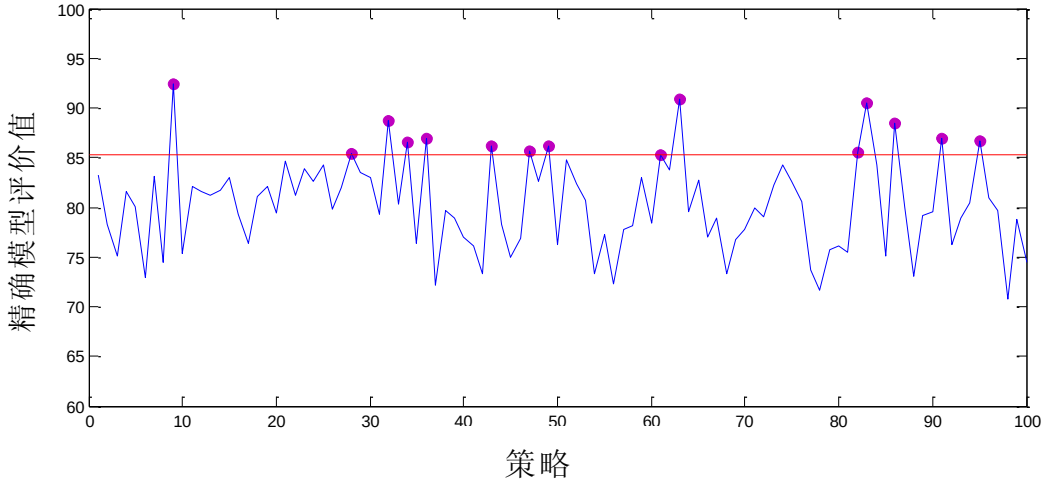


图 7.9 各个策略在所有评价目标下的平均值

优化目标是在以上 200 个评价目标中寻找若干个最佳的评价目标组合，使得给定进化仿真优化算法的搜索空间大小 s=15 （需要后期进行蒙特卡洛仿真，所以由下一阶段的计算量来确定），g=15 个足够好的策略集合，对准概率 AP 95%以上时，搜索结果中包含至少 k=10（占总搜索空间的 5%）个图 2 中足够好的策略。下面通过两类问题来进行说明：

7.4.1 问题 1：中等指标规模的进化仿真优化问题

采用文中讲述的逐步回归方法，回归残差均方差与目标个数的关系如图 7.10（回归残差均方差的定义和方法在文章里有详细的介绍）。

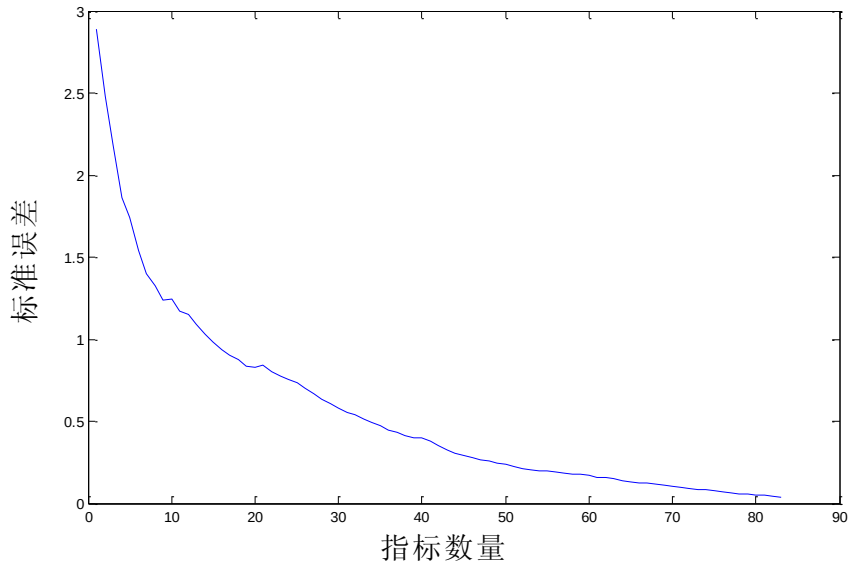


图 7.10 误差标准差与指标数量之间的关系

从图中可以看出，随着指标数量的增加，误差水平在降低，但是降低的速度在减缓。也就是说，在仿真进化的任意一个阶段选择目标约简的时候会有一个折衷，即到底选多少个评价目标作为可以接受的误差程度接受当前评估。而此时每一步进化阶段测试的目标个数的选择直接取决于当前阶段可以接受的计算开销的大小。如果计算开销足够大，则可以直接使用精确蒙特卡洛在每个目标上按照预定计算量进行分配，如果不够大，则根据标准误差取值和 EcGA 进化算法进行目标选择。

图 7.11 给出了当现在的计算量只够计算单个目标时，利用进化仿真算法进化到单一目标的粗糙模型与蒙特卡洛评价的精确模型进行的对比。

从图中可以看出，当利用逐步回归方法进化到只剩下一个目标（188 号目标）可以实现$s=15$，$g=15$，$k=8$，AP $=0.95$。此时的进化效果已经比较明显，但是依然没有满足 $k=10$ 个的评价需求，所以还需要继续增加评价目标满足 AP 的要求。

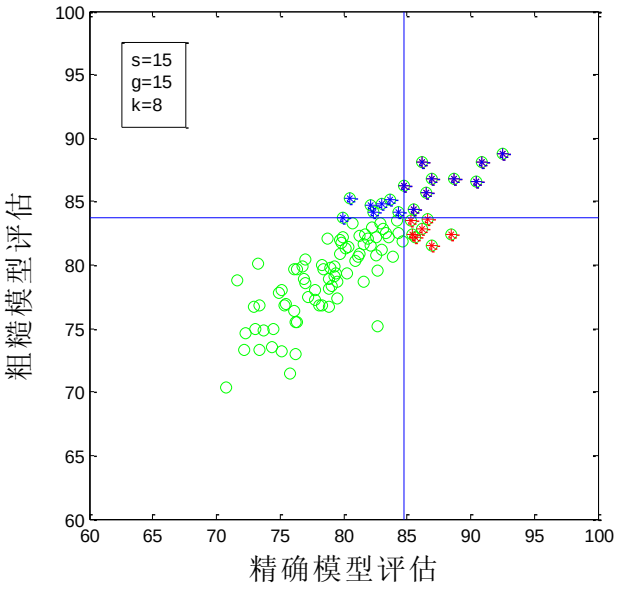


图 7.11 单目标粗糙模型与精确模型对比

图 7.12 给出了通过本进化仿真方法（正文中给出了本算法进化过程的相应步骤）优化出的最佳 7 个目标，这十个目标分别为：[188，189，131，99，170，186，59]。

此时$s=15$，$g=15$，$k=13$，$AP=0.96$，达到 k =10 要求，搜索停止。

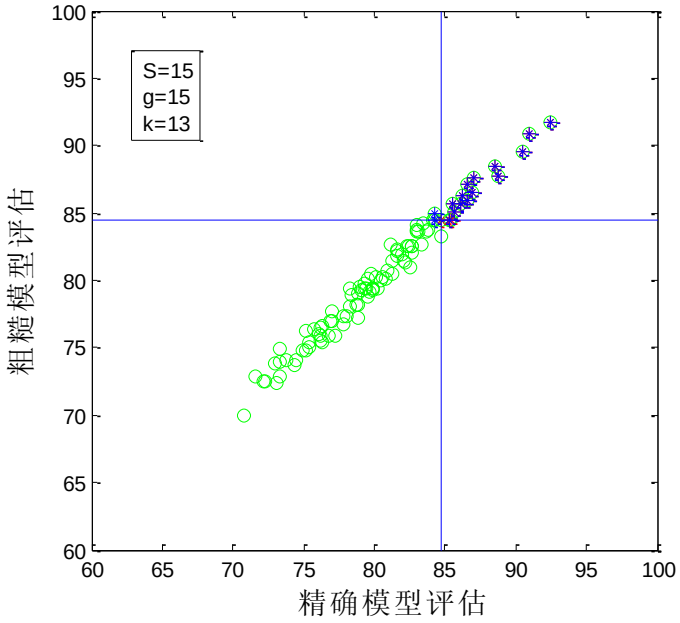


图 7.12 十个目标粗糙模型与精确模型对比

从图中可以看出，通过目标进化约简方法，原来 200 个目标的多目标规划问题经过进化约简后降为 7 个目标即可实现 AP>0.95 且 k>10 的要求，大大降低了计算时间，提升了仿真性能。

7.4.2　问题 2：大规模指标进化仿真优化及其可推广性研究：

假设多目标性能评价问题需要评价 2000 个目标，总计有 2000 个策略，从中随机抽取 10%的策略作为研究指标之间相关性，如图 6.13。同样对于本问题我们要求进化仿真优化以后对准概率 AP 满足 95%的要求且对准满意解 k 的数量大于 10 个。

图 7.13 中，左图是 2000 个策略，每个策略 2000 个目标的大型多目标评价问题的真实评价；中图是从 2000 个策略中随机抽取 10%的过程；右图是抽取出的 10%策略组成新的集合。该集合包含 200 个策略，每个策略 2000 个目标。本部分工作中我们首先将进化仿真优化方法在 200 个策略中进行，然后将其应用到剩下 1800 个策略中研究算法的推广性能。

首先分析图 7.14 中的实验结果。设置 s=10，g=5，k=5 三个指标来衡量和最大化 AP 取值，通过进化仿真优化算法，将 2000 个目标进化约简到 10 个，计算量为蒙特卡洛仿真的 0.5%就满足了试验的要求，使得 AP 大于 95%。

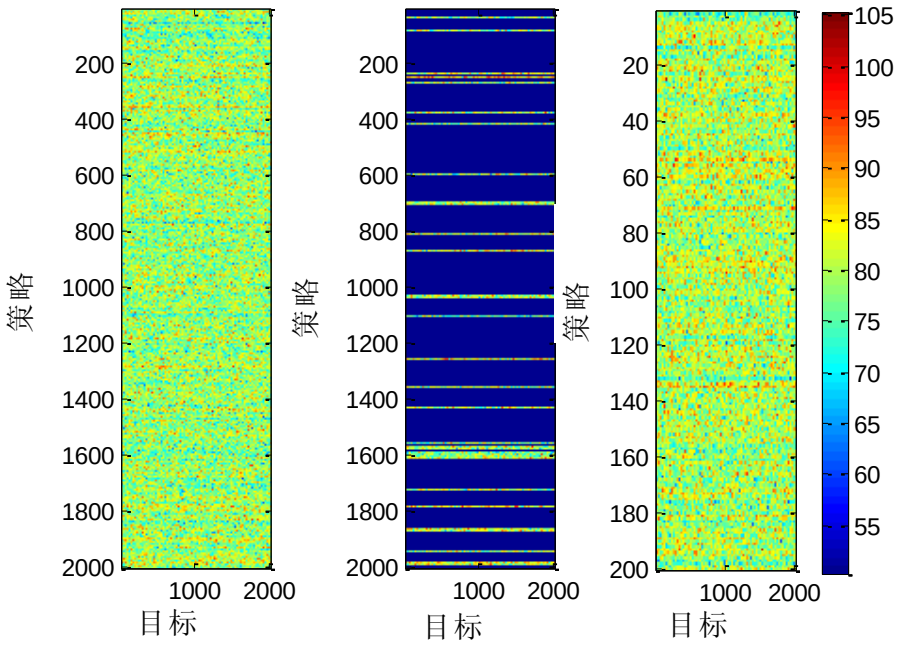


图 7.13 200 个策略，2000 个目标评估

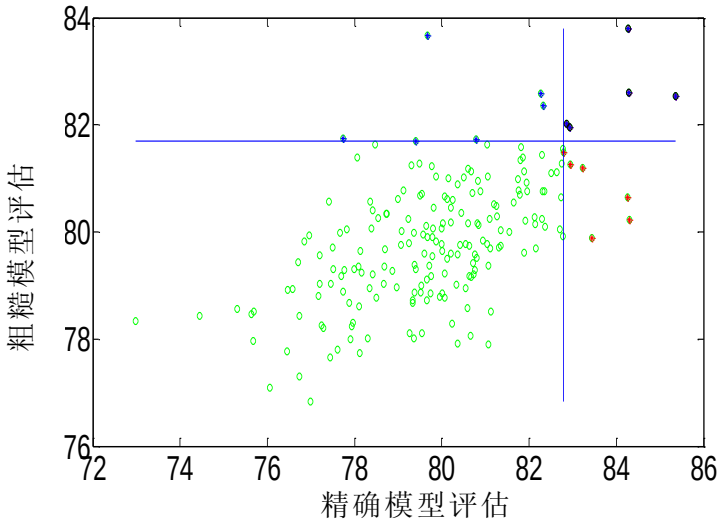


图 7.14 本问题中十个目标粗糙模型与精确模型对比

将进化仿真优化选择出的 10 个目标应用于剩余尚未评价的 1800 个策略研究模型的推广能力，此时粗糙模型的结果与真实模型的结果对比如图 7.15。

从图 7.15 中可以看出，针对本部分数据，进化仿真优化方法获取的 10 个关键指标能够在剩余 1800 个总策略，100 个满意策略中，挑出其中的 k = 84 个满意策略子集。实验结果说明了采用进化仿真优化方法设计的粗糙模

型在具有一定相关性的多指标评价系统中，具有良好的推广性，达到良好的评价效果。

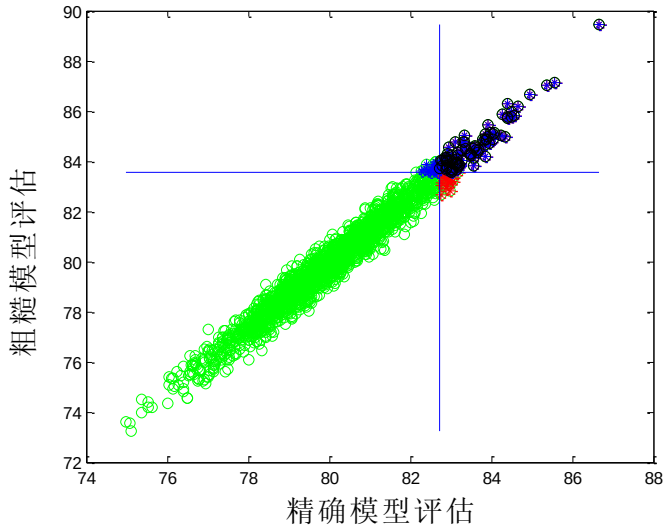


图 7.15 精确评价与粗糙模型评价结果对比

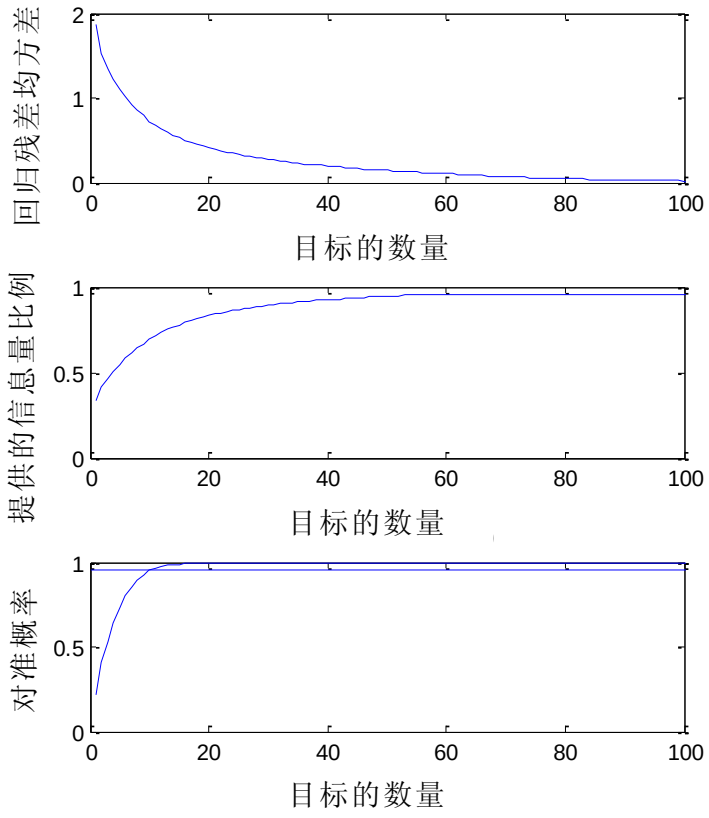


图 7.16 关键指标进化图，s=100，g=100，k=50

图 7.16 给出了几个关键指标的逐步进化图。其中上图为回归残差均方差的逐步进化图，中图为融合信息量的逐步进化图，下图为 AP 的逐步进化图。其中 AP 逐步进化图中，参数 s=100，g=100，k=50（保持和图 7.15 一样的搜索难度，选择 k 值占 g 的 50%）。针对本实验数据，在上述参数下，同样当评价目标个数进化到 10 个的时候，能够确保 AP>0.95。以下表格和图 7.17 中给出了粗糙模型目标个数与 k 的对于关系：

从表中可以看出，随着 k 的增加，对于精度的要求越来越高，所需目标的个数也越来越多，而进化仿真优化算法每次迭代过程需要多少目标则完全取决于当前可用计算量大小。

表 7.4 粗糙模型目标个数与 k 的对应关系

k	10	20	30	40	50	60	70	80	90	100
目标个数	1	2	4	7	10	15	23	36	159	2000

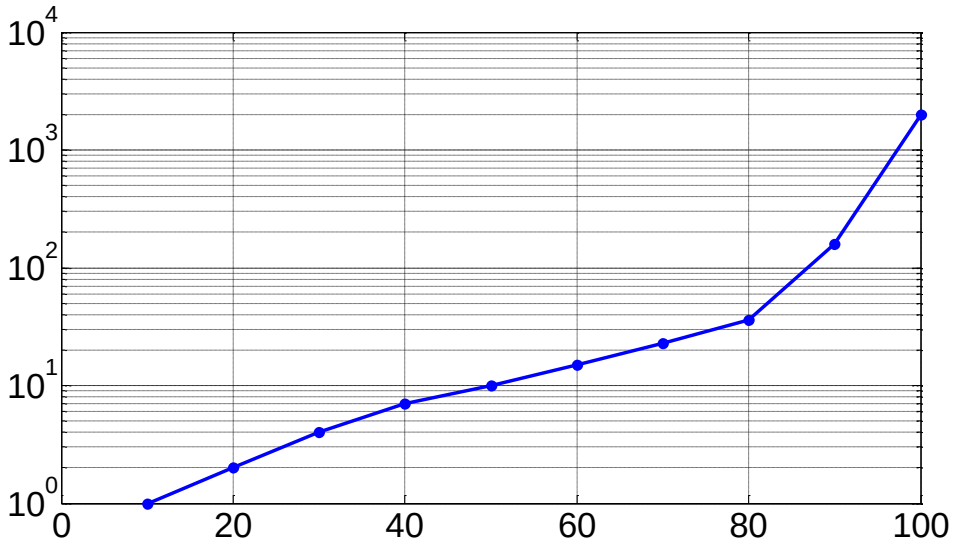


图 7.17 粗糙模型目标个数与 k 的对应关系

7.5 小结

本章是对低开销进化仿真优化方法的一个应用，其针对数据中心多目标策略优化问题，给出了两种目标约简模型，分别是传感器融合模型和进化仿真优化模型。两种模型都得到了一致的结果，即回归误差越小，粗糙模型越精确。基于这一结论，文中给出了基于 EcGA 的最优集合算法求解最佳精简

目标和基于逐步回归分析的最优集合约简算法。最后针对相关性实验数据验证了算法的可行性。

本文算法将来自 2000 个策略集中随机抽取的 200 个策略，结合 2000 个评价目标的多目标策略优化问题降低到 10 个评价目标，并且实现粗糙模型选出的15个最优策略至少包含真实前15满意策略中10个的概率大于0.95。同时将本方法选择出的目标推广到剩下的 1800 个策略中，实验结果证实了在 s=100，g=100，k=50 的新目标下，本方法能够搜索出 84 个满意策略子集，具有良好的适用性和推广型。

当然，实验结果的获取与实验数据的选取有关。假设所有目标都不关联，则“没有免费的午餐定理”能说明所有优化策略均等同于盲选法。本章中所提及的进化仿真优化方法适用于目标评价之间存有一定的关联性问题。

第8章　结论

8.1　全文研究工作总结

弹性云计算平台资源管理技术解决了构建在物理硬件资源上的虚拟资源的弹性和按需优化供给，以最合适的计算资源满足云服务提供商和终端用户在用户服务层级协议上的根本要求，提高资源综合利用率和用户满意度，这些都是云计算研究的重要内容。

相对于物理资源的供给方式，虚拟资源的存在有其特殊性。首先，虚拟机构建在 Hypervisor 之上，可以按需调整虚拟机的配置，比如虚拟 CPU 的个数，内存大小和扩展方式等等，这也是弹性云计算平台得名的原因。同时，虚拟资源能够在较高性能监控器下实现快速迁移从而实现从故障点快速恢复和调整。但是也正是因为存在这些便利，使得调度和管理虚拟资源变得异常复杂，需要综合考虑多方面的因素实现整体协同管理和优化调度。

本文围绕如何在云计算平台上弹性的提供虚拟资源这一核心问题，以性能提升和资源的敏捷供给效率为问题的根本出发点，以一条清晰的主线逐步递进的开展研究工作。

首先，针对弹性云计算平台虚拟资源的部署特点，构建了带有约束规划的数学模型，从而更加清晰的刻画了任务请求到达和响应时间之间的明确关系，实现资源快速最优分配；然而，这种快速分配并没有考虑系统运行期的诸多随机因素的存在，针对这种情况，我们进而利用仿真优化的思想在考虑虚拟机设备失效的情况下多次仿真任务的单位执行时间和内存消耗等动态因素，最后利用低开销仿真优化的调度思想来缩短单次调度时间开销；继而，我们考虑将这种敏捷调度的仿真优化方法迭代实施在多阶段调度过程中。我们在试验中反复多次执行低开销仿真优化方法，使得单次问题的求解时间大大缩短，同时也能产生比较优良的调度策略来适应剧烈变化的多任务负载；随后，我们通过分析对于调度策略有决定性因素的多任务负载的变化规律，在低开销迭代仿真优化调度方法上，自适应的在多任务负载变化剧烈之处对仿真优化区间进行宽度切割，而在变化平缓之处则对仿真优化区间进行宽度合并，从而能充分兼顾多任务负载变化的各种实际情况，实现真正的自适应弹性调度，这也是所谓的低开销进化仿真优化方法。最后，我们研究了弹性云平台多指标综合评价模型问题，通过多传感器融合和进化仿真优化方法两

方面获取了回归误差越小，粗糙模型约精确的结论，继而利用仿真优化方法的思想结合 EcGA 对多指标评价的仿真优化问题进行合理的降维，并在仿真实验中证实了方法的合理性和有效性。

本论文的主要研究结论可以从如下三点阐述：

（1）弹性云计算平台的资源的数学模型供给方法的有效性

在弹性云计算平台的资源管理逻辑层，利用多层次排队论模型构建请求在层次队列中的跳转概率，并对实际多层次平台概率进行收集和提取，可以利用数学关系直接表达请求到达率和响应时间之间的逻辑关系，公式上直接给出最佳性能虚拟资源调度方案。经过实验证实了，基于数学整数规划的约束模型以及排队论思想求解，能够在多任务负载变化相对不太剧烈，系统运行期动态性变化相对较小的情况下实现资源的有效配置和优化。

（2）弹性云计算平台的资源的敏捷调度和供给方法的有效性

在 1.5 节中我们论及了本文的工作的指导思想和核心是围绕在“敏捷”两个字的基础上进行。通过构建适应于多任务“敏捷”低开销调度方法，使本文工作能够在整个仿真过程中作出更多，细粒度的决策，虽然单一决策本身可能并非当阶段最优，但是其低开销调度方法使其能够更良好的捕获系统工作负载的局部性特点，从而决策本身更有针对性和适应性，最终导致整体吞吐率的提升和内存消耗量的减少；在弹性云计算平台综合模型评价体系中，通过对多指标进行降维，其核心也是体现在对于弹性云计算平台数据中心进行“敏捷”的评价和新核心基础之上，使得在应对大众云计算时代到来之时，用户能够更加便捷的选择适合自己的云服务提供商。弹性云计算平台对于资源的敏捷性供给，带来了服务性能的提升。

（3）仿真优化方法的进化实现

在多传感器融合模型与仿真优化目标约简两类模型中均可得出回归误差越小，粗糙模型越精确。基于这一结论，给出了建立在 EcGA 的最优集合算法之上求解最佳精简目标算法和基于逐步回归分析的最优集合精简算法。最后针对模拟数据验证了算法的可行性。通过本方法，可以将含有大量仿真优化目标的多目标策略优化问题进行降维，降维后的粗糙模型同样能够良好的应用于仿真优化和策略评价中。本降维体现了粗糙模型由原本复杂多样的实际情况，进化到一个远小于原本目标维度的目标子集中，从而实现评价的敏捷高效。

8.2　后续研究工作展望

基于现有的科研成果，探讨后续的研究思路，可以围绕以下几方面开展：

现有弹性云计算平台基于典型的多层队列模型，虽然队列节点部署并联虚拟机实例提供服务，但是其局部细节并未很好地刻画。实际上，前后层虚拟机资源是形成直接通信的复杂网络结构。后期工作可以将前后层请求转移概率，转化为虚拟机实例之间直接通信的请求转移概率，从而构建网络队列模型更好地捕获请求的局部性特征。本拓展方向则能更好的适应多任务负载的动态变化，使其和低开销迭代和进化仿真优化方法相得益彰；

现有弹性云计算平台多任务敏捷调度问题所考虑的背景是虚拟资源已经构建好并部署在资源池中等待调度的实际情形。虽然这种情况的存在具有普遍性，但是不排除某些商业或者科学应用系统在构建初期所申请资源并不足以满足不断增长的用户需求。后续工作包括针对虚拟机实例在调度之前动态创建的情形，选择和寻找在现有海量计算资源池中自动、实时和动态的构建、启停虚拟机实例执行多任务的敏捷优化调度方法。本拓展方向能够更好的体现云计算平台的弹性能力，使其在资源的有效利用上向前更进一步；

虽然对于弹性云计算平台数据中心的粗粒度综合性能评价体系可以参照本文中列举的方法来实施，但是具有针对性的细粒度评价体系仍然需要构建。具体说来，拓展工作包括针对 SaaS 和 PaaS 的多用户环境以及针对 IaaS 的多租户环境创建不同的评价模型，并开发和构建针对以上模型特征的基准测试程序，并基于现有给定的弹性云计算平台，探讨基准测试程序在更大规模的海量服务器集群中的可伸缩性。本拓展方向能够更好的、针对性的构建基于不同云计算平台的评价体系，对于尚未成熟并亟待商业化运作的云服务提供商具有指导性意义。

参考文献

[1] “SETI@HOME”[EB/OL]. http://setiathome.ssl.berkeley.edu/

[2] Abramovici A, Althouse W E, Drever R W P, et al. LIGO - the Laser Interferometer Gravitational-wave Observatory. Science, 1992, 256(5055): 325-333.

[3] Mirin A A, Wickett M. Climate Modeling using High-Performance Computing [EB/OL] //Technical Report No. UCRL-TR-220786. Lawrence Berkeley National Laboratory, Berkerley, CA.https://e-reports-ext.llnl.gov/pdf/333068.pdf

[4] Foster, I, Kesselman, C. Computational Grids. The Grid: Blueprint for a New Computing Infrastructure. Morgan Kaufmann, 1999.

[5] Foster I, Kesselman C, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. International Journal of Supercomputer Applications, 2001, 15(3):200-222.

[6] Foster I, Kesselman C, Nick J, et al. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration [EB/OL] (2002). http://www.Globus.org/research/papers/ogsa.pdf

[7] Armbrust M, Fox A, Griffith R, et al. Above the clouds: A Berkeley View of Cloud Computing, //Technical Report No. UCB/EECS-2009-28, University of California, Berkerley, Berkerley, CA, 2009.

[8] Brian H. Cloud computing. Communications of the ACM, 2008, 51(7):9-11.

[9] Buyya R, Yeo C S, Venugopal S. Market-Oriented Cloud Computing: Vision, Hype, and Reality of Delivering Computing as the 5th Utility[C]. // Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid. 2009, Shanghai, China.

[10] Miller K, Pegah M. Virtualization: virtually at the desktop[C]. // Proc. of the 35th annual ACM SIGUCCS fall conference. 2007, New York, USA.

[11] Wolf C, Halter E M. Virtualization: from the Desktop to the Enterprise. Apress, 2007.

[12] Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters[C]. //Proc. of the 6th Symposium on Operating System Design and Implementation, San Francisco, CA, December, 2004.

[13] Ghemawat S, Gobioff H, Leung S T. The Google File System[C]. // Proc. of 19th ACM Symposium on Operating Systems Principles, Lake George, NY, October, 2003.

[14] Basney J, Livny M. Deploying a High Throughput Computing Cluster. // Rajkumar

Buyya, High Performance Cluster Computing. Prentice Hall PTR, May 1999:1-20.

[15] Basney J, Livny M, Tannenbaum T. High Throughput Computing with Condor. HPC University news, 1997, 1(2) .

[16] “Amazon EC2” [EB/OL]. http://aws.amazon.com/ec2/

[17] “Amazon S3” [EB/OL]. https://s3.amazonaws.com/

[18] “Google App Engine” [EB/OL]. http://code.google.com/intl/zh-CN/appengine/

[19] Sukwong O, Kim H S. Is Co-scheduling Too Expensive for SMP VMs? [C] // Proc. of the 6th Conference on Computer Systems. 2011, New York, NY, USA.

[20] Urgaonkar B, Shenoy P, Chandray A, Goyal P. Dynamic Provisioning of Multi-tier Internet Applications[C]. //Proc. of the 2nd IEEE International Conference on Autonomic Computing, 2005, Seattle, USA.

[21] Ho Y C, Sreenivas R S. Vakili P. Ordinal Optimization of DEDS. Discrete Event Dynamic Systems. 1992, 2(1):61-88.

[22] Bhulai S, Sivasubramanian S, Mei R V D, et al. Modeling and Predicting End-to-end Response Times in Multi-tier Internet Applications[C]. //Proc. of the 20th International Teletraffic Conference on Managing Traffic Performance in Converged Networks. 2007, Ottawa, Canada.

[23] Diao Y, Hellerstein J L, Parekh S, et al. Controlling Quality of Service in Multi-Tier Web Applications[C]. // Proc. of 26th IEEE International Conference on Distributed Computing Systems, 2006, Lisboa, Portugal.

[24] Deelman E, Kesselman C, Mehta G, et al. GriPhyN and LIGO: Building a Virtual Data Grid for Gravitational Wave Scientists[C]. //Proc. of 11th IEEE International Symposium on High Performance Distributed Computing, 2002, Washington DC, USA.

[25] 许可. 网格服务流状态π演算形式化验证技术研究与应用[博士学位论文]. 北京: 清华大学自动化系, 2007.

[26] Chen C H. An Effective Approach to Smartly Allocate Computing Budget for Discrete Event Simulation[C]. //Proc. of the 34th IEEE Conference on Decision and Control, 1995, New Orleans, Louisiana, USA.

[27] “Rubis” [EB/OL].http://rubis.ow2.org/

[28] Hu L, Jin H, Liao X, et al. Magnet: A Novel Scheduling Policy for Power Reduction in Cluster with Virtual Machines[C]. //Proc. of the 10th IEEE International Conference on Cluster Computing , 2008, Tsukuba, Japan.

[29] Slothouber L. A Model of Web Server Performance[C]. // Proc. of the 5th International World Wide Web Conference, 1996, Paris, France.

[30] Doyle R, Chase J, Asad O, et al. Model-based Resource Provisioning in a Web

service utility[C]. //Proc. of the 4th USENIX Symposium on Internet Technologies and Systems, 2003, Seattle, WA, USA.

[31] Abdelzaher T, Shin K G, Bhatti N. Performance Guarantees for Web Server End-systems: A Control-theoretical Approach. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(1):80-96.

[32] Urgaonkar B, Shenoy P. Cataclysm: Handling Extreme Overloads in Internet Services[C]. //Proc. of the 23rd Annual ACM SIGACT-SIGOPS Symposium on Principles of Distributed Computing, 2004, St. John's, Newfoundland, Canada.

[33] Levy R, Nagarajarao J, Pacifici G, et al. Performance Management for Cluster Based Web Services[C]. //IFIP/IEEE 8th International Symposium on Integrated Network Management, 2003, Colorado Springs, USA.

[34] Menasce D. Web Server Software Architectures. IEEE Internet Computing, 2003, 7(6):78-81.

[35] Calheiros R N, Ranjan R, Beloglazov A, et al. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms, Software: Practice and Experience, ISSN: 0038-0644, Wiley Press, New York, USA.

[36] Kamra A, Misra V, Nahum E M. Yaksha: A Self-tuning Controller for Managing the Performance of 3-tiered Web Sites[C]. //Proc. of the 12th International Workshop on Quality of Service, 2004, Passau, Germany.

[37] Villela D, Pradhan P, Rubenstein D. Provisioning Servers in the Application Tier for e-commerce Systems. ACM Transactions on Internet Technology, 2007, 7(1):57-66.

[38] Urgaonkar B, Pacifici G, Shenoy P, et al. An Analytical Model for Multi-tier Internet Services and Its Applications[C]. //Proc. of the ACM International Conference on Measurement and Modeling of Computer Systems, 2005, Banff, Canada.

[39] Urgaonkar B, Pacifici G, Shenoy P, et al. Analytic Modeling of Multi-tier Internet Services and its Applications. ACM Transactions on the Web, 2005, 1(1): 1-35.

[40] Urgaonkar B, Shenoy P, Chandra A, et al. Dynamic Provisioning of Multi-tier Internet Applications[C]. //Proc. of the 2nd IEEE International Conference on Autonomic Computing, 2005, Seattle, USA.

[41] Urgaonkar B, Shenoy P, Chandra A, et al. Agile Dynamic Provisioning of Multi-tier Internet Applications. ACM Transactions on Adaptive and Autonomous Systems, 2008, 3(1):1-39.

[42] Rolia J, Sevcik K. The Method of Layers. IEEE Transactions on Software Engineering. 1995, 21(8): 689–700.

[43] Woodside C, Raghunath G. General Bypass Architecture for High-performance Distributed Algorithms[C]. //Proc. of the 6th IFIP Conference on Performance of Computer Networks, 1995, Istanbul, Turkey.

[44] Franks R G. Performance Analysis of Distributed Server Systems[D]. 1999, Ph.D. thesis, Carleton University.

[45] Liu T K, Kumaran S, Luo Z. Layered Queueing Models for Enterprise Java Beans Applications[C]. //Proc. of the 5th IEEE International Conference on Enterprise Distributed Object Computing, 2001, Washington, DC, USA.

[46] Xu J, Oufimtsev A, Woodside M, et al. Performance Modeling and Prediction of Enterprise Java Beanswith Layered Queuing Network Templates[C]. //Proc. of the 2005 Conference on Specification and Verification of Component-based Systems. 2005, New York, NY, USA.

[47] Kounev S, Buchmann A. Performance Modeling and Evaluation of Large-scale J2EE Applications[C]. //Proc. of the Computer Measurement Group's International Conference, 2003, Dallas, TX.

[48] Benani M, Menasce D. Resource allocation for Autonomic Data Centers Using Analytic Performance Models[C]. //Proc. of IEEE International Conference on Autonomic Computing, 2005, Seattle, WA.

[49] Menasce D, Almeida V, Dowdy L. Performance by Design: Computer Capacity Planning by Example. 2004, Prentice Hall.

[50] Duato J, Yalamanchili S, Ni L M. Interconnection Networks: An Engineering Approach. Morgan Kaufmann Publishers Inc. 2002, San Francisco, CA, USA.

[51] Cohen I, Chase J, Goldszmidt M, et al. Correlating Instrumentation Data to System States: A Building Block for Automated Diagnosis and Control[C]. //Proc. of the 6th USENIX Symposium in Operating Systems Design and Implementation, 2004, San Francisco, CA.

[52] Rao J, Xu C Z. Online Measurement of the Capacity of Multi-tier Websites Using Hardware Performance Counters[C]. //Proc. of the 28th International Conference on Distributed Computing Systems, 2008, Beijing, China.

[53] Zhang Q, Cherkasova L, Mi N, et al. A Regression-based Analytic Model for Capacity Planning of Multi-tier Applications. Cluster Computing. 2008, 11(3):197-211.

[54] Liu X, Sha L, Froehlich S, Hellerstein J L, et al. Online Response Time Optimization of Apache Web Server[C]. //Proc. of the 11th International Workshop on Quality of Service, 2003, San Jose, CA, USA.

[55] Raghavachari M, Reimer D, Johnson R D. The Deployer's Problem: Configuring

Application Servers for Performance and Reliability[C]. //Proc. of the 25th International Conference on Software Engineering, 2003, Portland, Oregon, USA .

[56] Zhang Y, Qu W, Liu A. Automatic Performance Tuning for J2EE Application Server Systems[C]. //Proc. of the 6th International Conference on Web Information System Engineering, 2005, New York City, USA.

[57] Scutari C P, Morajko A, Margalef T, et al. Scalable Dynamic Monitoring, Analysis and Tuning Environment for Parallel Applications, Journal of Parallel Distributed Computing. 2009, 70(4):330-337.

[58] Morajko A, Margalef T, Luque E. Design and Implementation of a Dynamic Tuning Environment. Journal of Parallel and Distributed Computing, 2007, 67(4):474 – 490.

[59] Berman F, Chien A, Cooper K, et al. The GrADS Project: Software Support for High-Level Grid Application Development. International Journal of High Performance Computing Applications, 2001, 15(4):327-344.

[60] Casanova H, Zagorodnov D, Berman F, et al. Heuristics for Scheduling Parameter Sweep Applications in Grid Environments[C]. //Proc. of the 9th Heterogeneous Computing Workshop, 2000, Washington, DC, USA.

[61] Maheswaran M et al. Dynamic Mapping of a Class of Independent Tasks onto Heterogeneous Computing Systems. Journal of Parallel and Distributed Computing, 1999, 59(2):107–131.

[62] Deelman E et al. Mapping Abstract Complex Workflows onto Grid Environments. Journal of Grid Computing, 2003, 1(1):25–39.

[63] Taylor I, Shields M, Wang I, et al. Triana Applications within Grid Computing and Peer to Peer Environments. Journal of Grid Computing, 2003, 1(2):199–217.

[64] Oinn T, Addis M, Ferris J, et al. Taverna: A Tool for the Composition and Enactment of Bioinformatics Workflows. Bioinformatics, 2004,20(17):3045–3054.

[65] Erwin D W, Snelling D F. UNICORE: A Grid Computing Environment[C]. //Proc. of the 7th International Euro-Par Conference Manchester on Parallel Processing, 2001, London, UK.

[66] Laszewski G V, Amin K, Hategan M, et al. A Client-Controllable Grid Workflow System[C]. //37th Hawai International Conference on System Science, 2004, Island of Hawaii, Big Island.

[67] Braun T D, Siegel H J, Beck N. A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems. Journal of Parallel and Distributed Computing, 2001, 61(6):810–837.

[68] Armstrong R, Hensgen D, Kidd T. The Relative Performance of Various Mapping Algorithms is Independent of Sizable Variances in Run-time Predictions[C]. //Proc.

of the 7th IEEE Heterogeneous Computing Workshop, 1998, Orlando, FL , USA.

[69] Freund R F, Gherrity M, Ambrosius S, et al. Scheduling Resources in Multi-user, Heterogeneous, Computing Environments with SmartNet[C]. //Proc. of the 7th IEEE Heterogeneous Computing Workshop, 1998, Orlando, FL , USA.

[70] Freund R F, Siegel H J. Heterogeneous processing, IEEE Computer, 1993, 26(6): 13-17

[71] Ibarra O H, Kim C E. Heuristic Algorithms for Scheduling Independent Tasks on Nonidentical Processors, Journal of the Association of Computer Society. 1977, 24(2): 280-289.

[72] Holland J H. Adaptation in Natural and Artificial Systems, University of Michigan Press, Ann Arbor, MI, 1975.

[73] Michalewicz Z, Fogel D B. How to Solve It: Modern Heuristics, Springer-Verlag, New York, 2000.

[74] Singh H, Youssef A. Mapping and Scheduling Heterogeneous Task Graphs Using Genetic Algorithms[C]. //Proc. of the 5th IEEE Heterogeneous Computing Workshop, 1996, .

[75] Tirat-Gefen Y G, Parker A C. MEGA: An Approach to System-level Design of Application Specific Heterogeneous Multiprocessors[C]. //Proc. of the 5th IEEE Heterogeneous Computing Workshop, 1996, US.

[76] Wang L, Siegel H J, Roychowdhury V P, et al. Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-algorithm-based Approach. Journal of Parallel Distributed Computing, 1997, 47(1):1-15.

[77] Laarhoven P J V, Aarts E H. Simulated Annealing: Theory and Applications. 1987, Springer.

[78] Kirkpatrick S, Gelatt C D, Vecchi M P. Optimization by Simulated Annealing, Science, 1983, 220(4598) :671-680.

[79] Chen H, Flann N S, Watson D W. Parallel Genetic Simulated Annealing: A Massively Parallel SIMD Approach, IEEE Transactions on Parallel and Distributed Computing, 1998, 9(2):126-136.

[80] Shroff P, Watson D, Flann N, et al. Genetic Simulated Annealing for Scheduling Data-dependent Tasks in Heterogeneous Environments[C]. //Proc. of the 5th IEEE Heterogeneous Computing Workshop, 1996, US.

[81] Falco I D, Balio R D, Tarantino E, et al. Improving Search by Incorporating Evolution Principles in Parallel Tabu Search[C]. //Proc. of the 1994 IEEE Conference on Evolutionary Computation, 1994, Orlando, Florida, USA.

[82] Glover F, Laguna M. Tabu Search. Kluwer Academic, Boston, MA, 1997.

[83] Kafil M, Ahmad I. Optimal Task Assignment in Heterogeneous Distributed Computing Systems, IEEE Concurrency, 1998, 6(3) :42-51.

[84] Russell S J, Norvig P. Artificial Intelligence: A Modern Approach. Prentice Hall, Englewood Cliffs, NJ, 1995.

[85] Shen C C, Tsai W H. A Graph Matching Approach to Optimal Task Assignment in Distributed Computing System Using a Minimax Criterion. IEEE Transactions on Computer, 1985, 34(3):197-203.

[86] Chow K, Liu B. On Mapping Signal Processing Algorithms to a Heterogeneous Multiprocessor System[C]. //Proc. of the 1991 International Conference on Acoustics, Speech, and Signal Processing, 1991, Toronto, Ontario, Canada.

[87] Dogan A, Özgüner F. Biobjective Scheduling Algorithms for Execution Time–Reliability Trade-off in Heterogeneous Computing Systems. The Computer Journal, 2005, 48(3):300-314.

[88] Dogan A, Özgüner F. Matching and Scheduling Algorithms for Minimizing Execution Time and Failure Probability of Applications in Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems, 2002, 13(3):308–323.

[89] Lee Y C, Zomaya A Y. On the Performance of a Dual-Objective Optimization Model for Workflow Applications on Grid Platforms, IEEE Transactions on Parallel and Distributed Systems, 2009, 20(9):1273–1284.

[90] Dongarra J J, Jeannot E, Saule E, et al. Bi-objective Scheduling Algorithms for Optimizing Makespan and Reliability on Heterogeneous Systems[C]. //Proc. of the 19th annual ACM Symposium on Parallel Algorithms and Architectures. 2007, New York, NY, USA

[91] Prodan R, Wieczorek M. Bi-criteria Scheduling of Scientific Grid Workflows. IEEE Transactions on Automation Science and Engineering, 2010, 7(2):364-376.

[92] Hakem M, Butelle F. A Bi-objective Algortithm for Scheduling Parallel Applications on Heterogeneous Systems Subject to Failures[C]. //Proc. of the Renpar, 2006, Perpignan, France.

[93] Li A, Yang X, Kandula S, et al. Comparing Public Cloud Providers. IEEE Internet Computing, 2011, 15(2):50-53.

[94] Li A, Yang X, Kandula S, et al. CloudCmp: Comparing Public Cloud Providers[C]. //Proc. of the 10th annual conference on Internet measurement, 2011, New York, NY, USA.

[95] “Microsoft Azure” [EB/OL]. http://www.microsoft.com/windowsazure/

[96] “Rackspace” [EB/OL]. http://www.rackspace.com/index.php

[97] “Standard Performance Evaluation Corporation” [EB/OL] http://www.spec.org/jvm2008/

[98] Chen C H, Fu M, Shi L. Simulation and Optimization, Tutorials in Operations Research, Informs, Hanover, MD, 2008.

[99] Fu M, Chen C H, Shi L, Some Topics for Simulation Optimization[C]. //Proc. of 2008 Winter Simulation Conference, 2008, Miami, FL, USA.

[100] Chen C H, Lin J, Yücesan E, et al. Simulation Budget Allocation for Further Enhancing the Efficiency of Ordinal Optimization. Journal of Discrete Event Dynamic Systems: Theory and Applications, 2000, 10(3):251-270.

[101] Chen C H, He D, Fu M, et al. Efficient Simulation Budget Allocation for Selecting an Optimal Subset. Informs Journal on Computing, 2008, 20(4):579-595.

[102] Fu M C, Hu J Q, Chen C H, et al. Simulation Allocation for Determining the Best Design in the Presence of Correlated Sampling, Informs Journal on Computing, 2007 19(1):101–111.

[103] Chen C H, Yücesan E, Dai L, et al. Efficient Computation of Optimal Budget Allocation for Discrete Event Simulation Experiment, IIE Transactions, 2010, 42(1):60-70.

[104] Shi, L. Chen C H, A New Algorithm for Stochastic Discrete Resource Allocation Optimization, Journal of Discrete Event Dynamic Systems: Theory and Applications, 2000, 10(3):271-294.

[105] He D, Lee L H, Chen C H, et al. Simulation Optimization Using the Cross-Entropy Method with Optimal Computing Budget Allocation, 2009, 20(1):1-22.

[106] Lee L H, Chew E P, Teng S Y, et al. Optimal Computing Budget Allocation for Multi-objective Simulation Models[C]. //Proc. of 2004 Winter Simulation Conference, 2004, Washington D.C., USA.

[107] Chen E J, Lee L H. A Multi-objective Selection Procedure of Determining a Pareto Set. Computers and Operations Research, 2009, 36(6): 1872-1879.

[108] Lee L H, Chew E P, Teng S Y, et al. Finding the Pareto Set for Multi-objective Simulation Models, to appear in IIE Transactions.

[109] Teng S, Lee L H, Chew E P. Integration of Indifference-zone with Multi-objective Computing Budget Allocation. European Journal of Operational Research, 2010, 203(2):419-429.

[110] Hsieh B W, Chen C H, Chang S C. Efficient Simulation-based Composition of Dispatching Policies by Integrating Ordinal Optimization with Design of Experiment. IEEE Transactions on Automation Science and Engineering, 2007, 4(4):553-568.

[111] Hsieh B W, Chen C H, Chang S C. Scheduling Semiconductor Wafer Fabrication by

Using Ordinal Optimization-Based Simulation. IEEE Transactions on Robotics and Automation, 2001, 17(5):599-608.

[112] Chen, C H, He D. Intelligent Simulation for Alternatives Comparison and Application to Air Traffic Management. Journal of Systems Science and Systems Engineering, 2005, 14(1):37-51.

[113] Chen C H, Donohue K, Yücesan E, et al. Optimal Computing Budget Allocation for Monte Carlo Simulation with Application to Product Design. Journal of Simulation Practice and Theory, 2003, 11(1):57-74.

[114] Chen C H, Wu S D, Dai L. Ordinal Comparison of Heuristic Algorithms Using Stochastic Optimization. IEEE Transactions on Robotics and Automation, 1999, 15(1):44-56.

[115] Chew E P, Lee L H, Teng S Y, et al. Differentiated Service Inventory Optimization using Nested Partitions and MOCBA. Computers and Operations Research, 2009, 36(5):1703-1710.

[116] Lee L H, Chew E P, Teng S Y, et al. Multi-objective Simulation-based Evolutionary Algorithm for an Aircraft Spare Parts Allocation Problem, European Journal of Operational Research, 2008, 189(2):476-491.

[117] Arlitt M, Jin T. Workload Characterization of the 1998 World Cup Web Site. //Technical report, HPL-1999-35R1, HP Labs.

[118] Murphy M A, Kagey B, Fenn M, et al. Dynamic Provisioning of Virtual Organization Clusters[C]. //Proc. of the 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, 2009, Washington, DC, USA.

[119] Emeneker W, Jackson D, Butikofer J, et al. Dynamic Virtual Clustering with Xen and Moab, ISPA 2006, Springer-Verlag LNCS 4331

[120] Ho Y C, Zhao Q C, Jia Q S. Ordinal Optimization: Soft Optimization for Hard Problems, 2007, New York, NY: Springer.

[121] 贾庆山. 增强序优化理论研究及应用[博士学位论文]. 北京: 清华大学自动化系, 2008.

[122] 吕天文. 企业级数据中心评价体系初探[EB/OL]. 2009. http://home.jifang360.com/space.php?uid=326&do=blog&id=185

[123] Saaty T L. Analytic Hierarchy Process. John Wiley & Sons, Ltd, 2005.

[124] Harik G R, Lobo F G. Sastry K. Linkage Learning via Probabilistic Modeling in the Extended Compact Genetic Algorithm, Scalable Optimization via Probabilistic Modeling, Springer. 2006, 33(12):39-61.

[125] Jolliffe I. Principal Component Analysis. John Wiley & Sons, Ltd, 2005.

[126] Wang C, Liu H. Generalized Ridge Estimation and Its Application in Factor

Analysis Method[C]. //International Conference on Research Challenges in Computer Science, 2009, Shanghai, China.

[127] David A. Freedman, Statistical Models: Theory and Practice, Cambridge University Press. 2005.

致　谢

衷心感谢导师中国工程院院士、清华大学自动化系吴澄教授对我的指导。师者，所以传道授业解惑也。四年以来，导师孜孜不倦对我传学术之道、授科研之业、解生存之惑，已深深的影响了我，给我一生难忘的箴言。

衷心感谢副导师清华大学信息技术研究院院务委员会副主任曹军威研究员对我四年以来的精心指导。曹老师为人师表，作风严谨，对科学兢兢业业，对事业孜孜不倦，对生活坦坦荡荡，是我一生效仿的楷模。

衷心感谢南加州大学电子工程系、清华大学 IV 讲席教授黄铠两年以来对我科研和论文的帮助。您对我严厉的批评依然在耳边萦绕，却时刻督促我进步。

感谢 CIMS 研究中心宋士吉教授、范玉顺教授、刘连臣副教授、稽承军老师、智网中心的贾庆山副教授、中国科学院自动化技术研究所沈震博士对我研究工作的批评与建议，感谢所里全体同学的热情帮助和支持。感谢美国 IBM 公司慷慨的为我提供两年的博士生英才计划奖学金资助。感谢 IBM 中国开发中心蔡弘博士在实习期给我的指导。感谢麻省理工学院空间实验室 LIGO 项目组 Erik Katsavounidis 资深研究科学家、Lindy Blackburn 博士在我访问期间提供的帮助。感谢实验室的张文、殷杰、王震、李俊伟、万宇鑫、陈伟、涂国煜、杜剑平、高建芳、刘晓苗等同学，和你们在一起的时光永远难忘。

最后，特别的感谢要献给我的家人，他们精神上的鼓励，物质上的支持和生活上的帮助。

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：_____

个人简历、在学期间发表的学术论文与研究成果

个人简历

1983 年 6 月 22 日出生于湖北武汉市。

2001 年 9 月考入湖北工业大学计算机科学与技术学院，2005 年 7 月本科毕业并获得工学学士学位。

2005 年 9 月考入华中科技大学控制科学与工程系，2007 年 7 月硕士毕业并获得工学硕士学位。

2007 年 9 月考入清华大学自动化系攻读控制科学与工程博士至今。

在学期间发表的学术论文

[1] **Zhang F**, Cao J, Hwang K, and Wu C. Ordinal Optimized Scheduling of Scientific Workflows in Elastic Compute Clouds[C]. // Proceedings of the 3rd IEEE International Conference on Cloud Computing Technology and Science, Athens, Greece, 2011. （EI 源刊）

[2] **Zhang F**, Cao J, Cai H, Mulcahy J, and Wu C. Adaptive Virtual Machine Provisioning in Elastic Multi-tier Cloud Platforms[C]. // The 6th IEEE International Conference on Networking, Architecture, and Storage, DaLian, China, 2011. （EI 源刊，检索号：20113814348319）

[3] **Zhang F**, Cao J, Cai H, Mulcahy J, and Wu C. Adaptive Resource Provisioning in Virtualized Cloud Platforms. International Journal of Web Services Research. 2011, 8(3):54-69. (SCI 源刊 IF: 1.053)

[4] **Zhang F**, Cao J, Song X, Cai H, Wu C. AMREF: An Adaptive MapReduce Framework for Real Time Applications[C]. // The 9th International Conference on Grid and Cloud Computing. 2010:157-162. （EI 收录，检索号: 20112914162180）

[5] **Zhang F**, Cao J, Liu L, Wu C. Qualification Evaluation in Virtual Organizations Based on Fuzzy Analytic Hierarchy Process[C]. // 7th International Conference on Grid and Corporative Computing, 2008:539-547. （EI 收录，检索号: 20090111828628）

[6] **Zhang F**, Cao J, Liu L, Wu C. Fast Autotuning Configurations of Parameters in Distributed Computing Systems using Ordinal

Optimization[C]. The 38th International Conference on Parallel Processing Workshops, 2009:190-197. （EI 收录，检索号: 20101212783019）

[7] **Zhang F**, Cao J, Liu L, Wu C. Adjacent Matrix based Deduction for Grid Workflow Applications[C]. // The 1st International Conference on Networking and Distributed Computing, 2010: 349-356.（EI 收录，检索号: 20110113538109）

[8] **Zhang F**, Cao J, Cai H, Liu L, Wu C. Redundant Virtual Machines Management in Virtualized Cloud Platform. International Journal on Modeling, Simulation, and Scientific Computing, 2011, 2(2) : 151-168. （EI 收录，检索号: 20113114191489）

[9] **Zhang F**, Cao J, Liu L Wu C. Performance Improvement of Distributed Systems by Autotuning of the Configuration Parameters. Tsinghua Science and Technology. 2011,16(4):440-448.（EI 源刊，检索号：20113114207386）

[10] Cao J, **Zhang F**, Xu K, Liu L Wu C. Formal Verification of Temporal Properties for Reduced Overhead in Grid Scientific Workflows. Journal of Computer Science & Technology. （SCI 源刊，影响因子：0.656）

[11] Mulcahy J-J, Huang S, Cao J, **Zhang F**. How Are You Feeling? A Social Network Model to Monitor the Health of Post-Operative and Remote Patients[C]. // IEEE International Systems Conference, Montreal, Canada, 2011. (EI 收录，检索号：20113014173940)

[12] Zhao C, Cao J, Wu H, **Zhang F**. Cost Estimation of Advance Reservations over Queued Jobs: a Quantitative Study. International Journal on Modeling, Simulation, and Scientific Computing, 2010, 1(3):317-332. (EI 收录，检索号：20113514276158)

专利和软件著作权

[13] 曹军威，**张帆**. 提高分布式系统性能调优速度的方法. 国家发明专利 申请号: 200910088225 公开号: 101609416

[14] 曹军威，**张帆**，许可. 网格服务流建模和验证系统. 软件著作权. 版本号: V1.0 登记号: 2009SRBJ5041

[15] 曹军威，**张帆**，吴冲. 工作流程建模和梳理系统.软件著作权. 版本号: V1.0 登记号: 2009SRBJ5042

在学期间参与出版的专著

[16] Cao J, **Zhang F**, Xu K, Liu L, Wu C. From Enabling to Ensuring Grid Workflows, In L. Wang and J. Chen (Eds.), Quantitative Qality of Service

for Grid Comuting: Applications for Heterogeneity, Large-Scale Distributed and Dynamic Environments, IGI Publishing, 2008.

在学期间获得的主要奖励

在学期间个人获奖情况主要包括：

[1] 2011—2012 年度 IBM Ph.D. Fellow（大中华区 10 人，工程领域 3 人）

[2] 2010—2011 年度 IBM Ph.D. Fellow（大中华区 11 人，工程领域 2 人）

[3] 2008 年度国家档案科技成果三等奖（主要完成人）

[4] 2010 年昌平第二届晨光杯青年创业大赛金奖

[5] 清华大学光华奖学金（2009-2010 年度校级综合三等奖）

[6] 清华大学腾讯奖学金（2008-2009 年度校级综合二等奖）

[7] 清华大学光华奖学金（2007-2008 年度校级综合三等奖）

在学期间参加的科研项目

[1] 国家重点基础研究发展计划（973 计划）项目“高通量计算系统的构建原理、支撑技术及云服务应用”（No.2011CB302505），2011 年 1 月至 2015 年 8 月。

[2] 国家重点基础研究发展计划（973 计划）项目“物联网基础理论及设计方法研究”（No.2011CB302805），2011 年 1 月至 2013 年 8 月。

[3] 国家高技术研究发展计划（863 计划）项目“数字农业知识网格关键技术研究”（No.2006AA10Z237），2006 年 12 月至 2010 年 6 月。

[4] 国家高技术研究发展计划（863 计划）项目“海量农业知识资源组织、高效存储技术研究”（No.2007AA01Z197），2007 年 7 月至 2009 年 12 月。

[5] 国家高技术研究发展计划（863 计划）项目“大规模网络数据集成与查询技术及其应用研究”（No. 2008AA01Z118），2008 年 1 月至 2009 年 12 月。

[6] 教育部高等学校本科教学质量与教学改革工程项目 “国家精品课程集成项目门户分系统研制”，2007 年 6 月至 2010 年 6 月。

[7] 国家自然科学基金项目 “动态时变约束下的赛百平台资源优化理论与算法研究”（No.60803017），2009 年 1 月至 2011 年 12 月。