



本科生毕业论文（设计）

Undergraduate Graduation Thesis (Design)

题目 基于 ACO 的一种云平台任务调度
Title

算法研究

院系 数据科学与计算机学院
School (Department)

专业 软件工程
Major

学生姓名 陈慕远
Student Name

学号 16340025
Student No.

指导教师（职称） 温武少（教授）
Supervisor (Title)

时间： 2020 年 3 月 25 日

Date: March 25, 2020

说 明

1. 毕业论文（设计）的写作格式要求请参照《中山大学本科生毕业论文的有关规定》和《中山大学本科生毕业论文（设计）写作与印制规范》。
2. 除完成毕业论文（设计）外，还须填写三份表格：
 - （1）表一 毕业论文（设计）开题报告；
 - （2）表二 毕业论文（设计）过程检查情况记录表；
 - （3）表三 毕业论文（设计）答辩情况登记表。
3. 上述表格均可从教务部主页的“下载中心”处下载，如表格篇幅不够，可另附纸。每份毕业论文（设计）定稿装订时应随同附上这三份表格。
4. 封三是毕业论文（设计）成绩评定的主要依据，请认真填写。

Instruction

1. Please refer to '*The Guidelines to Undergraduate Graduation Thesis (Design) at Sun Yat-sen University*' and '*The Writing and Printing Format of Undergraduate Graduation Thesis(Design) at Sun Yat-sen University*' for anything about the thesis format.
2. Three forms should be filled up before the submission of the thesis (design):
 - （1）Form 1: Research Proposal of Graduation Thesis.
 - （2）Form 2: Process Check-up Form.
 - （3）Form 3: Thesis Defense Performance Form.
3. All the above forms could be downloaded on the website of the Office of Education Administration. If there is not enough space in the form, please add extra sheets. Each thesis (design) should be submitted together with the three forms.
4. The form on the inside back cover is the grading sheet. Please fill it up before submission.

表一：毕业论文（设计）开题报告
Form 1: Research Proposal of Graduation Thesis (Design)

<div>论文（设计）题目</div> <div>Thesis (Design) Title: 基于 ACO 的一种云平台任务调度算法研究</div>	
<p>近年来，云计算——一种基于互联网的，可按需将共享的软硬件资源 and 信息提供给计算机各种终端和其他设备的计算方式，开始在多个产业中兴起。传统的云计算平台在网络基础设施架构、服务模式、部署上基本采用一种普适、通用、集中的设计思路，而这种设计方案只适合以计算为主要特征的云服务需要，在各行各业对云计算的分布式、集群式、大并发、高带宽和实时服务的高要求方面面临重大挑战。新型的云计算体系架构中，计算任务的分配、调度算法对服务提供的质量有很大影响。</p> <p>本课题研究基于蚁群算法的一种优化调度算法。主要目标为缩短算法处理后的任务完成时间，以及提高虚拟机的负载均衡程度。改进这两方面将优化用户体验，具有一定的现实意义。</p> <p>进度控制：计划在 2020 年一月底前进行大部分代码阅读工作以及部分代码编写工作；在 2020 年 2 月底前完成大部分代码工作，开始论文撰写工作；在 2020 年 3 月 27 日前完成论文初稿。</p> <div style="display: flex; justify-content: space-between; margin-top: 20px;"><div>Student Signature:</div><div>Date:</div></div>	

指导教师意见

Comments from Supervisor:

1.同意开题

2.修改后开题

3.重新开题

1.Approved()

2. Approved after Revision ()

3. Disapproved()

Supervisor Signature:

Date:

表二：毕业论文（设计）过程检查情况记录表
Form 2: Process Check-up Form

指导教师分阶段检查论文的进展情况（要求过程检查记录不少于 3 次）
The supervisor should check up the working process for the thesis (design) and fill up the following check-up log. At least three times of the check-up should be done and kept on the log.

第 1 次检查（First Check-up）：

学生总结

Student Self-summary: 已综合阅读几篇相关论文，确定了项目的实际方向和以后的工作内容。

指导教师意见

Comments of Supervisor:

第 2 次检查（Second Check-up）：

学生总结

Student Self-summary: 已完成项目相关运行环境的搭建以及大部分代码阅读工作，准备进行具体的代码工作。

指导教师意见

Comments of Supervisor:

第 3 次检查 (Third Check-up):

学生总结

Student Self-summary: 已完成项目的大部分内容, 准备进行论文的撰写并进一步完善项目代码。

指导教师意见

Comments of Supervisor:

第 4 次检查

Fourth Check-up

学生总结

Student Self-summary: 已完成项目的代码部分工作, 并提交了论文初稿, 准备进行论文的进一步修改。

指导教师意见 (Comments of Supervisor):

学生签名 (Student Signature):

日期 (Date):

指导教师签名 (Supervisor Signature):

日期 (Date):

总体完成情况
(Overall
Assessment)

指导教师意见 Comments of Supervisor:

- 1、按计划完成, 完成情况优 (Excellent): ()
- 2、按计划完成, 完成情况良 (Good): ()
- 3、基本按计划完成, 完成情况合格 (Fair): ()
- 4、完成情况不合格 (Poor): ()

指导教师签名 (Supervisor Signature):

日期 (Date):

表三：毕业论文（设计）答辩情况登记表

Form 3: Thesis Defense Performance Form

答辩人 Student Name		专 业 Major	
论文（设计）题目 Thesis (Design) Title			
答辩小组成员 Committee Members			
<div>答辩记录 Records of Defense Performance:</div> <div></div> <div>记录人签名（Clerk Signature）:日期（Date）:</div>			

学术诚信声明

本人所呈交的毕业论文，是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。本毕业论文的知识产权归属于培养单位。本人完全意识到本声明的法律结果由本人承担。

本人签名：

日期：

Statement of Academic Integrity

I hereby acknowledge that the thesis submitted is a product of my own independent research under the supervision of my supervisor, and that all the data, statistics, pictures and materials are reliable and trustworthy, and that all the previous research and sources are appropriately marked in the thesis, and that the intellectual property of the thesis belongs to the school. I am fully aware of the legal effect of this statement.

Student Signature:

Date:

【摘 要】

作为一项新兴的、热门的 IT 技术，云计算提供了一种方便快捷、按需自取的商业服务模式。云计算由许多项子技术共同构成，包括而限于虚拟机调度、任务调度。其中任务调度控制了任务在虚拟机资源间的分配调度，决定着任务的完成时间与虚拟机资源的负载均衡程度，对用户体验好坏有着很大影响。

本文对云计算背景下的任务调度算法进行了深入的分析、研究。为了提高云计算平台下的用户体验，笔者试图在对多种传统调度算法进行分析的基础上对蚁群算法进行改进，以得到一种优化的蚁群算法，主要优化方向在于算法的完成时间、负载均衡度和计算时间。论文的主要内容和如下：

- (1) 首先对云计算、任务调度、仿真实验工具等一些相关知识进行介绍。
- (2) 其次对一些传统的任务调度算法进行介绍，分析它们的优缺点。重点介绍其中的蚁群算法，结合蚁群算法在云平台问题中的使用，为之后的工作做准备。
- (3) 为了对蚁群算法进行优化，笔者从蚁群算法在云平台任务调度问题中的信息素配置与虚拟机负载入手，并创新地引入投票机制，试图改善算法处理后任务的完成时间及虚拟机的负载均衡情况。
- (4) 之后，笔者于仿真平台搭建了实验环境，并对多个调度算法开展了负载均衡、完成时间两方面的对比实验。经过实验，笔者验证了本文所提出算法的可用性及先进性。

【关键词】 云计算；任务调度算法；完成时间

[ABSTRACT]

As an emerging and popular computing technology, cloud computing provides a not only convenient but also fast and on-demand business service model. Cloud computing is composed of a variety of sub-technology, including VM-scheduling, task-scheduling, etc. Task-scheduling controls the allocation and scheduling of tasks among multiple virtual machines, determines the completion time of tasks and the load-balancing of virtual machines, and thus has a great impact on user experience.

This paper conducts in-depth analysis as well as research on task-scheduling algorithms in the context of cloud computing. In order to improve the user experience under cloud computing platform, the author try to obtain an optimized, well-balanced ant colony algorithm to reduce makespan and computing time on the basis of other traditional scheduling algorithms. The main contents and work of this paper are as follows:

- (1) Firstly, the related theories such as cloud computing, task-scheduling, and simulation experiment tools are introduced.
- (2) Secondly, some traditional task-scheduling algorithms are introduced and analyzed, both their advantages and disadvantages are distinctly illustrated. It then focuses on the ant colony algorithm, which is the very basis for the next step.
- (3) Thirdly, the author mainly concentrates on the allocation of the pheromone and the load-balance of the virtual machines, together with an innovative voting scheme to develop an optimized ant colony algorithm, which performs well in both the completion time of tasks and the final result.
- (4) Fourthly, the author builds an environment on the simulation platform, then carries out comparative experiments on load-balance and makespan for multiple scheduling algorithms. The experiments verify the usability and advancement of the algorithm.

[Keywords] Cloud computing; Task scheduling; makespan

目 录

第一章 概述/引言.....	1
1.1 研究背景.....	1
1.2 国内外研究现状.....	1
1.3 本文的工作.....	2
1.4 论文结构简介.....	3
第二章 相关技术概述.....	4
2.1 云计算.....	4
2.1.1 云计算的定义.....	4
2.1.2 云计算的服务模型.....	5
2.1.3 虚拟化.....	6
2.2 云计算下的任务调度.....	8
2.2.1 任务调度概述.....	8
2.2.2 任务调度的评价标准.....	9
2.3 CloudSim.....	10
2.3.1 CloudSim 简要介绍.....	10
2.3.2 CloudSim 系统架构.....	11
第三章 一种优化的任务调度算法研究.....	14
3.1 常用任务调度算法.....	14
3.2 蚁群算法.....	14
3.3 一种优化的蚁群算法.....	15
第四章 仿真实验与结果分析.....	20
4.1 实验配置.....	20
4.2 参数配置.....	21
4.3 实验结果.....	24
第五章 总结与展望.....	27
5.1 总结.....	27
5.2 优缺点.....	27
5.3 展望.....	27
参考文献.....	29
致谢.....	31

第一章 概述/引言

1.1 研究背景

近年来,计算机技术不断革新,互联网与 IT 产业迅速发展,随着并行计算^[1]、网格计算^[2]、分布式计算^[3]的相继成熟,云计算——一种基于互联网的,可按需将共享的软硬件资源和信息提供给计算机各种终端和其他设备的计算方式,也开始在多个产业中兴起。传统的云计算平台在网络基础设施架构、服务模式、部署上基本采用一种普适、通用、集中的设计思路,而这种设计方案只适合以计算为主要特征的传统云服务需要。在各产业对云计算的分布式、集群式、大并发、高带宽和实时服务的高要求下,这种传统的设计思路受到极大挑战。新型的云计算体系架构需要充分考虑到应用场景与网络特征来构建微云系统,这其中,计算任务的分配、调度算法对服务提供的质量有很大影响。

任务的调度算法的好坏决定了任务的完成效率的高低。用户提交任务后,系统就要根据调度算法做出决策,决定把任务分配给哪些虚拟机。一个好的调度算法其目标是实现一种最高效的任务到机器的映射,从而使得任务的完成时间(第一个任务开始到最后一个任务完成的时间)最短。更短的完成时间意味着商业系统中更低的成本(对服务商)及更优的服务(对用户),因此,如何改进优化算法,缩短任务完成时间,是一个很有实际意义的问题。

基于上述现实,作者提出一个想法,可以对常用的任务调度算法如 Min-Min、ACO 进行深入探究,并做出一些优化,从而得到一个更好的任务调度方案。

1.2 国内外研究现状

云计算的起源最早可以追溯到上世纪中叶的大型机时代,而其实际的概念则最早于 1996 年在 Compaq Computer Corporation 的一份内部文件中出现。经过了半个多世纪的发展,各种新的概念、技术、方法层出不穷。现如今,像 Google(谷歌)、Amazon(亚马逊)、Microsoft(微软)这些互联网的巨头公司都或早或晚的在云计算领域开展了自己的业务。

Google 作为可能是世界最著名的搜索业务的提供者,其用户群是非常之庞大的,这就对 Google 的搜索服务提出了很高的要求。每时每刻都有大量的数据查询请求流经 Google 的服务器,传统的集中计算式平台当然无法满足 Google 的需求,

经过专业人员的努力，Google 开发了他们所专有的云平台服务（Google Cloud Platform），它提供基础架构即服务（IaaS）、平台即服务（PaaS）以及无服务器的计算环境。此外，Google 的云平台服务还是其云服务的一部分，其他服务包括 Google 公有云架构、G Suite、Android 和 Chrome OS 的企业版以及用于机器学习和企业映射的应用程序编程接口（API）服务。

Amazon 早在 2006 年就发布了自己的弹性计算云服务（Elastic Compute Cloud），它使用户可以随时通过 Internet 使用虚拟计算机集群。现如今 Amazon 的弹性计算云服务已是 AWS（Amazon Web Services）所提供服务的一部分，AWS 作为 Amazon 旗下的子公司，以按需付费的方式向个人、公司和政府提供按需云计算的平台及 API。

Microsoft 创建的云计算服务名为 Microsoft Azure，用户可通过 Microsoft 管理的数据中心进行应用程序的构建、测试、部署与管理，它提供了软件即服务（SaaS），平台即服务（PaaS）和基础架构即服务（IaaS）三种服务模式，支持许多不同的编程语言、工具和框架。

我国对于云计算的研究相对来说起步较晚，目前国内较为著名的一些云计算服务提供商有阿里云、腾讯云、华为云、百度云等。其中，阿里云因为起步较早，在世界云计算服务中也可排到前列，其余一些公司所提供的云计算服务则仍在发展中，但也可以预见巨大的潜力。

另一方面，就任务调度算法来说，国内外也已有很多任务调度算法被提出。有一些实现较为简单的调度算法，如先到先服务（FCFS）算法，Min-Min 算法，Max-Min 算法、优先级调度算法等，也有一些结构比较复杂的调度算法，比如模拟退火算法、蚁群算法、粒子群算法等。

1.3 本文的工作

本文主要以学习——分析——优化——验证的流程进行，从对其他一些任务调度算法的学习开始，再对这些算法进行适当优化，最后根据实验做出验证而结束，主要工作包括：

- 对包括 Min-Min 算法、蚁群算法在内的任务调度算法进行研究，其中，重点对蚁群算法进行学习、研究。跟随算法的工作流程来分析相应任务调度

算法的优劣，探寻可以优化的部分。

- 从蚁群算法在云平台任务调度问题中的信息素配置与虚拟机负载入手进行优化。
- 进行实验验证。

1.4 论文结构简介

本文共分为五个章节，内容组织如下：

第一章是概述/引言部分，首先介绍了本文的研究背景，接着对国内外当前云计算技术、任务调度算法的研究现状做介绍，最后说明本文所做的工作以及文章结构。

第二章是相关技术概述，首先对云计算进行介绍，包括其定义及体系结构。接着对云计算的任务调度方面做一个详细的介绍，最后对仿真实验工具 CloudSim 的架构、模块、功能进行说明。

第三章是对任务调度算法的研究，对一些传统任务调度算法进行深入分析，其中着重研究蚁群算法在云计算任务调度中的原理，详细阐述算法的工作流程，从任务的完成时间等方面探讨算法的优势与不足，并引出一种优化的任务调度算法。

第四章是仿真实验与结果分析，将对上述优化的任务调度算法与原始的蚁群算法做比较实验分析。

第五章是总结与展望，将对本文工作的优缺点做分析，并对未来可能的改进工作做出展望。

第二章 相关技术概述

2.1 云计算

随着计算机技术的迅速发展，计算任务的处理方式也发生了很大变化。从一开始的集中式处理——将计算任务交予一个中心大型计算机进行统一处理；到之后的分布式处理——构建基于网络的分布式处理系统，将大的计算任务分布交予网络中的计算机分别处理，最后整合；再到如今的云计算式处理——不仅限于计算方面，而更是一种服务，提供的是一种按需服务模式，用户无须关心内部实现细节，而只需根据自己需要申请相应资源。云计算提供可靠，安全，容错，可持续和可扩展的计算服务，并以软件即服务、基础架构即服务或平台即服务的形式呈现。本节将对云计算的相关部分做阐述。

2.1.1 云计算的定义

由上可见，云计算即是一种服务方式，它提供了一种对计算机系统资源（一般是存储或计算）的按需获取模式，在服务过程中用户不会直接接触到实际的资源管理细节，而只需即拿即用。云计算的理想目标是用户能够通过网络访问服务，以在任何时间、地点都能够根据自己需要而在虚拟资源池中获取资源来进行自己的计算任务。

在云计算中，“云”一般指的是一些大型的数据中心（如服务器集群），用户可以通过网络进行访问。现今的一些大型云，往往除了中心服务器集群外还在多个地方区域建有集群，这样当用户申请服务时，一些距离他更近的边缘服务器就会被指派给他，减少网络路径损耗带来的费用。

云计算具有如下五个关键特性^[4]：

- 按需自助服务。用户可自行获取计算资源如服务器访问时间与网络存储，无需与服务提供商进行人工交互。
- 广泛的网络访问。功能可通过网络获得，并通过标准机制进行访问，这种标准机制促进功能在异构型平台上的通用性。
- 资源池式服务。利用多租户模型将提供商的计算资源集中起来为多个用户提供服务，并根据用户的需求动态分配物理和虚拟资源。

- 弹性配置。功能可被弹性提供或取消，以便根据需求快速地向外或向内扩展功能。对于用户来说，可用于配置的功能通常看起来是无限的，并且可以随时以任意数量使用。
- 可测的服务。云系统可以根据服务类型来自动施加计量功能，以便监控与报告资源使用情况，从而为服务的提供者和服务的使用者提供服务的透明性。

2.1.2 云计算的服务模型

云计算主要有三种服务模型，分别是基础架构即服务（IaaS）、平台即服务（PaaS）、软件即服务（SaaS）。

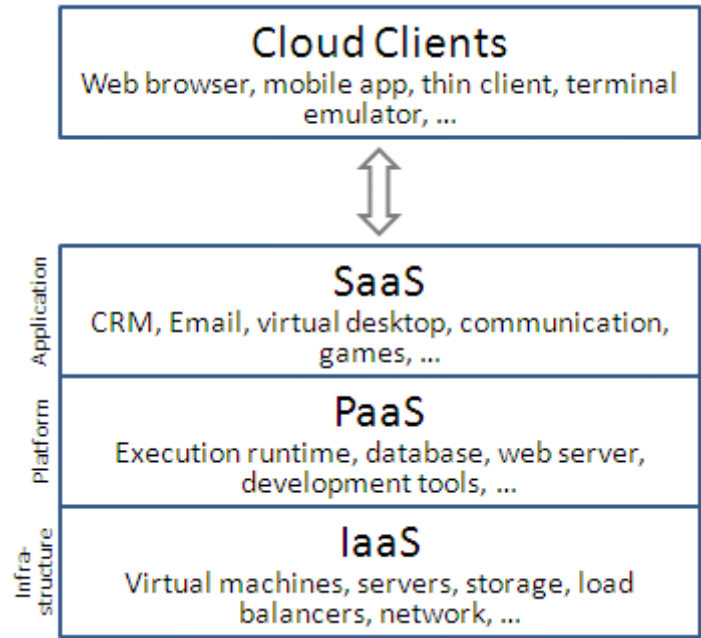


图 1 云计算服务模型

基础架构即服务（Infrastructure as a Service）。根据 NIST（National Institute of Standards and Technology）所给的定义，IaaS 提供这样一种服务，用户能在其上部署运行任意软件，包括操作系统与应用程序。用户不直接管理和控制云的底层基础架构，但能够控制操作系统、存储以及部署的应用程序，还对网络组件有一定程度的控制。服务提供商则从其部署在数据中心的大型设备集群按需提供这些设备。

平台即服务（Platform as a Service）。根据 NIST 所给的定义，PaaS 提供这样一种服务，用户可使用服务提供者支持的编程语言、库、工具创建应用程序并将其部署到云上。在这个过程中，用户不对网络、服务器、操作系统及存储在云的底层基础架构进行管理或控制，但可以控制已部署的应用程序，以及对应用程序的托

管环境进行一些配置设置。

软件即服务（Software as a Service）。根据 NIST 所给的定义，SaaS 提供这样一种服务，用户可通过网络浏览器界面或程序界面来从各种客户端设备访问应用程序。用户不对包括网络、服务器、操作系统、存储甚至单个应用程序功能在内的云底层基础架构进行管理或控制，且可能会受限于特定于用户的应用程序配置设置。也就是说，用户在此处只是应用程序的使用者。

2.1.3 虚拟化

虚拟化指的是一种创建事物的虚拟版本的行为，通常包括虚拟计算机硬件平台、存储设备与计算机网络资源。其概念最早诞生于 1960 年代，当时指的是一种于逻辑上划分大型计算机在不同应用间提供的系统资源的方法，今天其概念已被延拓伸展许多。虚拟化主要可分为以下几种：

- 网络虚拟化。将可用带宽分成多个通道来整合网络中的可用资源的方法，通道间彼此独立，且可被实时地重分配给特定的服务器或设备。如此将网络分为可管理的部分，可以隐藏网络的复杂性，提高网络资源的利用率。
- 存储虚拟化。将异构的存储资源组成一个巨大的存储池，对于用户来说透明化了底层的存储细节，直接进行存储。可以使管理人员将不同的存储作为单个集合的资源来进行识别、配置和管理。
- 服务器虚拟化。抽象化服务器的各种资源（如磁盘存储、带宽、内存等），对用户隐藏服务器资源的复杂细节，将单台服务器虚拟化为多台相互独立的虚拟服务器，可提高资源共享能力和利用率，并保持日后扩展的能力。
- 桌面虚拟化。对计算机的终端系统（如工作站）进行虚拟化，用户可通过任意设备在任何时间、地方访问自己的桌面系统（实际在远端服务器运行），提供了良好的安全性与便捷性。
- 应用虚拟化。封装应用程序，为应用程序提供一个虚拟的运行环境，使得应用程序与操作系统解耦，解决程序版本不兼容的问题（如使 windows 应用程序可在 linux 上运行）。

虚拟化的实现形式也有两种。一种是直接在物理硬件上运行虚拟化监控器（Hypervisor），即裸机虚拟机管理程序，如 Linux KVM、Vmware ESXi、Xen、Hyper-

V 等。

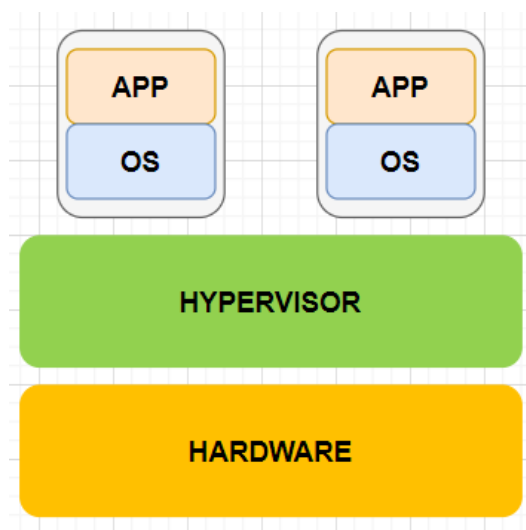


图 2 裸机虚拟化

第二种是作为一个应用程序在现有的操作系统上运行（操作系统安装于裸机上），如 VirtualBox、Vmware workstation、Xvisor、Lguest 等。

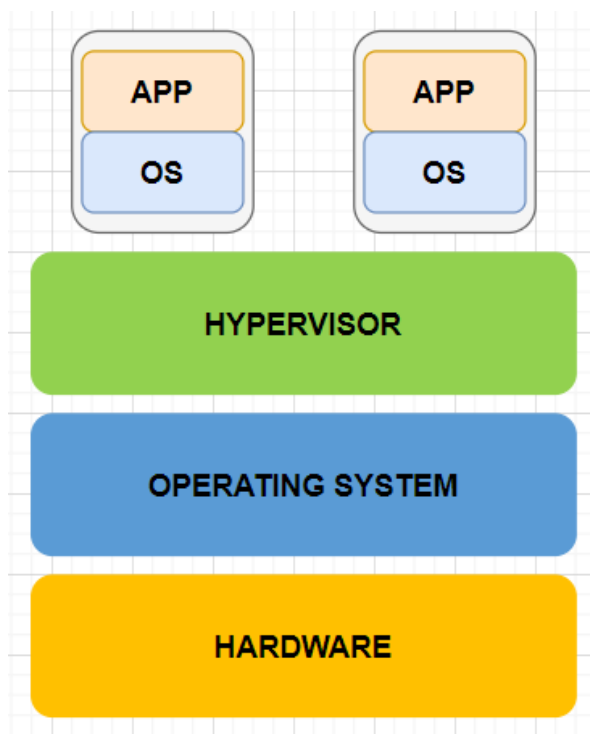


图 3 应用程序虚拟化

2.2 云计算下的任务调度

2.2.1 任务调度概述

在复杂且虚拟化的云计算平台下，任务调度是一个具有深刻现实意义的问题。一般来说，任务调度的目的是在任务 $\text{Task} = \{T_1, T_2, T_3, \dots, T_m\}$ 与虚拟机资源 $\text{Resource} = \{R_1, R_2, R_3, \dots, R_n\}$ 间构成一种最佳映射，使得任务与其最合适的虚拟机相互匹配。好的任务调度算法可以优化任务的完成时间（makespan），提高资源利用率，以及减小负载的压力，对服务商与用户来说可以构成一个双赢的局面。

云计算平台下的任务调度过程通常如下图所示：

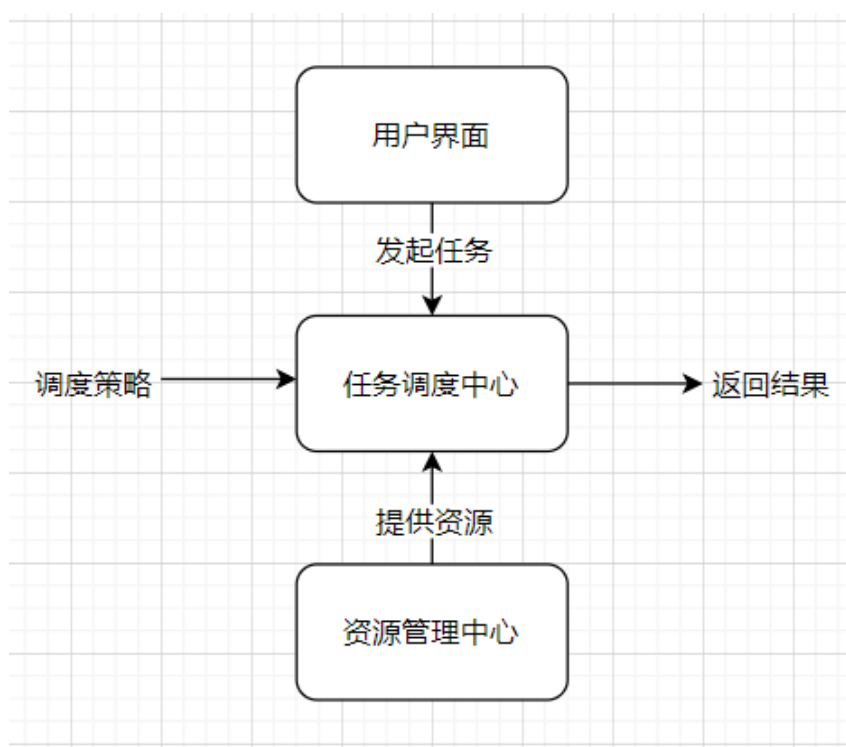


图4 任务调度过程

(1) 用户发起任务

用户通过服务商提供的服务界面（通常为 web 界面）提交任务，任务内容通常需要包含以下内容：首先，任务的执行类型、属性信息，如计算任务、配置管理任务等；其次，任务的执行条件，某些任务可能设定了执行时间限制、执行标准等；最后，任务的执行要求，指对云计算平台的一些硬性要求，如处理器核数、带宽、存储大小等。这些信息都会经任务调度中心进行处理。

(2) 监控资源信息

实际工作中云计算平台的各种硬件资源的使用情况往往较为复杂，因此

需要设置资源监控系统。资源管理中心实时接收资源监控系统返回的资源利用情况，在任务调度进行时据此向任务调度中心提供可利用的资源。

（3） 设定任务调度策略

从资源管理中心获取到足够的资源，并从用户界面获取到任务相关信息后，如果可能，任务调度中心会将一些大的任务分割，形成大小一定的一些小任务。之后，任务调度中心将依据设定的任务调度策略（可能有多个，由情况不同设定不同策略），实现任务与虚拟机资源间的映射关系，尽可能地优化任务的分配。

（4） 处理返回结果

最终，各节点将运行后的结果反馈给系统，系统整合任务运行情况后向用户汇报运行情况。

2.2.2 任务调度的评价标准

在云计算应用商品化早已非常成熟的今天，同类型、不同类型的服务都是层出不穷。无论是服务提供商还是用户都需要一些科学合理的评判标准来评估服务的好坏，以下是一些常见的任务调度的评判标准：

- （1） 任务完成时间（**makespan**）：指的是从第一个任务开始到最后一个任务完成的时间，这一项指标反映了任务调度算法的执行效率，在节奏越来越快的今天是一项很重要的评判指标。
- （2） 系统负载：指的是任务分配给虚拟机资源执行的过程中，虚拟机集群的负载均衡程度。一般来说，负载过于集中时，某些虚拟机闲置而另一些虚拟机高负荷运转，会导致虚拟机总体运行效率下降，因此任务调度算法也应该尽可能地追求负载的均衡。
- （3） 费用：对于商业化平台来说，无论服务提供商还是用户都会对费用比较在意。任务调度算法的好坏可能影响虚拟机的处理时间乃至数据的传输，进而影响服务费用。

本文中主要采用任务完成时间以及系统负载程度作为指标对任务调度算法进行比较分析。

2.3 CloudSim

已经广泛发展的云体系生态系统，以及不断增长的对 IT 技术节能方面的需求，要求在云产品实际开发之前就需要以可重复可控制的方法来评估算法、应用程序及策略。若使用实际测试平台进行测试，将限制实验在测试平台的规模上，并使得实验结果的再现极为困难。一种合适的替代方式使用仿真工具进行测试，如 CloudSim。

2.3.1 CloudSim 简要介绍

CloudSim 由澳大利亚墨尔本大学的网络实验室和 Gridbus 项目组联合推出，它是在离散事件模拟包 SimJava 基础上开发的函数库，继承了 GridSim 的编程模型，支持云计算的研究^[5]。

CloudSim 提供一个通用的、可扩展的仿真框架，该框架能够对云计算基础架构和应用程序服务进行无缝建模、仿真和实验。通过使用 CloudSim，研究人员以及开发人员可以专注于他们特定的系统设计、算法设计问题，而不必关注基于云的基础架构和与服务有关的底层搭建细节。

像 CloudSim 这类的仿真软件允许用户在可重复可控的环境中免费测试其服务，在云计算商用的情况下，对基础架构的访问、使用通常需要不小的花费，因此这一点具有显著的优势。特别是对服务提供商来说，仿真环境允许服务商评估各种不同的资源租赁方案，而非依靠低效的试错来开展实验。这可以帮助服务商优化资源利用、降低成本、提高利润。

CloudSim 还有如下特点：

- (1) 支持大型云计算数据中心的建模和仿真。
- (2) 支持对虚拟服务器主机进行建模和仿真，并具有可自定义的策略，用于向虚拟机提供资源。
- (3) 支持对应用程序容器进行建模和仿真。
- (4) 支持对引入能源因素的计算资源进行建模和仿真。
- (5) 支持对数据中心的网络拓扑以及应用间的消息传递进行建模和仿真。
- (6) 支持动态插入模拟元素，实时暂停或继续模拟。

2.3.2 CloudSim 系统架构

CloudSim 中几个比较重要的类:

- (1) Cloudlet 类: 用于构建云计算平台中的任务。
- (2) DataCenter 类: 数据中心, 处理虚拟机信息的查询并提供虚拟化的资源。
- (3) VMProvisioner 类: 描述数据中心的主机与虚拟机的映射关系。
- (4) DataCenterBroker 类: 隐藏虚拟机的管理细节, 如虚拟机的创建、销毁及任务的提交等。
- (5) Host 类: 可扩展物理机器对虚拟机除处理单元 (PE) 之外的参数配置策略, 如带宽、存储空间、内存等, 一台 Host 主机中可以建立多个虚拟机。
- (6) VirtualMachine 类: 虚拟机类, 虚拟机在 Host 主机上运行, 彼此共享资源。
- (7) VMScheduler 类: 制定虚拟机的调度策略。
- (8) VMCharacteristics 类: 提供虚拟机特性描述。
- (9) VMAllocationPolicy 类: 虚拟机监视器策略类, 描述多台虚拟机在 Host 之上共享资源的策略。

在系统架构上, CloudSim 仿真软件采用分层的结构, 自底向上由 SimJava, GridSim, CloudSim, UserCode 四个层次组成, 具体结构如下几张图所示:

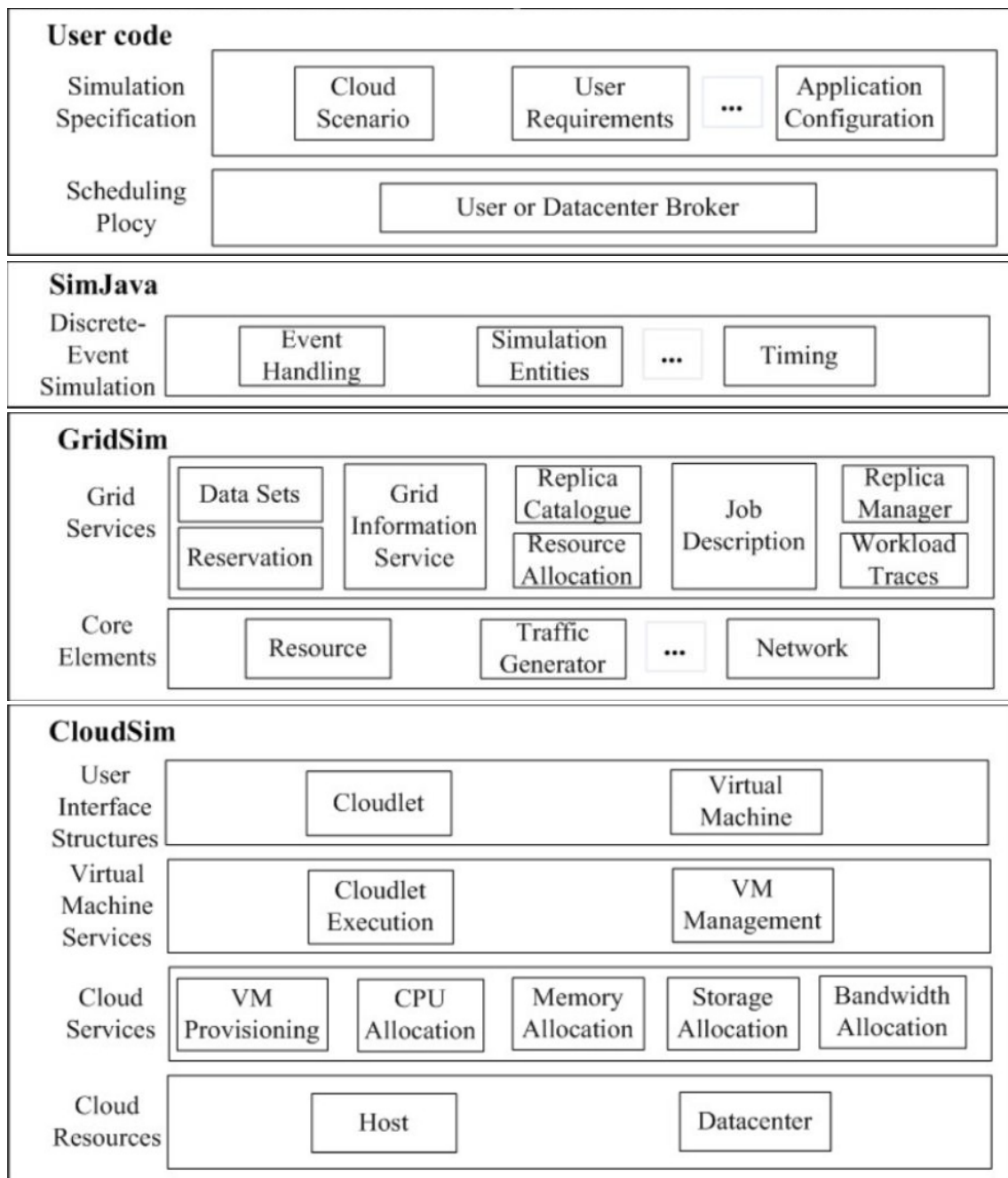


图 5 CloudSim 系统架构

SimJava: 最底层的组件 SimJava 是一个离散事件模拟引擎，它负责高层模拟框架的核心功能的执行，如：事件查询与处理，系统组件（数据中心、虚拟机等）的创建，不同组件之间的通信以及模拟时钟的管理。

GridSim: 在 SimJava 之上一层的组件是 GridSim，它支持高层组件的功能，可建模多个网格基础设施，如网络、网络流量文件等。它还提供一些基础的网络组件，如数据集、负载测量、信息服务等。

CloudSim: CloudSim 组件扩展了 GridSim 提供的核心功能，提供了基于云的

虚拟数据中心中诸如虚拟机、内存、存储和带宽等部分的管理接口，在仿真阶段管理如上的核心实体的实例与执行，这一层是 CloudSim 功能的核心。

UserCode: 这一层处在架构的最高层，是面向用户的一层。用户可以根据自己的研究内容对 CloudSim 中可扩展的部分进行自定义，制定自己的仿真测试工程。

第三章 一种优化的任务调度算法研究

3.1 常用任务调度算法

先来先服务（FCFS）算法：它为每个资源（虚拟机）设置一个等待队列，当新任务到来时，为它选择等待队列最少的资源。先来先服务算法的缺点在于其不可抢占性，排在队列后方的短任务需要等待前面的长任务完成后才开始，但该算法的周转时间和响应时间都很短^[6]。

轮询调度（Round-Robin）算法：该算法将用户提交的任务轮流分配给所有的虚拟机，从 1 开始，直到 N（所有虚拟机的个数），然后重新开始循环。这种算法的优点是其简洁性，无需记录所有虚拟机的状态，是一种无状态调度。它的缺点是当任务请求时间变化较大时，容易造成虚拟机间负载不均衡。

Min-Min 算法：该算法需要计算每个任务在各机器上的期望完成时间，从待选任务中选择长度最短（指令数少）的任务，再将其分配给期望完成时间最短的机器。这种算法的任务执行时间比较快，实现也较为简单，但其缺点在于那些较长的任务的等待时间较长^[7]，且它总是倾向于使用那些性能较强的机器（完成时间短），性能偏弱的机器就经常闲置，容易造成负载不均衡。

Max-Min 算法：该算法也需要计算每个任务在各机器上的期望完成时间，但它从待选任务中选择最长的任务，再将其分配给期望完成时间最短的机器。其缺点在于那些较短的任务的等待时间较长^[8]，同时和 Min-Min 算法相同，它也容易造成负载不均衡。

优先级调度算法：该算法包含很多种类型，总的来说都是考虑任务的优先级进行调度，在任务列表中首先根据优先级排序，再从中选择优先级最高的任务，并将其分配给期望完成时间最短的机器。其缺点在于低优先级的任务可能等待较长时间^[9]。

3.2 蚁群算法

蚁群算法（ACO: Ant Colony Optimization）：由 Dorigo 基于真实蚂蚁的行为提出^[10]。它是一种求解组合优化问题的启发式算法，其基本思想是模拟蚁群的觅食行为。当一群蚂蚁离开蚁巢寻找食物时，它们会释放一种称为信息素的特殊化学物质进行通讯，一些蚂蚁发现食物之后，其他蚂蚁可以跟随它们的踪迹而找到食物。

大多数蚂蚁倾向于选择拥有更多信息素的路径，这又使得路径上的信息素越来越多，这就是 ACO 算法的灵感来源。具体来说，初始时蚂蚁们被安置在不同地点，边 (i,j) 上的信息素初始强度 $\tau_{ij}(0)$ 也被设置好，每只蚂蚁的禁忌表（tabu list，不可选择的集合）中的第一个元素设置为其开始的地点。之后，第 k 只蚂蚁从地点 i 移动到地点 j 的概率被定义如下：

$$P_{ij}^k = \begin{cases} \frac{\tau_{ij}(t)^\alpha \cdot \eta_{ij}(t)^\beta}{\sum_{l \in allowed_k} \tau_{il}(t)^\alpha \cdot \eta_{il}(t)^\beta} & \text{if } j \in allowed_k, \\ 0 & \text{otherwise} \end{cases} \quad (3.1)$$

式 (3.1) 中， $allowed_k$ 指蚂蚁下一步允许选择的地点集合（ $allowed_k = \{N\} - \{tabu_k\}$ ）， $tabu_k$ 指的是第 k 只蚂蚁的禁忌表， $\tau_{ij}(t)$ 指的是边 (i,j) 上的信息素， $\eta_{ij}(t)$ 指的是边 (i,j) 上的启发因子，且 $\eta_{ij}(t) = \frac{1}{d_{ij}}$ ，其中 d_{ij} 为 i 到 j 的距离。 α 、 β 控制信息素与启发因子的相对重要程度。最终，算法在迭代中选择、更新最有效的路径。蚁群算法的缺点在于它没有考虑虚拟机的负载状态。

3.3 一种优化的蚁群算法

观察蚁群算法的运行流程可以发现，蚁群算法本身是针对像 TSP 一类的路径规划问题而提出的算法，它将蚂蚁随机设置在一些节点上，试图通过算法找到一条遍历所有节点的最优路径。但在云平台的任务调度问题中，算法所处理的节点不只是一类，而是包括任务和虚拟机这两种节点。同时，蚁群算法中也没有考虑到在任务调度问题里如何设定相关的启发因子，以及如何将虚拟机负载这个因素加入算法中。基于此，本节提出一种针对云平台任务调度的负载均衡蚁群算法，主要适用于任务和虚拟机间的分配调度。

为了对原蚁群算法做出相应的改进，算法主要在以下几个方向上进行研究，探寻具体的优化实现：

- (1) 考虑调度过程中的任务负载均衡情况，设定负载均衡因子，对算法的启发式函数进行优化。
- (2) 改进算法的信息素更新规则，使之更适合云平台调度问题，避免陷入局部最优。
- (3) 改进算法的计算方式，优化计算时间。

算法首先计算每个请求任务的预期完成时间以作备用，将任务的预期完成时间整合可得到虚拟机的预期完成时间。接着对每个任务，依据重新设置的计算式（包括完成时间等其他一些因子）计算得出选择每个虚拟机的概率（其和为 1）。每只蚂蚁依据概率来对虚拟机进行投票，最终选择有最高票数的虚拟机。每一轮过后，对预期完成时间和蚁群使用的信息素进行更新。因为计算式中包含了除完成时间以外的其他一些影响因子，因此有机会跳出局部最优。

算法详细流程如图 6:

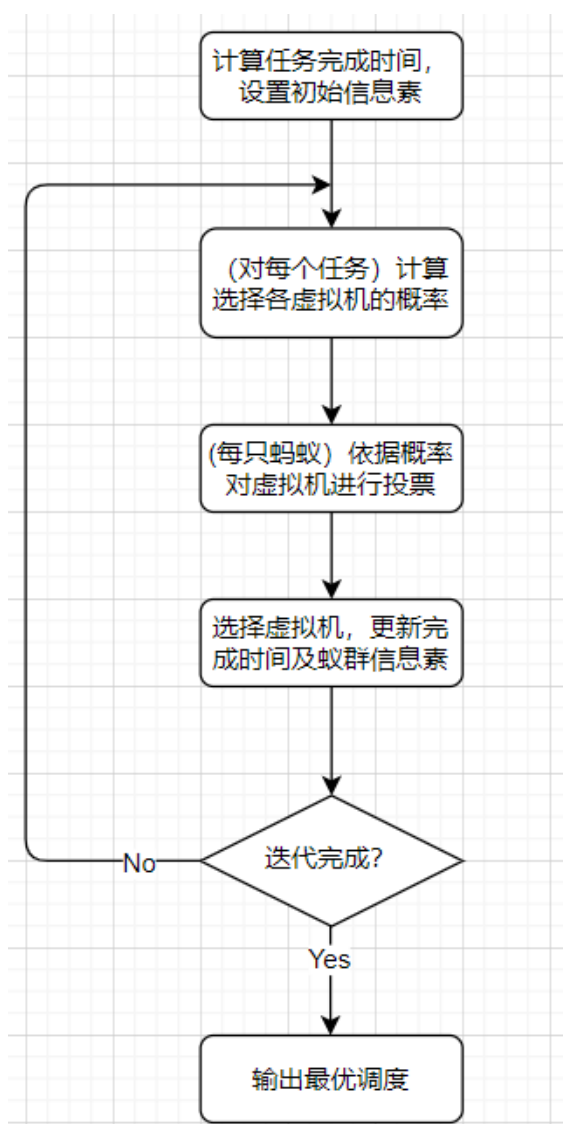


图 6 优化算法流程

算法主要包括初始化，选择虚拟机和更新信息素三大步骤，其中在初始化阶段中，需要对蚁群信息素进行初始化：

$$\tau_j(0) = CC_j。 \quad (3.2)$$

其中， CC_j 衡量虚拟机 VM_j 的计算能力，计算式如下：

$$CC_j = pe_num_j \cdot pe_mips_j + vm_bw_j。 \quad (3.3)$$

式（3.3）中 pe_num_j 是虚拟机 VM_j 的处理元的数量， pe_mips_j 是虚拟机 VM_j 的处理元的处理速度（百万指令每秒）， vm_bw_j 是虚拟机 VM_j 的通信带宽能力。

在选择虚拟机的阶段中，每只蚂蚁都要计算概率值以提供决策，计算式包括多个计算因子，首先是任务 n_i 在虚拟机 VM_j 中的预期完成时间 $ET_j(i)$ ：

$$ET_j(i) = \frac{task_length_i}{pe_num_j \cdot pe_mips_j}。 \quad (3.4)$$

其中， $task_length_i$ 是任务 n_i 的长度（按指令条数）。接着根据分配给虚拟机 VM_j 的任务的 $ET_j(i)$ 计算得到虚拟机 VM_j 的预期完成时间 $EET_j(t)$ ：

$$EET_j(t) = \sum_{n_i \in VM_j} ET_j(i) - R_j(t)。 \quad (3.5)$$

式（3.5）中， $R_j(t)$ 是虚拟机 VM_j 中当前正在运行的任务的剩余处理时间，是根据任务的提交时间，当前时间以及任务执行情况得出，如下图所示：

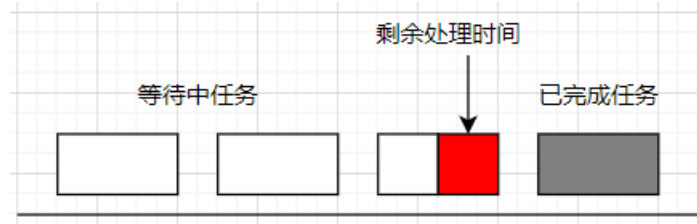


图 7 剩余处理时间

计算式中还需考虑虚拟机负载因素以改进原始蚁群算法在这方面的不足，虚拟机 VM_j 在时间 t 时的负载因子 $LB_j(t)$ 由式（3.6）计算得出，负载因子是系统的总负载与虚拟机 VM_j 的负载之比，其值越高表示该虚拟机负载越小。

$$LB_j(t) = \frac{\sum_{VM_k} EET_k(t)}{EET_j(t)}。 \quad (3.6)$$

最终，蚂蚁 Ant_k 选择虚拟机 VM_j 的概率计算式如下：

$$P_j^k(t) = \frac{\tau_j(t)^\alpha \cdot CC_j^\beta \cdot LB_j(t)^\gamma}{\sum_k \tau_k(t)^\alpha \cdot CC_k^\beta \cdot LB_k(t)^\gamma} \quad (3.7)$$

上式中， α ， β ， γ 分别控制了信息素，计算能力，负载因素这三个计算因子的影响程度大小。同时可以看到，上述计算式都是以虚拟机 VM_j 为计算中心，而在传统的蚁群算法中，计算式需要以 (i, j) 这个对应关系作为计算中心（在任务调度中就是任务与虚拟机的对应），这样一来，优化后的计算式的时间复杂度就小了许多，能够减少计算时间。

计算完所有虚拟机的选择概率后，每只蚂蚁都会根据概率值向一个虚拟机进行投票。为了防止陷入局部最优，这里结合随机的方法进行投票，投票方法流程图如下图所示：

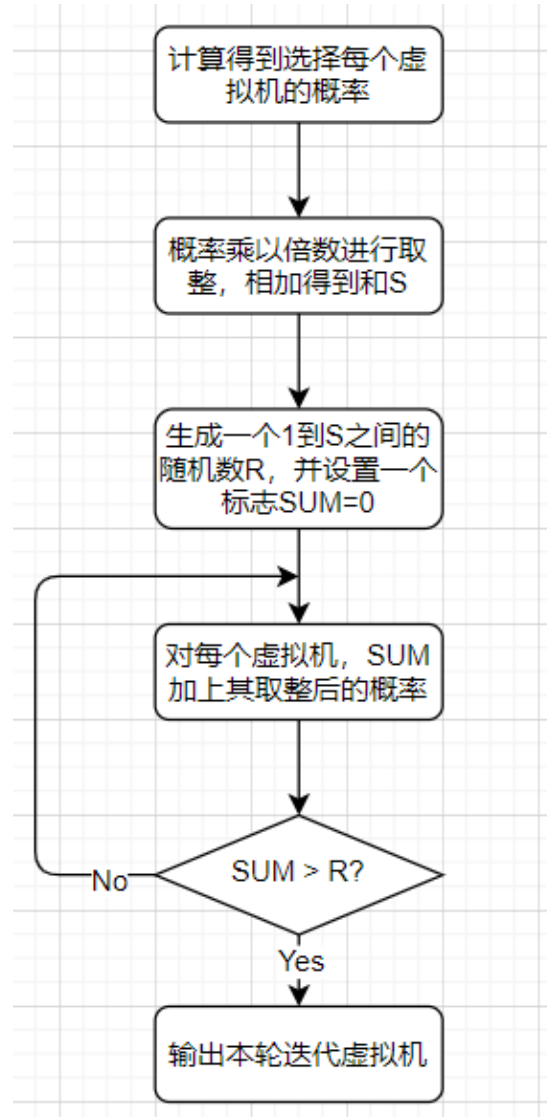


图 8 投票方法流程

在更新信息素阶段，算法根据本轮所选择的虚拟机进行更新：

$$\tau_j(t+1) = \tau_j(t)(1-\rho) + \begin{cases} \frac{1}{T(t)} & \text{if } VM_j \text{ is selected} \\ 0 & \text{otherwise} \end{cases} \quad (3.8)$$

其中， ρ 为挥发因子，代表信息素随时间而衰减的比例。 $T(t)$ 为本轮任务在所选择的虚拟机上的预期完成时间。

第四章 仿真实验与结果分析

第三章对本文提出的基于蚁群的优化任务调度算法进行了详细的设计和实现，本章将使用 CloudSim 仿真软件搭建一个虚拟环境，并通过几种算法间的对比试验来对本文提出的优化算法进行评测，进行评测前首先进行相关设置。

4.1 实验配置

本实验使用一台 MSI GE62 笔记本电脑来运行 CloudSim 并构建仿真虚拟环境，硬件配置如下表所示：

表 1 实验环境硬件配置

类型	配置
操作系统	Windows 10
处理器	Inter(R) Core(TM) i7-6700HQ CPU @ 2.60GHz
内存	16G
磁盘	1TB

于 CloudSim 3.0.3 平台上开展仿真实验，CloudSim 具体设置如下表所示：

表 2 CloudSim 具体设置

类型	配置	设定值
云任务	任务长度(MI)	100-1000
	任务数量	50-500
虚拟机	虚拟机数	20
	虚拟机 MIPS	500-1000
	虚拟机内存(MB)	512
	虚拟机带宽(Mb/s)	500-1000
	计算核心(PE)数	1
数据中心	数据中心数	4
	虚拟机调度器	Space_shared

4.2 参数配置

优化算法中对算法结果有影响的参数包括 α : 信息素权重, β : 启发因子权重, γ : 负载因子权重, m : 蚂蚁数量, ρ : 信息素挥发因子, 以及 t_{max} : 迭代次数。这里将设计对比实验确定参数的设定, 当对一个参数进行测试时其他参数保持不变, 总任务数量设置在 500。

参数的默认值为: $\alpha = 0.8$, $\beta = 1.0$, $\gamma = 3.0$, $\rho = 0.4$, $m = 30$, $t_{max} = 100$ 。实验过程中, 各参数的取值范围是: $\alpha \in \{0, 0.2, 0.4, 0.6, 0.8\}$, $\beta \in \{0, 0.5, 1.0, 1.5, 2.0\}$, $\gamma \in \{1.0, 2.0, 3.0, 4.0, 5.0\}$, $\rho \in \{0, 0.1, 0.2, 0.3, 0.4\}$, $m \in \{25, 30, 35, 40, 45\}$, $t_{max} \in \{50, 75, 100, 125, 150\}$ 。实验使用任务完成时间作为评判标准。

首先是参数 m , 如下图所示, m 可取 35。

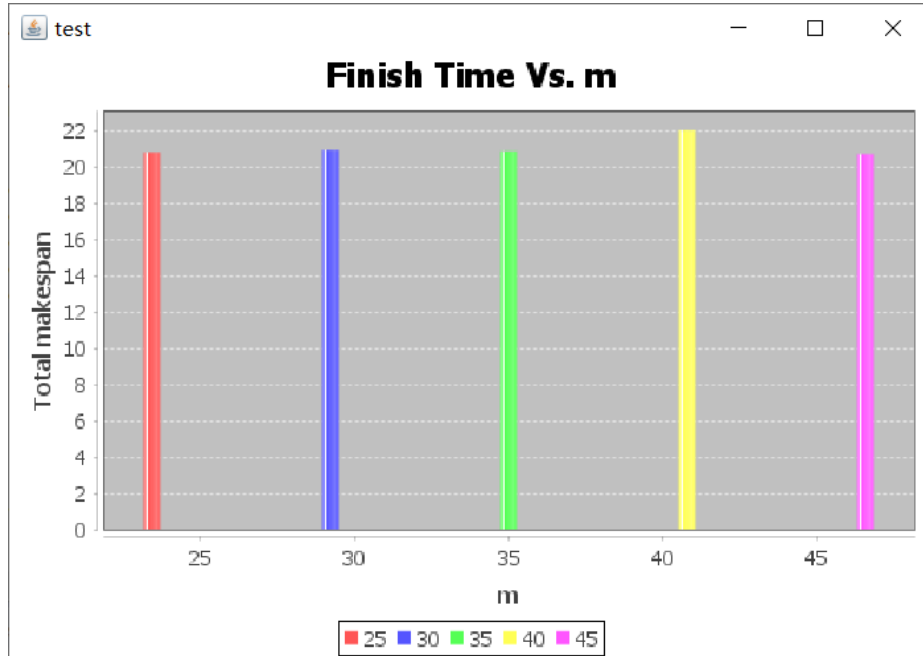


图9 m 取值

然后是参数 α , 如下图所示, α 可取 0.2。

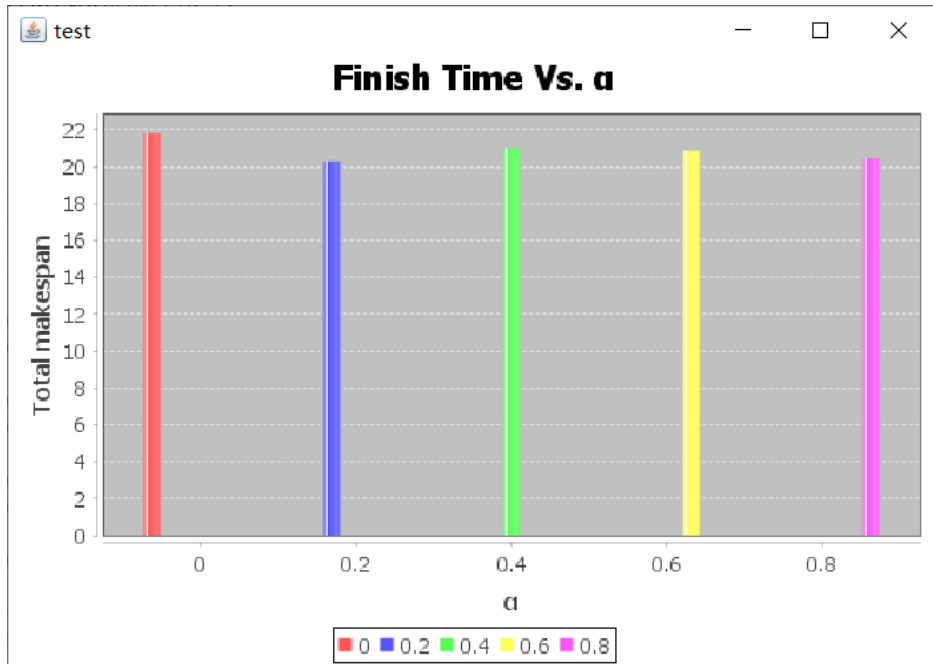


图 10 α 取值

然后是参数 β ，如下图所示， β 可取 1.0。

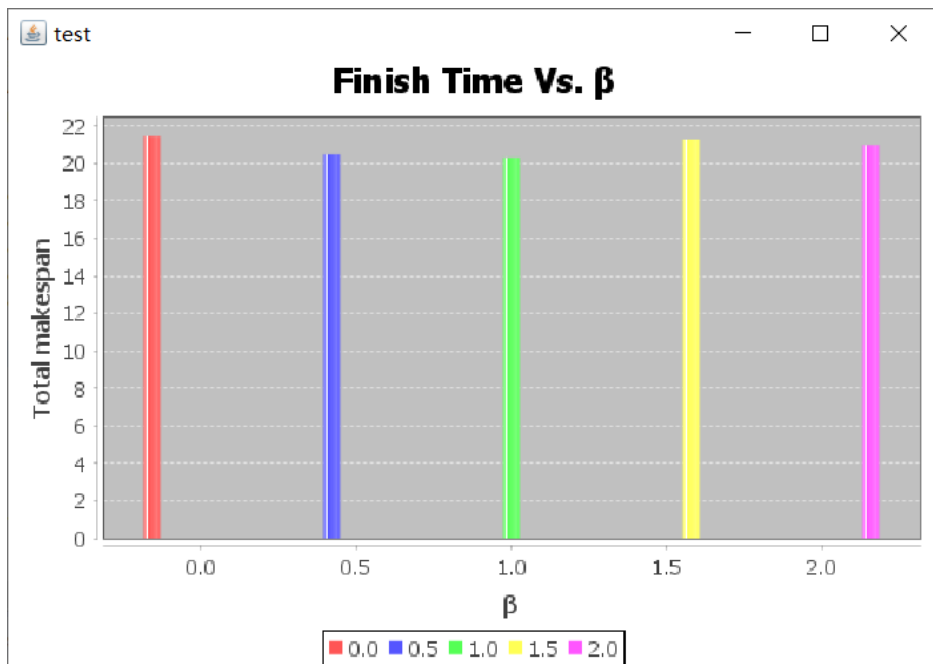


图 11 β 取值

然后是参数 γ ，如下图所示， γ 可取 5.0。

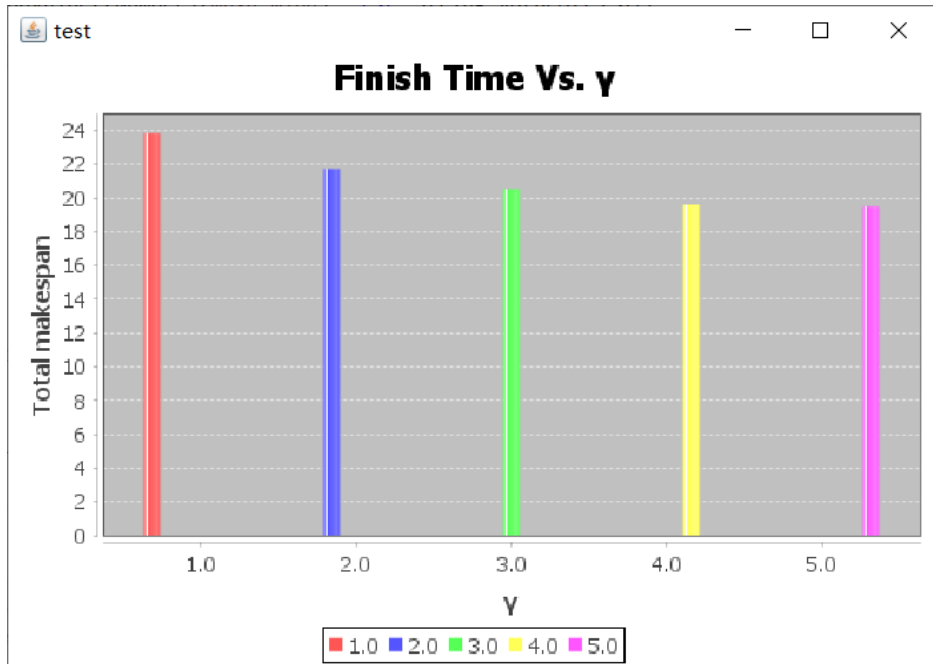


图 12 γ 取值

然后是参数 ρ ，如下图所示， ρ 可取 0.2。

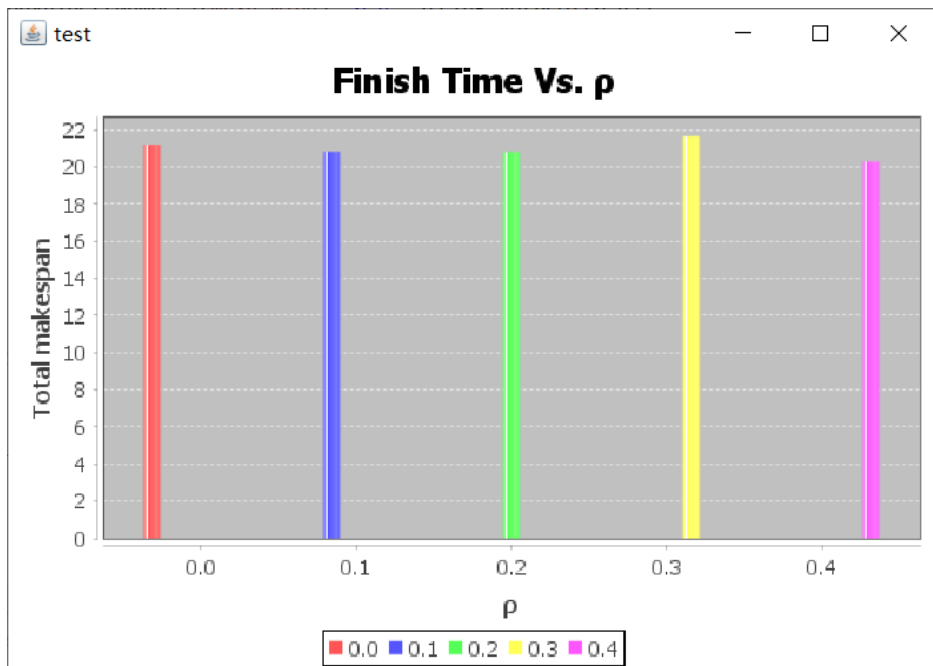


图 13 ρ 取值

然后是参数 t_{max} ，如下图所示， t_{max} 可取 150。

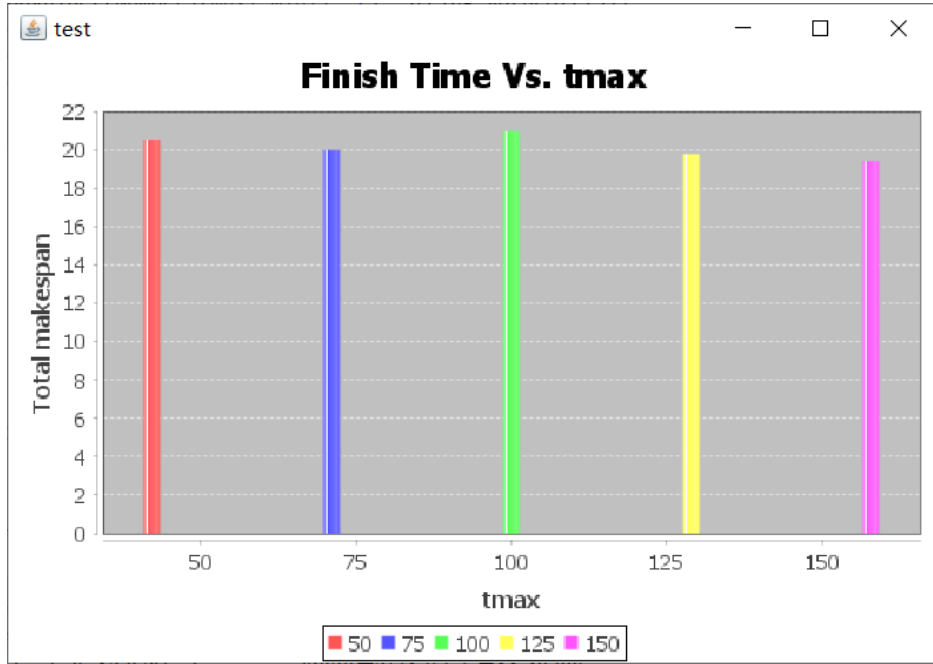


图 14 t_{max} 取值

综上所述,在下一实验中优化算法的各参数配置为: $\alpha = 0.2, \beta = 1.0, \gamma = 5.0, \rho = 0.2, m = 35, t_{max} = 150$ 。

4.3 实验结果

本文所述优化算法主要希望在执行效果和负载均衡方面对原算法做出改进,因此本文将设置两方面的对比试验,将轮询调度算法(RR)、蚁群算法和优化后的蚁群任务调度算法进行对比,其中原始蚁群算法依照 Tawfeek 等人的论文^[11]进行实现。第一项实验是比较不同算法的任务完成时间(从第一个任务开始到最后一个任务完成),任务完成时间从结果上反映了任务调度算法的好坏,是对任务调度算法的基本性能进行评测,结果如下图:

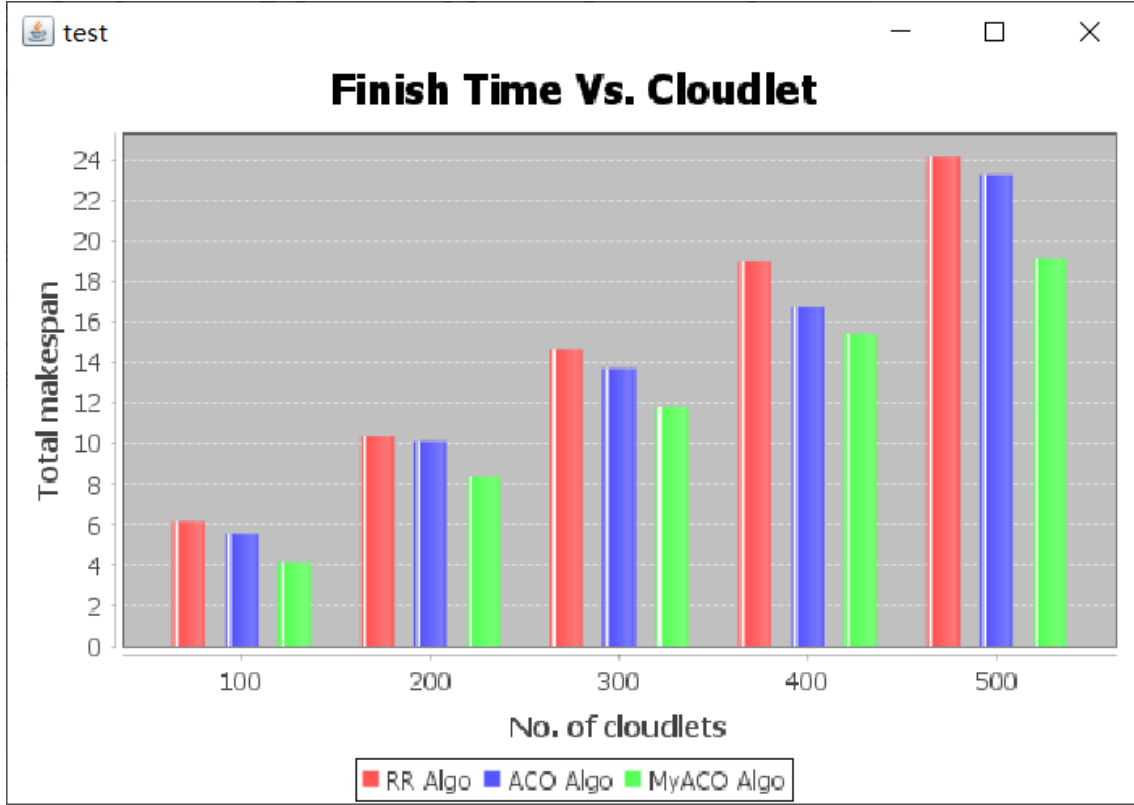


图 15 任务完成时间

可以看出，轮询算法在任务完成时间这一项上的表现最差，任务完成时间最长。蚁群算法的表现要更好一些，而优化后的蚁群任务调度算法在三种算法中是最优的，一直保持了较短的任务完成时间。

第二项实验是比较不同算法运行时的虚拟机负载均衡程度，首先需要定义衡量标准即不平衡度（DI: Degree of Imbalance），根据 Tawfeek 所述^[11]，可由以下两式计算而得：

$$T_j = \frac{Tasks_length}{pe_num_j \cdot pe_mips_j} \quad (4.1)$$

其中， $Tasks_length$ 指的是提交给该虚拟机 VM_j 的所有任务的总长度。

$$DI = \frac{T_{max} - T_{min}}{T_{avg}} \quad (4.2)$$

其中， T_{max} ， T_{min} ， T_{avg} 分别为所有虚拟机的 T_j 中的最大值，最小值，平均值。

实验结果如下图：

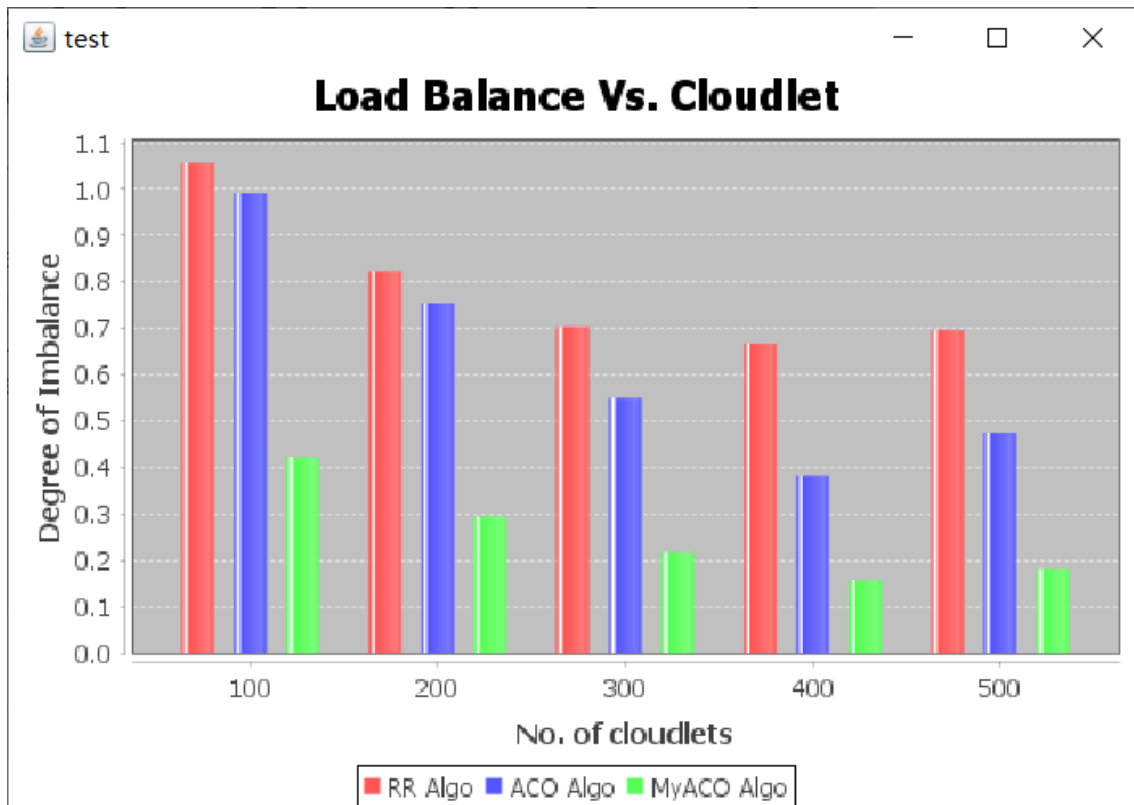


图 16 负载不均衡度

可以看出，任务数为 100 时，虚拟机无法被完全均衡覆盖，导致一开始的负载不均衡度都较高，其中，轮询算法与蚁群算法的负载不均衡度更高，优化后的蚁群算法的负载则一直保持在一个较为均衡的水平。

由以上两个实验可看出，本文优化的蚁群算法在任务完成时间和资源负载均衡程度上均表现较好，能够在提高算法执行效率的同时在一定程度上缓解调度的负载不均衡程度，是对蚁群算法的一个合理、良好的改进。

第五章 总结与展望

5.1 总结

经过日新月异的技术换代，云计算技术已成为一项新兴的、热门的 IT 技术，它提供的方便快捷、按需自取的商业服务模式与传统计算服务模式相比有很大的优越性。在云计算的各项技术中，任务调度控制了任务在虚拟机资源间的分配调度，决定着任务的完成时间与虚拟机资源的负载均衡程度，是云计算提供的计算服务所以能够如此便捷的一大因素，对用户体验好坏有着很大影响。

本文首先对云计算、任务调度、仿真实验工具等一些研究相关知识进行了介绍，接着对云计算背景下的任务调度算法进行了深入的分析、研究，如轮询算法、Min-Min 算法、蚁群算法等。为了提高云计算平台下的用户体验，笔者试图从蚁群算法在云平台任务调度问题中的信息素配置与虚拟机负载入手，对蚁群算法进行改进，以得到一种优化的蚁群算法，提高算法的完成时间与负载均衡度。最后本文开展了两方面的对比实验，一定程度上证明了本文中优化算法的可用性。

5.2 优缺点

本文实现的优化算法相比原始算法有如下优点：

1. 算法考虑了在云平台进行任务调度时虚拟机的负载均衡因素，在计算概率函数时引入负载因子，实验结果显示算法的负载均衡度确实更优。
2. 算法改进了信息素的配置与更新规则，避免陷入局部最优解，实验结果显示算法的任务完成时间相比原算法更优。
3. 原始算法每次迭代时，蚂蚁都需分配所有任务，改进之后算法以任务为核心，每次蚂蚁对一个任务以投票的形式进行虚拟机选择，大大改进了算法计算时间。

然而本算法设计仍有不足之处，如：

1. 相比别的一些优化算法，本算法在信息素的更新方面仍可以进一步优化。
2. 实验结果具有一定偶然性，可以做多次实验进行平均。
3. 算法未对鲁棒性进行测试，在异常情况下的表现尚未可知。

5.3 展望

在本文的工作之外，学术界中的研究者也早已对任务调度算法进行了许多研究，对于蚁群算法也做出了不少改进。如将蚁群算法与其他启发式算法（模拟退火算法，遗传算法^[12]，粒子群算法^[13]等）结合形成混合式算法，在任务完成时间与负载均衡度上的表现都不错。本文算法也可在这一方面做进一步的优化。

除此之外，本文仅对任务调度算法进行了理论分析，对算法所做的对比试验也是在仿真软件中进行，算法究竟优劣与否还需通过真实的云计算平台来进行测试，如何考虑真实情况对算法进行改进是未来的优化方向。

参考文献:

- [1]. Calderon, Alejandro and Garcia-Carballeira, Felix and Bergua, Borja and Sanchez, Luis Miguel and Carretero, Jesus. Expanding the volunteer computing scenario: A novel approach to use parallel applications on volunteer computing[J]. Future Generation Computer systems, 2012, 28(6):881-889.
- [2]. Syed, Raheel Hassan and Pazardzievska, Jasmina and Bourgeois, Julien. Fast attack detection using correlation and summarizing of security alerts in grid computing networks[J]. The Journal of Supercomputing, 2012, 62(2):804-827.
- [3]. Borgdorff, Joris and Falcone, Jean-Luc and Lorenz, Eric and Bona-Casas, Carles and Chopard, Bastien and Hoekstra, Alfons G. Foundations of distributed multiscale computing: Formalization, specification, and analysis[J]. Journal of Parallel and Distributed Computing, 2013, 73(4):465-483.
- [4]. Mell, Peter and Grance, Tim and others. The NIST Definition of Cloud Computing[R]. National Institute of Standards and Technology: U.S. Department of Commerce. 2011.
- [5]. Calheiros, Rodrigo N and Ranjan, Rajav and Beloglazov, Anton and De Rose, Cesar AF and Buyya, Rajkumar. CloudSim: A Toolkit for Modeling and Simulation of Cloud Computing Environments and Evaluation of Resource Provisioning Algorithms[J]. Software: Practice and Experience (SPE), 2011, 41(1):23-50.
- [6]. Agarwal, Dr and Jain, Saloni and others. Efficient Optimal Algorithm of task scheduling in Cloud computing Environment[J]. International Journal of Computer Trends and Technology(IJCTT), 2014, 9(7).
- [7]. Kobra Etmnani. A Min-Min Max-Min Selective Algorithm of task scheduling in Cloud computing Environment[A]. 3rd IEEE/IFIP International Conference in Central Asia on Internet 2007[C], Iran, 2007.
- [8]. Pawar, Chandrashekhar S and Wagh, Rajnikant B. Priority based Dynamic Resource Allocation in Cloud Computing[J]. International Symposium on Cloud and Service Computing(ISCOS), 2012.
- [9]. Pinal Salot. A survey of various scheduling algorithm in cloud computing environment[J]. International Journal of Research in Engineering and Technology, 2013, 2(2):131-135.

- [10]. Dorigo, Marco and Di Caro, Gianni. Ant Colony optimization: a new meta-heuristic[A].
Proceedings of the 1999 Congress on Evolutionary Computation-CEC99[C], United States:
IEEE, 1999:1470-1477.
- [11]. Tawfeek, Medhat A and El-Sisi, Ashraf and Keshk, Arabi E and Torkey, Fawzy A. Cloud task
scheduling based on ant colony optimization[A]. 2013 8th International Conference on
Computer Engineering & Systems (ICCES) [C], Egypt: IEEE, 2013.
- [12]. 刘继梅. 基于改进遗传-蚁群算法的云计算资源调度研究[D]. 延安大学, 2018.
- [13]. 尹亚日. 基于改进粒子群和蚁群的云计算任务调度研究[D]. 南京邮电大学, 2019.

致谢

在这里，首先我要向我的母校中山大学致以诚挚的感谢，感谢这四年来学校的栽培，老师的教导，教我编程，教我原理，教我实操，将一个踏出高三的懵懂少年引入计算机的世界中，帮助我建立一座基础知识的高塔。

感谢温武少老师。温老师学识渊博、治学严谨、工作精益求精，是我学习的榜样。在计算机网络课程上，温老师的授课生动、有趣而又易懂。在毕业设计的完成过程中，温老师也给我以意见、帮助。

感谢我的同学们。四年大学生活中，他们在学习上和生活上都给予我莫大的支持，愿大家都前途似锦、万事顺遂。

最后我要感谢我的家人，无论我遇到什么困难，你们都竭尽全力地帮助我，不管我做出什么选择，你们都坚定不移地支持我。正是由于你们无私的付出和关爱，今天我才得以成为中山大学的毕业生。祝愿你们永远健康快乐。

愿我的母校再创辉煌，愿我的老师同学万事如意，愿我的家人永葆健康，愿我的祖国繁荣昌盛！

毕业论文（设计）成绩评定记录

Grading Sheet of the Graduation Thesis (Design)

指导教师评语 Comments of Supervisor:


论文针对分布式云系统下的资源调度需求,采用了一种改进蚁群算法进行以优化算法的完成时间、负载均衡度和计算时间。

论文来源于工程实践,选题有一定意义。论文总体结构清楚,写作规范,有一定工作量,体现作者有一定的软件工程基础、工程问题分析、解决和实现能力。

论文已经达到学生专业毕业论文要求。

成绩评定 Grade:

良好



指导教师签名 Supervisor Signature:

Date: 5/13/2020

答辩小组意见 Comments of the Defense Committee:

成绩评定 Grade:

签名 Signatures of Committee Members:

Date:

院系负责人意见 Comments of the Academic Chief of School:

成绩评定 Grade:

签名 Signature:

院系盖章 Stamp:

Date:



中山大學
SUN YAT-SEN UNIVERSITY