分 类 号：　　密级：
论文编号：2020022165

贵　州　大　学

2023 届硕士研究生学位论文

RPA 云服务系统的设计与实现

学科专业：软件工程

研究方向：不区分研究方向

导　　师：陈梅

研 究 生：陈睿

中国 ▪ 贵州 ▪ 贵阳

2023 年 06 月

硕士学位论文答辩委员会名单

RPA 云服务系统的设计与实现

答辩人：陈睿

答辩委员会委员：

贵州大学 教授:王以松（主席）

贵州大学 副教授:杨朝树

贵州大学 副教授:王崎

贵州大学 副教授:叶晨

贵州师范大学 教授:刘彬

答辩秘书：罗媛 讲师/硕士

答辩时间：2023 年 6 月 1 日

答辩地点：博学楼 610

摘要

办公自动化技术是当下企业降本增效中不可或缺的一部分，随着办公自动化的发展，该技术逐渐演变为 RPA(Robotic Process Automation)技术。RPA 是一种将繁琐且重复的用户操作自动化执行的软件，它可以模拟人工操作，有效帮助用户减少重复性工作。在 Gartner 预测报告中，RPA 的市场占有率正在快速上升。伴随着 RPA 商业公司对云服务的不断完善，RPA 与云服务的结合是必然趋势，本文将对 RPA 云服务系统与云任务调度两方面内容展开研究。

针对 RPA 云服务系统的设计，本文在网页自动化场景下，利用 Selenium 引擎为 RPA 核心功能提供支撑，使用 Kubernetes 集群作为容器云技术设施，实现了一个 RPA 云服务系统。在 RPA 云服务系统基础上，本文对五大领域常见的工作流程设计了 23 个任务场景，并设计了一个 Micro-Benchmark，通过实际测试，得出了 RPA 云任务负载不均衡的结论。因此，为了保证云服务的服务质量，本文使用一种改进粒子群 SN-PSO(Based on Sliding Window and Non-Dominant Pareto PSO)算法对 RPA 任务进行调度优化，通过实际并发测试，验证了 SN-PSO 在并发环境下任务调度优化的有效性。本文的总体工作内容分为以下三点：

（1）设计了一种基于网页自动化的 RPA 云服务系统，通过组合功能组件的方式实现用户自主按需设计任务流程，利用无人值守方案提供了用户任务以云端执行的模式。针对 RPA 在实际场景下的应用，本文针对制造业、财务、人力资源、物流及在线教育五大领域设计了共计 23 个任务集，在此任务集基础上，设计了一个 Micro-Benchmark，为 RPA 任务调度优化提供数据支撑。

（2）提出了一种基于滑动窗口与非支配解集的改进粒子群算法(Based on Sliding Window and Non-Dominant Pareto PSO, SN-PSO)。SN-PSO 通过滑动窗口检测出陷入局部最优的粒子，并调整粒子位置中贡献较低的维度。该方案避免了随机变异导致的不确定性，减少了粒子的无效搜索行为。通过非支配解集的优化策略提高了种群的多样性，增强了粒子全局寻优能力。利用 SN-PSO 对 RPA 云任务调度作优化。

（3）实现了 RPA 云服务系统并针对主要工作进行实验评估。针对 RPA 云服务系统的实现做了功能测试，且为了改善 RPA 任务负载不均衡问题，对 RPA 云服务系统中使用的改进粒子群算法进行了实验评估与分析，实验表明在实际使用场景下，SN-PSO 能够有效地降低任务总执行时间，提高系统的运行效率。

关键词： RPA；云服务；任务调度；粒子群算法

Abstract

Office Automation technology is an indispensable part of the current enterprise to reduce costs and increase efficiency. With the development of office automation, this technology has gradually evolved into RPA(Robotic Process Automation) technology. RPA is a software that automates the execution of tedious and repetitive user operations. It can simulate manual operations and effectively help users reduce repetitive work. In Gartner's forecast report, RPA market share is increasing rapidly. With the continuous improvement of cloud services by RPA commercial companies, the combination of RPA and cloud services is an inevitable trend. This paper will study two aspects of RPA cloud service system and cloud task scheduling.

Aiming at the design of RPA cloud service system, in the scenario of web automation, this paper uses Selenium engine to provide support for RPA core functions, and uses Kubernetes cluster as a container cloud technical facility to realize an RPA cloud service system. Based on the RPA cloud service system, this paper designs 23 task scenarios for the common workflows in five fields, and designs a Micro-Benchmark. Through actual testing, it comes to the conclusion that the load of RPA cloud tasks is unbalanced. Therefore, in order to ensure the quality of service of cloud services, this paper uses an improved particle swarm SN-PSO(Based on Sliding Window and Non-Dominant Pareto PSO) algorithm to optimize the RPA task scheduling. It is proved that SN-PSO is effective in task scheduling optimization in concurrent environment. The overall work of this paper is divided into the following three points:

(1) An RPA cloud service system based on web automation is designed, which realizes the user's independent and on-demand design task process by combining functional components, and provides a mode for user tasks to be executed in the cloud by using unmanned scheme. Aiming at the application of RPA in actual scenarios, this paper designs a total of 23 task sets for manufacturing, finance, human resources, logistics and online education. Based on this task set, a Micro-Benchmark is designed to provide data support for RPA task scheduling optimization.

(2) An improved particle swarm optimization Based on Sliding Window and Non-Dominant Pareto PSO (SN-PSO) is proposed. SN-PSO detects particles trapped in local optima through a sliding window and adjusts the dimensions with lower contributions in the particle positions. This scheme avoids the uncertainty caused by random mutation

and reduces the invalid search behavior of particles. Through the optimization strategy of non-dominated solution set, the diversity of population is improved and the global optimization ability of particles is enhanced. SN-PSO is used to optimize RPA cloud task scheduling.

(3) The RPA cloud service system is implemented and the experimental evaluation is carried out for the main work. A functional test was done for the implementation of RPA cloud service system, and the experimental evaluation and analysis of the improved particle swarm optimization algorithm used in RPA cloud service system were carried out. The experiment shows that in actual use scenarios, SN-PSO can effectively reduce the total execution time of concurrent tasks and improve the operating efficiency of the system.

Key words: RPA；Cloud Service；Task Scheduling；Particle Swarm Optimization

目录

第1章　绪论

1.1 研究背景和意义

早在上世纪 80 年代，办公自动化就在替代效率低下、出错率较高的人工作业方式。当时的办公自动化技术主要由传真、高级用户电报、电话等终端设备与计算机相结合的通信系统组成[1]。20 世纪 90 年代到 21 世纪初期，办公自动化则以互联网技术和协同办公为主要特征[2]。但是，传统的办公自动化技术通常提高的是部门内部的工作效率，对于多部门之间的数据同步、协同办公等问题，并没有得到很好的解决。此外，由于公司规模的快速扩张，员工繁杂且内容重复的工作大大增加了易错率，降低了工作效率，因此为了解决上述问题，办公自动化技术逐渐演变成机器人流程自动化技术，即RPA(Robotic Process Automation)。

RPA是一种将繁琐且重复的用户操作自动化执行的软件，它可以模拟人工操作，有效帮助用户减少重复性工作[3][4]。IEEE(电气电子工程师学会)对机器人流程自动化进行了如下的定义：RPA是一种预先配置好的软件，它使用业务规则和预定义的活动编排来在一个或多个不相关的软件系统中自动完成流程、活动、事务和任务的组合，最后提供相关的处理结果[5]。以采购过程为例，每当员工需要执行采购操作时，只需上传相关Excel表单至系统，此时RPA将根据扫描的商品名称、商品数量等信息与采购计划中的商品信息进行比对，并将采购信息传达至制造部等相关部门，完成采购操作。相比传统的员工核查方式，RPA能够有效避免核对数据的误差。相关采购任务流程如图 1-1 所示。

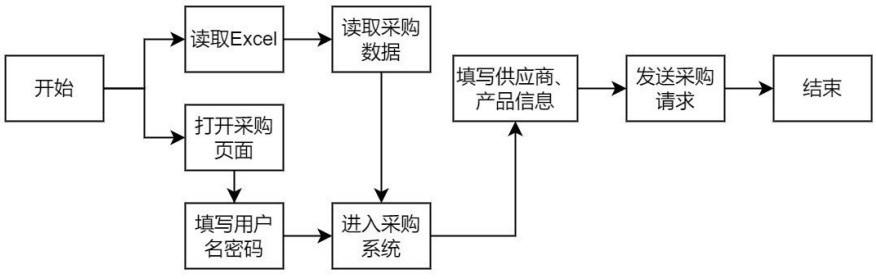


图 1-1 采购任务示例图

在《Gartner Top 10 Strategic Technology Trends For 2020》中提到，自动化技术使用了一系列复杂的工具和技术，但是核心组件是RPA和智能业务流程管理套件(iBPMS)。并且到2022年，机器人流程自动化交付的应用程序将同比增长40%[6]。

根据图 1-2 中的IDC预测数据，到 2023 年，全球RPA软件市场规模将达到 39 亿美元，2018—2023 年复合增长率将达到 36%。2023 年，中国RPA市场规模将达到 10.18 亿美元[7]。因此，RPA将会是未来技术的新浪潮[8]，并且RPA系统能够帮助企业降低成本[9][10]，在许多领域的应用场景下都将有着重大的应用价值和参考意义。

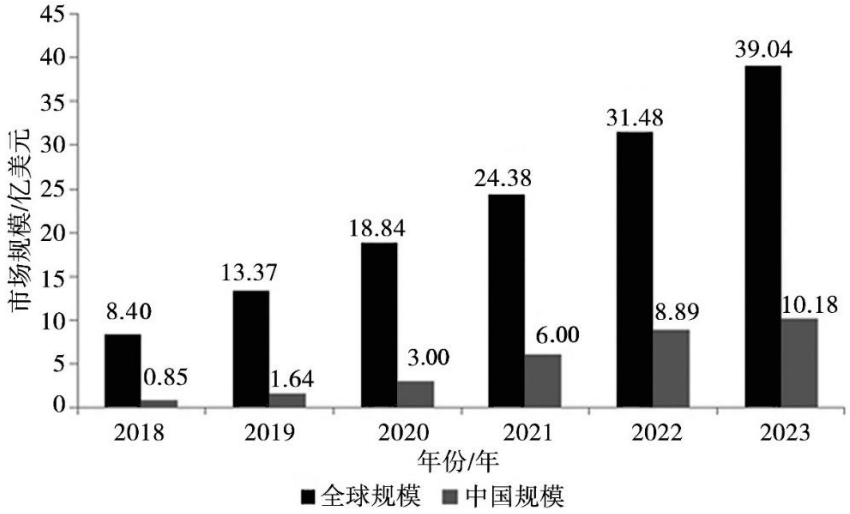


图 1-2 2018—2023 年全球及中国 RPA 软件市场规模

目前RPA常用于金融、财务等部分场景，理论上来说，RPA可以轻松应用于业务组织中的几乎所有领域[11]。例如，RPA在业务流程外包方面的应用[12]。然而，针对RPA技术的研究非常稀少，并且当下的RPA相关技术研究集中在利用用户操作日志挖掘自动化流程方面[13][14]，而不是RPA技术本身。在RPA其他的相关研究中，大多数文献内容集中在：RPA的使用优势、RPA的应用以及对RPA的定义这几个部分[3]。尽管RPA相关研究及文献正在快速增加[15][16]，但是关于RPA技术细节的研究是极为稀缺的。

在RPA商业软件领域，目前涌现出越来越多不同的RPA产品[17]。本文列举了一部分知名RPA系统，如表 1-1 所示。

表 1-1 中，部分知名RPA软件大多以桌面端的形式提供给用户，用户需要在本地对任务进行编排，而后再选择本地运行或云端运行的模式，这种方式需要用户手动安装软件，增加了软件的使用门槛。此外，随着云需求的不断增长和普及，大多数软件以云服务形式提供给用户，并以云服务中的软件即服务(SaaS)方式交付[18]，大量科技公司正试图抢占这个新兴市场[19]。目前大型的RPA商业软件对云技术的支持越来越完善，这意味着RPA上云是一个必然趋势。

表 1-1 部分知名 RPA 系统表

产品	优点	缺点
影刀 RPA	功能完备，易用，拥有 GUI	闭源商业软件，不以云服务形式提供
UiPath	拥有一套完整的生态体系，拥有 GUI	闭源商业软件，需要使用生态体系内的服务，否则无法获得最佳体验
TagUI	可以用编程的方式设计 RPA 流程，且功能较为完备，开源	只能通过命令行的方式执行 RPA 任务计划，对用户不友好无云服务形式
Robot Framework	拥有 GUI，用户群体活跃	不以云服务形式提供并需要独立安装

就目前RPA的发展而言，RPA从常见的金融领域开始向工业、物流等多种领域展开应用，由于用户任务执行方式具有多样性，并且不同领域下的任务特性有所不同，这将极大地考验RPA云服务提供商的能力。在用户任务以云端执行模式的场景下，云环境中的各种资源管理问题将会成为需要解决的问题之一[20]，若用户任务无法以一种合理的策略或方式进行分配，使得资源不能够充分利用，将极大的降低RPA云提供商的服务质量，增加运行成本。而云任务调度算法能够有效的为云环境下的服务质量(QoS)提供保障。

综上，实现一个RPA云服务系统，并针对RPA云任务负载特性进行任务调度优化是非常有意义的。

1.2 国内外研究现状

1.2.1 RPA 研究现状

如前文所述，RPA目前的研究较少，并且缺少技术性研究，大多数文献只描述了RPA的理论基础或在特定场景中实施 RPA 后的产业成果，尤其是在金融领域[错误!未找到引用源。]。一些作者将 RPA （机器人流程自动化）概念与 BPM（业务流程管理）战略联系起来，作为提高组织竞争力及其生产力的机制[22]。Lacity指出，RPA 在工业中实施一直较为缓慢[23]，仅在部分领域中使用RPA的频率较高。例如，在计算机测试领域，张红[24]提出了一种自动化测试流程的生成方法，成功实现了在不同业务流程场景下生成自动化流程，但是如作者所言，该方法尚未在大规模的关系图或关系树上进行实践，可能在复杂关系树中，DFS方法会失效。

在RPA应用方面，沈港[1]实现了一个基于RPA的自动化办公系统，论文中详细描述了关于化妆品零售公司机器人流程自动化系统的实现，但是在实现的这个RPA系统中存在一个重大缺陷：该RPA技术采用绝对位置的方式获取GUI元素，

对于不同分辨率的场景下会失效。姜明煜[25]实现了基于UiPath的机器人流程自动化系统，实现了面向自动化开发人员的模板框架，并且基于这套框架成功设计完成了管理会计报告自动化方案。祁伟等人[26]实现的RPA系统提高了电网接派单的效率，平均派单时间从 5 分钟降低至 1 分钟。英国的一家移动电信提供商部署了RPA系统，三年产生了 650%-800%的投资回报，响应时间缩短了 1500 倍[27]。Vajgel等人[28]为电力部门开发了一套智能RPA系统，极大的降低了用户对电力部门的投诉概率。但是，上述所有的自动化流程是固定的，或者相对应用场景而言，具有依赖性，如果使用者需要主动修改或者调整运行流程，需要具备一定的专业知识，甚至出现业务变更等需求时，修改代价会更高。此外，由于系统核心数据处理部分的规则是由开发人员制定的，难以保证数据的安全，尤其是敏感数据的处理，一旦相关内容泄露，将会带来严重的后果[29]。因此，如果能够提供一套由用户自定义规则且代码不涉及具体数据操作细节，仅提供数据操作方式的系统，就可以避免敏感数据处理过程中的安全性问题。

在RPA技术研究方面，较多的工作集中于流程挖掘技术，旨在利用流程挖掘技术，找出可以被自动化的流程任务清单。Leno等人[12]提出了一种Robidium工具，这是一种从用户交互日志中发现可自动化执行的任务，并可以生成机器人流程自动化脚本。与商业RPA工具提供的记录和回放功能不同，Robidium会将未专门记录的用户交互日志作为其输入捕获任务。Choi等人[15]采用了流程挖掘技术通过用户交互日志以获得RPA任务。此外，在网页流程自动化方面，Dong Rui等人[30]提出了一种基于用户演示的RPA流程生成系统，这是为数不多的针对RPA技术细节的研究。该系统采用了PBD(programming-by-demonstration)方法，需要用户对操作流程进行演示，系统将记录整个操作过程，并生成一系列操作代码，因此该方法生成的流程会与原有流程存在一定的误差，在此文章中，有 68%的样例能够达到 95%的准确率，系统效果较好。

在RPA上云的潮流下，云任务调度算法为云服务提供了重要保障，因此本文针对云任务调度展开了相关研究工作。

1.2.2 云任务调度研究现状

云调度的概念指的是将作业映射或者分配到一系列虚拟机，以利用可用资源来满足用户需求的技术[31]。在云计算中，采用调度方法的目的是提升系统吞吐量

和负载均衡，降低成本，提高资源利用率，节省成本，缩短处理时间[32]。目前，大多数云任务调度都以降低能耗和提高服务质量为主要目标。在降低能耗方面，Gao 等人[33]提出了一种基于蚁群算法和订单交换迁移的算法，其融合了信息素沉淀策略，有效减少了活跃主机数量，特别是在具有瓶颈资源特性的 VM 迁移问题上，该算法具有显着的节能效果。Chen 等人[34]提出了一种节能的任务集成启发式算法，该算法考虑了闲置计算资源的功耗，降低了数据中心的总能耗，保证了用户任务的完成质量。在服务质量方面，Farid 等人[35]建议在 IaaS 平台中使用 PSO 算法来优化工作流以确定最合适的 QoS 影响因素。Jing 等人[36]提出了一种调度算法 QoS-DPSO，实验结果表明 QoS-DPSO 能够有效提高性能，实现高可用。此外，由于云环境的状态变化较快，且云环境复杂，不同的任务调度优化算法影响着云调度的效果。

在任务调度算法中，根据调度目标可将其分为两类：单目标优化与多目标优化。单目标优化任务调度会导致结果具有很大的局限性而无法推广，而多目标优化可以使得多个实例尽量满足需求条件下的极优解[37]。为了解决多目标优化问题，启发式智能算法应运而生，该类算法通常为两类：基于生物启发和基于群体智能。遗传算法作为生物启发式算法的经典算法，许多学者采用该算法对云任务调度场景做相关优化。王国豪[38]等人为了实现云环境中科学工作流调度的执行跨度和执行代价的同步优化，提出了一种多目标最优化进化遗传调度算法 MOEGA，该算法以遗传算法为主，基于遗传算法对云环境中的工作流任务进行优化。而作为群体智能优化的关键算法，粒子群算法(PSO)的原理是模拟生物群落的社会行为[39]，从而有效的利用粒子群算法的信息共享能力对任务调度做算法优化。Wang 等人[40]基于工作流调度问题，提出了免疫粒子群算法(IMPSO)，有效的提高了优化的质量和速度，所提出的 IMPSO 克服了 PSO 收敛速度慢，容易陷入局部优化的问题。此外，Moon 等人[41]提出了一种基于蚁群优化的新型云任务调度算法，通过蚁群优化提高云计算环境中任务调度的性能，采用从属蚂蚁的多样化和强化策略，他们提出的算法通过避免信息素被领头蚂蚁错误积累的长路径解决了从属蚂蚁的全局优化问题，以高效的方式将云用户的任务分配给云计算环境中的虚拟机。

目前关于任务调度的优化算法类型繁多，但是，Nabi 等人[42]在文献研究中提到，无论是在网格计算还是在云计算环境中，粒子群能够取得更好的任务调度

效果。除此之外，粒子群拥有更少的参数和更可靠的计算性能[43]。综上，粒子群算法因其简单和低成本的特性而广受欢迎，粒子群优化算法也非常适合云计算中的任务调度[44]。

1.2.3 粒子群算法的研究现状

粒子群算法最初由 Kennedy 等人[45]提出，此后出现了大量的学者对粒子群算法做研究，Rodriguez 等人[46]提到粒子群算法不仅简单易用，并且收敛速度较快。

在通过参数的优化粒子群算法中，Duan 等人[47]提出了一种混沌自适应粒子群优化算法(CAPSO)，该算法采用混沌思想，不仅可以有效防止错过全局最优解，而且有很高的概率跳出局部最优解，具有良好的适应性。Cheng 等人[48]提出了一种精度高、收敛速度快的改进粒子群算法(ChPSO)，在 CES 生产函数模型中的参数估计上利用该算法进行优化，具有良好效果。张晓丽[49]在传统粒子群算法基础上，克服传统粒子群算法陷入局部最优的缺陷，将混沌优化搜索技术应用于粒子群优化算法中，提出了一种基于 Tent 映射的自适应混沌粒子群任务调度算法，实验表明改进算法有效地缩小了总任务的完成时间，具有较好的寻优能力和实时性。

在融合其他启发算法的研究中，马钰等人[50]提出了模拟退火结合粒子群的优化方案，利用模拟退火的全局搜索能力弥补粒子群的缺陷。Manasrah 等人[51]结合了遗传算法与粒子群算法，成功地降低了任务完成时间。马学森等人[52]提出了一种优化多目标任务调度粒子群算法(MOTS-PSO)，通过引入非线性自适应惯性权重，避免算法陷入局部最优，其次引入花朵授粉算法概率更新机制，平衡粒子的全局搜索和局部寻优，并对粒子的全局搜索位置更新进行改进。最后引入了萤火虫算法，产生精英解，利用精英解对粒子位置进行扰动，跳出局部最优状态。实验表明，MOTS-PSO 相较于 PSO 与 FA 算法，在收敛精度和速度上具有较大提升。Dordaie 等人[53]提出了一种融合爬山算法与粒子群算法结合的优化方案。Laskar 等人[54]提出了一种与鲸鱼算法混合的粒子群优化算法，相关算法表现出了良好的性能。Suyono 等人[55]提出了一种混合蚁群与粒子群混合的算法解决电力系统的调度问题。

Alkayal 等人[56]根据优化的目标数量划分了粒子群的研究工作。研究表明，

在云环境下，为了平衡工作负载和提高 QoS 参数，需要更多的改进以达到效果。Verma 等人[57]提出了一种混合粒子群算法对 IaaS 平台做优化。Saeedi 等人[58]使用改进粒子群对云计算工作流进行调度。Kumar 等人[59]提出了一种基于粒子群的资源调度算法，有效的优化了时间、成本、吞吐量等指标。

以上大多数粒子群算法没有考虑到增强种群多样性以提高全局搜索能力，且没有一种有效检测粒子群是否陷入局部最优解并帮助粒子跳出局部最优状态的方案。难以根据粒子的迭代状态有效及时地帮助粒子群算法跳出局部最优解，避免大量的无效搜索行为。以 CAPSO[47]与 ChPSO[48]为例，两种算法均通过优化尽可能避免粒子陷入局部最优解，提高算法的搜索性能，其方法并未针对已经陷入局部最优的粒子展开优化。

1.3 主要研究工作

本文设计并实现了一个RPA云服务系统，通过实际RPA应用场景设计了一个Micro-Benchmark，并对RPA云服务系统展开负载测试，得出RPA云任务负载不均衡的结论。本文针对负载不均衡的情况，对任务调度展开优化，以提高RPA云服务系统的执行效率。本文的主要工作有以下三点内容：

(1) 设计了一种基于网页自动化的RPA云服务系统，通过组合功能组件的方式实现用户自主按需设计任务流程，利用无人值守方案提供了用户任务以云端执行的模式。通过对RPA在实际场景下的应用调研，本文对制造业、财务、人力资源、物流及在线教育五大领域共设计了 23 个任务，构建了一个Micro-Benchmark，为后续RPA实际并发任务调度的实验提供支撑。

(2) 提出了一种基于滑动窗口与非支配解集的改进粒子群算法(Based on Sliding Window and Non-Dominant Pareto PSO, SN-PSO)。SN-PSO通过滑动窗口检测出陷入局部最优的粒子，并调整粒子位置中贡献较低的维度。该方案避免了随机变异导致的不确定性，减少了粒子的无效搜索行为。通过非支配解集的优化策略提高了种群的多样性，增强了粒子全局寻优能力。通过上述优化方案，使用SN-PSO算法对RPA云任务调度作优化。

(3) 实现了RPA云服务系统并针对主要工作进行实验评估。针对RPA云服务系统的实现做了功能测试，对RPA云服务系统中使用的改进粒子群算法

进行了实验评估与分析，实验表明在实际使用场景下，SN-PSO能够有效地降低并发任务总执行时间，提高系统的运行效率。

1.4 文章组织结构

本文的整体结构共分为六个章节。

第一章是绪论。其中主要介绍了RPA的相关研究背景及其意义，并结合目前现有的RPA产品与研究和任务调度算法与粒子群算法的国内外研究现状做介绍。最后针对本文的主要研究工作进行了相关介绍。

第二章是相关理论与技术。主要介绍了RPA、云服务、任务调度与粒子群的基础知识，为后续的章节提供理论支持。

第三章是RPA云服务系统的设计。首先，本章介绍了RPA系统的需求分析与总体设计，对RPA系统进行整体的阐述，明确RPA系统的功能，然后详细介绍了系统中的总体设计及技术细节的剖析，最后对RPA云任务负载做了背景设计，并利用设计好的任务进行负载测试与负载探索，得出了不同背景下的系统负载表。

第四章介绍了一种基于滑动窗口与非支配解集的改进粒子群的RPA云任务调度优化算法。本章中，首先对任务调度的问题背景进行了描述，并对多目标优化的任务调度数学模型进行了介绍，然后对粒子群的适应度函数设计与粒子编码策略两大前置性工作做了一定的阐述，最后介绍了本文粒子群算法中的基于滑动窗口的变异策略与基于非支配解集的优化策略。

第五章主要是系统实现与测试。主要介绍了RPA系统的详细实现结果，分别针对系统的功能做了演示。最后介绍了基于改进粒子群的RPA云任务调度实验，实验结果表明，在实际的并发调度场景下，本文的调度算法在RPA系统中能够有效的降低系统中实际任务的执行时间。

第六章是总结与展望。本章对论文的主要工作进行了一定的总结，分析目前系统工作中依旧存在的不足，并对接下来的进一步工作做了展望。

第2章　相关技术及理论

本章主要介绍构建 RPA 云服务系统的相关技术及其理论，其中包括了机器人流程自动化相关理论，容器云技术 docker 和 kubernetes 框架，并对本文涉及到的任务调度相关理论与本文采用的粒子群优化算法进行介绍，为后续章节的研究奠定基础。

2.1 机器人流程自动化

RPA的工作过程主要是模拟人类的行为，但它可以比人类更快、更精确地完成工作，旨在完全遵循公司提出的业务规则，实现每天 24 小时不间断的工作目标。尽管如此，RPA并不是要取代人类劳动，而是要将他们从重复繁琐的工作中解放出来，让他们可以专注于需要创造力和创新的工作。

因此，RPA的业务流程任务应该满足以下标准：知识水平较低的任务、高频执行的任务、查询不同系统或者应用程序的任务及人工易错率较高的任务[60]。综上，RPA具有以下特点：

（1） RPA易于配置，因此开发人员不需要编程技能。

（2） RPA 软件不是侵入性的，它基于现有系统，无需创建、替换或开发昂贵的平台。

（3） RPA 对公司来说是安全的，RPA 是一个强大的平台，旨在满足公司在安全性、可扩展性、可审计性和变更管理方面的 IT 要求。

由于 RPA 软件的非侵入性，RPA 较为关键的一部分是模拟用户操作[61]，然而在实际的使用场景中，用户的需求极有可能非常复杂，所设定的机器人未必能够完全依赖于代码控制完成任务，因此，随着机器人流程自动化技术的发展，RPA具有三种类型的运行模式：

（1） 有人值守模式。有人值守模式使用机器人作为工作人员的个人助手，在个人 PC 上运行并处理个人工作人员的任务。机器人接收到工人的指令后开始工作，例如：登录系统，搜索数据或者信息，识别需要的数据，将结果发送给工作人员，等待工作人员的反馈。如果数据正确，将转发给下一个过程，例如：生成新报告或更新旧报告。有人值守模式的缺点是，在机器人工作执行过程中，工作人员的电脑将被机器接

管，工作人员无法用电脑做任何事情。

（2） 无人值守模式。与有人值守模式不同，无人值守模式旨在为多个工作人员（非个人助理的形式）执行流程或任务。它不需要人为干预，只需要工作人员创建相应的规则和流程，即可在后台自动执行。机器人将会自动根据开发人员指定的时间执行任务流程。该机器人在虚拟机中工作，而无需干预本地计算机。工作人员能够远程执行任务并实时监控执行流程。

（3） 混合模式。混合模式结合了有人值守和无人值守模式的特点。它用于执行一个包含依赖人工主动决策和机器全自动执行的长过程。例如，工作人员启动一个有人值守模式下的机器人，然后该机器人在人工干预结束后，将自动启动无人值守的机器人来执行剩余流程，且该部分流程不再需要人工干预。它将一直运行到需要人工干预的部分。此时机器人又会转变为有人值守模式，以此往复。

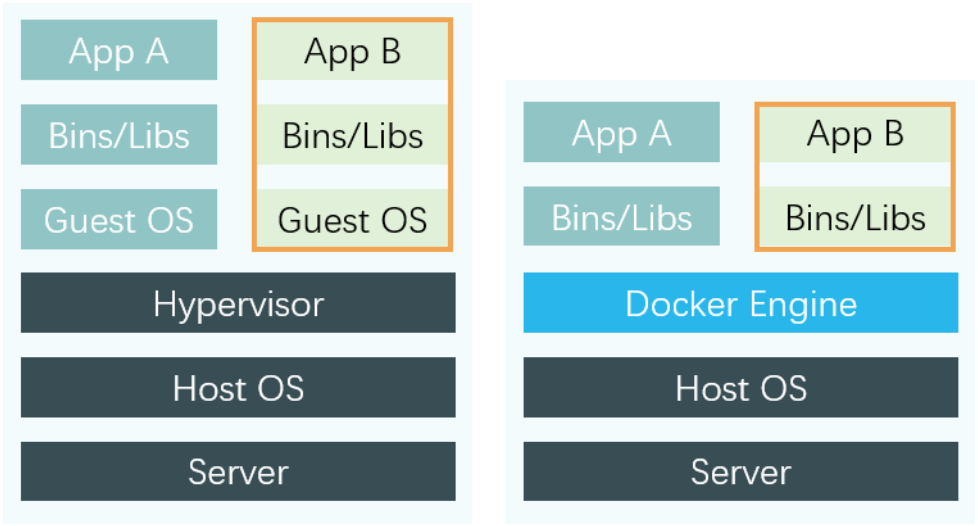
根据本文系统的特点，由于本文的 RPA 云服务系统中所有的任务均在服务器端运行，因此，为了便于系统设计，本文面向的 RPA 云服务系统运行模式仅提供无人值守型一种执行方案。

2.2 Kubernetes 简介

Kubernetes 技术起源于 Google 公司的 Borg/Omega[62]系统，Kubernetes 每一个新特性的提出都基于 Google 公司的 Borg/Omega 系统设计经验。但是 Kubernetes 的兴起，得益于 Docker 技术的风靡，因此，本节首先针对 Docker 容器化技术进行简要介绍，而后针对 Kubernetes 进行详细介绍。

2.2.1 Docker 容器化技术

Docker 容器是一种平台即服务(PaaS)产品，它通过操作系统虚拟化的方式交付软件。Docker 容器的核心功能就是通过约束和修改进程在操作系统中的表现，从而实现操作系统级别的虚拟化。对于 Docker 容器化技术而言，Cgroup 是进程约束手段，而 Namespace 技术则是进程隔离手段。因此，可以认为，Docker 容器仅仅是一种较为特殊的进程。



(a) 传统硬件虚拟化　　(b) Docker 虚拟化

图 2-1 虚拟化对比图

如图 2-1 所示，图(a)表示的是虚拟机的工作原理，其中，Hypervisor 是虚拟机最主要的部分，其用于模拟硬件环境。而图(b)使用了 Docker Engine 技术替换了 Hypervisor，通过此技术实现操作系统级虚拟化。

本文利用 Docker 容器化技术，将 RPA 云服务系统进行封装，为后续采用 Kubernetes 技术对容器进行管理与编排提供了基础。

2.2.2 Kubernetes 技术

Kubernetes 又称为 K8S，它是一个用于自动化容器部署、编排与管理容器的开源系统。Kubernetes 遵循主从架构模式，其中包含控制平面、节点等。

Kubernetes 中的主节点，即 Master 节点，处理集群中的控制平面，管理其工作负载与通信过程。各种组件构成了 Kubernetes 控制平面，每个组件都有自己的进程，在单个主节点或多个支持高可用性集群的主节点上运行。控制平面所包含的组件有：

- Etcd。它是 CoreOS 开发的数据存储系统，该存储系统具有分布式、轻量级、可持久化的特性。它可以可靠地存储集群配置，反映集群在任何时间下的整体状态。Etcd 更着重于一致性，而不是可用性。一致性对于正确调度和操作服务非常重要。
- API Server。API 服务器通过 JSON over HTTP 提供 Kubernetes API 的服务，它是 Kubernetes 的内部和外部接口。它处理和验证 REST 请求，并更新 etcd 中 API 对象的状态，让客户端可以跨工作节点配置工作负

载和容器。API 服务器使用 etcd 的 watch API 监控集群，推进关键配置改变，或者把集群状态的任何差异恢复到部署者声明的状态。如果 Deployment Controller 发现只有两个实例在运行，它会安排创建该 pod 的第三个实例。

- 调度器。调度器是一种可扩展的组件，它选择在某节点上运行未调度的 pod，并考虑可用性。调度程序跟踪每个节点上的资源使用情况，以防止调度超过可用资源的工作负载。
- 控制器。它通过与 API 服务器通信来新建、更新和删除它管理的资源，从而使实际集群状态达到所需状态。如果底层节点发生故障，它还会构建备用 pod。

Kubernetes 的节点(Node)也称为 worker，即工作节点，它是实际任务执行的节点。在 Kubernetes 的节点中，包含以下关键组件：

- Kubelet。Kubelet 监控节点和容器，并按控制平面指令管理 pod。它确保节点中所有的容器都是健康的。Kubelet 通过心跳机制检测 pod 状态。若发现故障，则重启对应的 pod。
- Kube-proxy。Kube-proxy 是网络代理和负载均衡器，支持服务抽象等网络操作。它根据请求的 IP 和端口号把流量转发到对应的容器。

综上，Kubernetes 的架构图如图 2-2 所示。

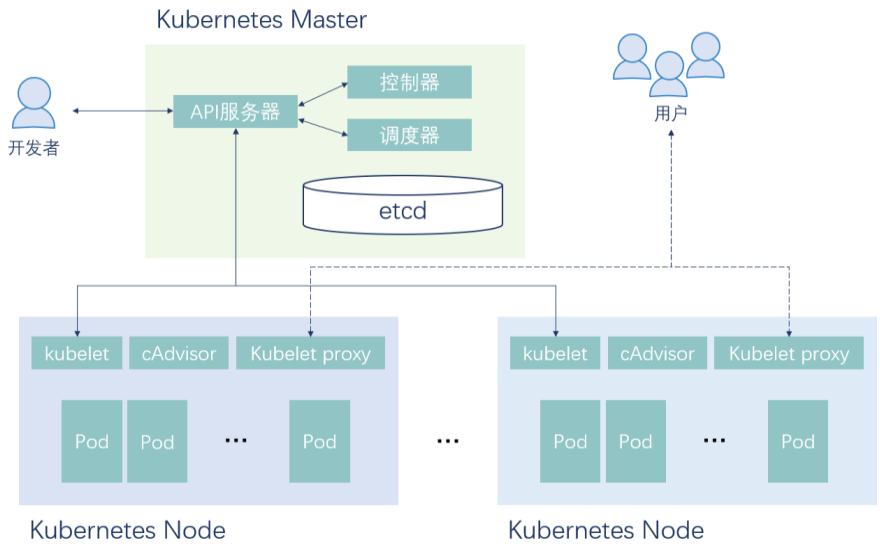


图 2-2 Kubernetes 架构图

在 Kubernetes 的节点中运行着多个 pod，每个 pod 都代表着一个服务实例，

一台节点上可能对应多个服务实例，本文面向的 RPA 云服务系统实际运行在 Kubernetes 节点中的 pod 内。

为了有效管理与编排RPA云服务系统的Docker容器，本文采用了Kubernetes技术构建了一个集群，提供了容器云技术基础，为后续 RPA 任务调度优化提供了环境支撑。

2.3 任务调度

近年来，云计算的进步有助于互联分布在各个不同地理位置的数据中心，以提高更好的服务质量。随着数据中心的访问量增加，云规模也将变得越来越大，在云环境中将会运行着海量的任务以处理繁杂的请求。然而，各种请求与任务如果没有正确的分配与安排，其性能将会受到较大的影响，因此，任务调度机制在云计算中起着至关重要的作用。调度算法主要用于规划云计算中的任务分配，以获得最大的收益，包括更好的负载均衡度、执行效率等。

2.3.1 任务调度介绍

任务调度可以表示为如下的问题：给定一组包含 n 个任务的集合$\{T_1, T_2, \ldots, T_n\}$，该任务集合需要被分配到一组包含 m 个可用的机器集合$\{M_1, M_2, \ldots, M_m\}$中，当 m=1 时，该调度问题称之为单机调度问题，当 m>1 时，调度问题被称之为多机调度问题，上述任务调度问题如下图例所示。

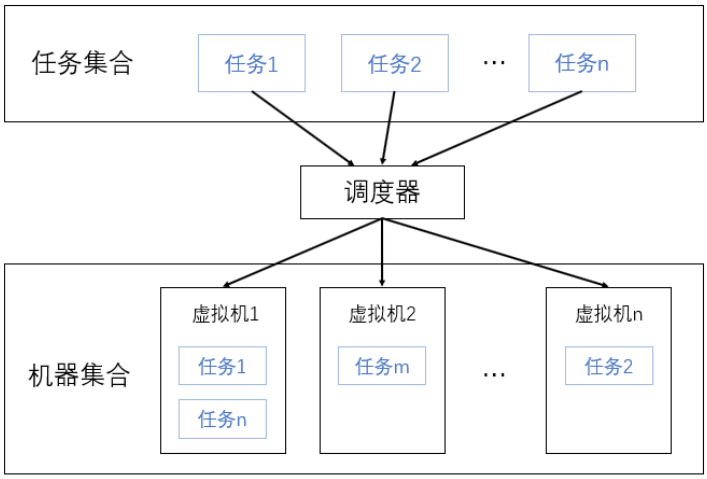


图 2-3 任务调度示例图

根据任务调度方法的特征，可以将任务调度分为：静态和动态、用户级和系统级、在线和离线、抢占式和非抢占式等[63]。

- 静态和动态调度：静态调度算法需要有关传入任务的信息（如任务计数、

任务长度、任务截止日期等）和可用资源情况（如内存、处理能力、节点处理能力等）的静态信息。当工作负载变化不频繁且系统行为的变化很小时，静态调度算法的性能会更好。在云环境中，负载不断波动，其可用资源信息变化较大，利用该种方式计算出的预计结果与实际调度结果具有较大的差别，因此静态调度算法不适合云计算。静态调度算法包括先进先出（FIFO）、最短作业优先（SJF）等。反之，动态调度算法会更加准确、高效和适用，因为当云环境中的任何一个节点的性能资源发生较大幅度变化时，算法可以监测到该节点上的资源状态，并将需要执行的任务转移到资源充足的节点。如元启发式算法广泛应用于云环境的动态调度算法，其中包括粒子群算法、蚁群算法等。

- 用户级和系统级调度：用户级调度是由调度器解决服务提供者和服务消费者间的服务提供问题，效率很高，尤其是在面向市场的虚拟化资源的环境下。而系统级调度，则是数据中心负责资源管理，这将影响数据中心的性能，导致效率降低。
- 在线和离线调度：在线调度是由调度器将请求映射到虚拟化机器上，以便调度过程在一段时间内保持稳定，每次对一个任务执行一次调度任务。离线调度又称为批处理调度，该种调度方式是根据配置好的时间节点响应传入的用户请求，从而分配计算资源，离线调度在任务数量的并发量较高时能够快速计算出调度结果。
- 抢占式和非抢占式调度：抢占式调度指的是，当前正在执行的任务可能会被新的请求中断，这将导致旧任务的执行效率变低。而在非抢占式调度策略中，当前机器上只有将已分配的任务请求在任务执行成功后才能释放，并执行下一个任务，没有中断。
- 集中式和分布式调度：集中式调度是所有任务请求都由中央处理单元捕获，然后经过计算得到调度结果后，根据调度表发送到从属处理单元，每个处理单元均需要接受一个调度队列。反之，在分布式调度策略中，没有中央控制单元，因为所有需要处理的请求由本地调度器负责，并通过与其他处理单元持续共享调度信息，来维护所有的状态。
- 合作和非合作调度：在合作调度中，任何一个处理单元在做出调度决策

时，需要通过与其他处理单元合作的方式执行调度过程。即，所有的任务将通过处理单元之间的合作模式顺序执行。但是，在非合作调度中，每个处理单元独立做出决策，而不会影响其他处理单元。

2.3.2 任务调度算法分类

任务调度可以根据所使用的算法不同，分出以下四类：传统调度方法、启发式调度算法、元启发式调度算法和基于人工智能的调度算法[63]。

（1） 传统调度

在传统调度方面，通常此类方法会在任务特征维度进行分析，如权重、截止日期、处理时长等，而后根据设定的约束进行调度。后续所有的任务请求都将根据设计好的规则进行调度，实现起来非常简单，但是对于云计算环境而言，该种方法并不合适。

（2） 启发式调度

启发式调度方法的效果通常取决于问题的特性，可能在某些问题上表现良好，但是在其他问题中表现不佳。启发式调度算法依赖于问题中的场景特性，若该问题场景中的条件和特点不断变化，那么启发式调度算法就难以在此类问题上发挥优势，而云环境中的任务调度问题就是如此，因此，启发式调度算法并不适用于云环境下的任务调度问题。以下是几种经典的启发式算法。

Min-Min 算法是一种基本的启发式算法，该算法的核心原理是：从所有提交的任务中挑选出来一个最短执行时间的任务，并将该任务分配到处理速度最快的虚拟机器上，这个过程一直持续到所有任务分配完毕。这种方法可以有效的处理短任务场景，长任务必须等到短任务分配完毕后才能被响应。虽然，该算法极大地提高了整个系统的吞吐量，但是，任务可能会出现饥饿问题。

HEFT 算法是一种基于列表的启发式调度算法，算法将构建任务优先级列表，以便根据任务的预计完成时间为每个任务计算出最佳分配方案。

Sufferage 算法也是一种经典的启发式算法。该算法首先计算每个任务在每个机器上的完成时间，其次，计算每两个连续任务之间的最短完成时间的差，并将其称为 sufferage 值。最后，将执行时间最短的机器分配给 sufferage 值最大的任务，更新所有机器的执行时间，重复上述步骤，直到所有任务调度完成。尽管该算法在很多问题场景下表现优秀，但如果多个任务具有相同的 sufferage 值，

则该策略将会简单地选择第一个到达的任务请求并执行，而不考虑后续任务，这可能会出现饥饿问题。

（3） 元启发式调度

在过去的二十年里，元启发式算法因其在解决复杂和大型计算问题方面的有效性而受到了极大的关注。元启发式算法具有如下特征：不受问题的影响、算法能够有效地探索解空间并找到 NP 完全问题的较优解决方案、元启发式通常是非确定性的算法。元启发式算法可以用于解决不同领域的问题，具有较快的搜索速度，因为它们不会过度依赖于要解决的问题特性和场景。在云环境的任务调度场景下，通常一个调度方案的解空间会比较大，因此任务调度过程通常需要较长时间才能找到最佳解决方案。元启发式调度算法可以避免长时间搜索的问题，由此可见，元启发式调度算法非常适用于云环境下的调度问题。

蚁群优化算法（ACO）：蚁群优化算法通过蚂蚁的自然行为进行模拟，通过找到蚁群和食物源之间的最佳路径的思路，对解空间进行搜索。其原理是蚂蚁在沿途移动时会排出信息素。随着时间的流逝，信息素会形成最短的路径，信息素的强度可以确定通往食物供应的最短路径。但是蚁群算法搜索时间较长，在云环境下，这将提高用户的响应时间，不利于云任务调度。

遗传算法（GA）：遗传算法基于受自然启发的进化模型，算法中每个染色体（种群中的个体）都包含一个基因，这些基因实际上代表了一个可能的解决方案，基因将被适应度函数计算并评估，根据适应度值选择染色体，然后进行交叉和变异操作，产生新的后代，用于组成新的种群。重复这个过程，直到产生足够数量的后代。但是，遗传算法搜索速度较慢，因此需要较长的迭代时间才能获得一个较好的解，在云环境下不利于任务调度过程。

人工蜂群优化算法（ABC）：ABC 算法最初由 Karaboga 提出[64]。该算法模仿了蜜蜂的探索模式和觅食的行为。ABC 算法具有两个过程，即前向传播和反向传播。在前向传播中，每只蜜蜂搜索解空间中的解决方案并选择部分解决方案后返回到蜂巢。在反向传播之前，蜜蜂们集中在一起，用他们新获得的个体解决方案开始反向传播。但是，人工蜂群存在收敛速度慢的问题。

粒子群优化算法（PSO）： 1995 年，Eberhart 和 Kennedy[44]提出了 PSO 算法。PSO 在算法初期通过生成一组随机的解，而后每个粒子利用分配好的速度进

行移动，并搜索解空间。在每次迭代中，PSO 算法使用每个粒子的个体历史最优解和整个群体的最优解来调整自身的速度。这种优化方法因其简单性和低计算成本的实用性而广受欢迎。

（4） 基于人工智能的调度

近年来，随着人工智能的发展，利用深度学习相关技术对任务调度进行优化的研究越来越多。其中一个最具代表性的例子就是强化学习算法。强化学习的本质是通过交互来学习。强化学习的代理与其环境交互，并在观察其行为的结果后，可以利用获得到的奖励值进行学习并改变自身的参数状态。这种试错学习方式源于行为主义心理学，是强化学习的主要基础之一。

强化学习在云计算任务调度领域中具有一个非常大的优势，它可以学习当前云环境中所有的节点状态与任务分配方式，并且能够充分的学习到节点与节点、任务与任务之间的协同工作关系，在云环境中理论上可以拥有更优调度效果。

但是强化学习算法的优势依赖于云环境信息不会出现变动的约束，一旦云计算环境中出现了需要增减机器的需求，那么之前所训练得到的强化学习模型在新的环境下难以有效分配任务至新机器上，必须重新对强化学习算法进行训练。而元启发式算法可以有效避免这个问题，对环境信息的兼容性更强。

综上，在本文的 RPA 云服务系统中，任务调度模块采用元启发式算法对任务调度问题进行优化。为了保证云环境下的任务调度具备足够高的效率，使得用户响应时间不会因调度算法增加，本文选择了收敛速度较快的粒子群优化算法，且粒子群算法具有较强的鲁棒性，在云环境下有较好的搜索效果和性能[44]。

2.4 粒子群调度优化算法

粒子群算法由 Kennedy 和 Eberhart 于 1995 年共同提出[44]，算法融入了个体认知与群体认知的社会学理论，是一种模拟鸟群觅食的简化算法模型。在粒子群算法中，有一定数量的粒子将散落在 D 维的搜索空间上，每个粒子的位置在搜索空间中随机初始化，而后通过群体最优值与粒子个体最优值构成约束，参考两者与个体速度向量共同执行迭代过程，直到满足条件为止，最后算法会得到搜索过程中的全局最优解。

2.4.1 标准粒子群算法

假设有一个 D 维的搜索空间，空间中存在 N 个粒子，对于任意一个粒子 i，

当粒子迭代到第 t 代时，其位置向量被记为$X_i^t = (x_{i1}^t, x_{i2}^t, x_{i3}^t, \dots, x_{iD}^t)$，粒子的速度向量被记为$V_i^t = (v_{i1}^t, v_{i2}^t, v_{i3}^t, \dots, v_{iD}^t)$，该速度向量用于在迭代过程中更新粒子的飞行位置。粒子将维护一个个体历史最优解，记为P_i^t，并记粒子群的群体最优解为G^t。

当粒子群迭代至t+1代时，位置和速度的更新策略按照如下的公式进行计算：

$$V_i^{(t+1)} = wV_i^t + c_1\zeta(P_i^t - X_i^t) + c_2\eta(G^t - X_i^t) \qquad (2-1)$$

$$X_i^{(t+1)} = X_i^t + V_i^{(t+1)} \qquad (2-2)$$

在标准粒子群算法中，w 的值恒为 1。后续 shi 等人提出惯性权重的概念[65]，即参数 w，此参数以控制粒子的更新方式，影响着粒子的全局搜索能力以及局部搜索能力，w 越大，粒子的全局搜索能力越强，反之，粒子的局部搜索能力越强。对于c_1与c_2，标准粒子群中通常将该值设置为 2.0。但是随着压缩因子的提出[66]，上述公式可以转换为具有 w=0.729，$c_1 = c_2 = 1.496$约束的公式，其数学上与压缩因子所计算得到的公式是等价的。因此，在本文中的参数设定中，采用了如上的参数设置值。

综上，粒子群在搜索过程中，空间的 N 个粒子是一个互动的种群，每个粒子都各自维护着一个个体历史最优解P_i^t，每当粒子迭代至下一个位置时，都将个体历史最优解与粒子群体最优解作比较，一旦出现更优解，则将群体最优解做替换，不断往复。粒子群信息交流的桥梁便是群体最优解，通过模拟自然界中的鸟群觅食行为，群体认知与个体认知的互相影响以搜索到空间中的最优解。

2.4.2 标准粒子群算法流程

标准粒子群的算法流程如下所示：

（1） 初始化。设定算法的粒子个数、迭代次数、算法涉及到的参数、边界值等相关信息，同时在规定的搜索空间内随机生成若干个粒子，数量与设定值相同，并对每个粒子的位置坐标作随机化操作。初始化每个粒子的个体最优适应度值。并设定初始种群中的群体最优适应度值。

（2） 速度与位置的更新。利用粒子的速度对其位置进行更新，并更新粒子的速度向量。

（3） 计算并更新个体适应度值，若当前适应度更优则替换，并替换个体最

优解。

（4） 计算并更新种群最优适应度值，若当前适应度值更优则替换，并替换种群最优解。

（5） 根据终止条件判断算法是否需要停止，若满足，则停止迭代。否则回到步骤 2 中。

（6） 输出算法的种群最优值与种群最优解。

本文将针对算法步骤(2)与(4)进行了优化，利用改进后的粒子群对本文的 RPA 云服务系统进行任务调度优化，提高系统的执行效率。

2.5 本章小结

本章介绍了 RPA 云服务系统中所涉及到的相关技术，对相关技术的技术原理和发展历史进行了介绍。首先叙述了机器人流程自动化相关理论，RPA 云服务系统依赖的容器云技术 Docker 容器与 Kubernetes 管理与编排框架，为后续 RPA 云服务系统设计与开发提供了基础。最后，本章介绍了本文涉及到的任务调度相关知识及本文使用到的粒子群优化算法，为本文的任务调度算法优化提供了理论支撑。

第3章　RPA 云服务系统设计

本章介绍 RPA 云服务系统的设计。首先，对 RPA 云服务系统的需求展开分析，其中包含功能性需求与非功能性需求。然后，介绍系统的总体架构设计与系统的功能模块设计，并介绍系统提供的所有功能组件，明确本文的 RPA 云服务系统面向的场景为网页自动化。最后，在 RPA 实际的任务场景基础上设计一个 Micro-Benchmark，用以测试 RPA 任务负载的性能消耗情况，得出 RPA 任务负载不均衡的结论。

3.1 系统需求分析

本节内容主要介绍了 RPA 云服务系统的需求分析，通过对 RPA 现有产品的调研，并根据面向场景得出系统主要目标包括：RPA 流程编排、RPA 任务执行、RPA 执行状态监控、RPA 任务处理和 RPA 任务执行计划五个部分，并针对 RPA 云服务系统的非功能性需求展开分析。

3.1.1 系统功能性需求

在前期大量对 RPA 系统调研及了解实际应用需求的过程中，本文明确了产品面向的使用场景是网页自动化过程，最终得出了系统功能性需求，RPA 云服务系统具有如下五点需求：

（1） RPA 流程编排需求

由于本系统面向的对象是非专业编码人员，因此，本系统需要提供一套基于用户拖拽式的使用模式，并且针对网页自动化操作中可能出现的功能进行分类并管理，以便用户可以尽可能避免专业性知识需求，即可通过类似搭建积木的方式构建一套符合自身需求的任务工作流。同时，为了保证用户的使用体验，本系统需要提供一个任务画布，使得用户能够在画布上随意编排组件，并用连线的方式将组件之间链接起来。

（2） RPA 任务执行需求

支持用户在任何状态下运行指定的任务工作流，并将任务工作结果与组件工作状态及时的反馈至用户界面，以帮助用户清晰的了解到当前的任务工作状态与

组件执行情况。除此之外，还需要在后台提供一套自动解析用户工作流并生成任务执行计划的功能，避免用户设计的用户流程出现基本的逻辑错误或者死循环问题，以此为非专业编码用户的使用背景做好充足的准备。

（3） RPA 执行状态监控需求

针对用户离开任务执行页面的场景，提供任务执行概览的功能，其中主要包括了当前任务执行状态、任务执行进度、任务名称等任务信息。系统不仅需要记录当前任务的执行状态信息，还需要在后台记录在任务执行出错时的组件及其出现错误的原因，帮助用户在设计的任务流出现疑难问题时可以发现问题，并在无法解决问题时向管理员求助，更快的定位任务设计问题。

（4） RPA 任务处理需求

为了使得用户有效处理不同任务执行过程中的问题，该需求能够提供任务协同编辑、任务中断恢复、执行机器人管理的功能。系统不仅需要保证任务工作流能够被单个用户编辑与使用，还需要使得任务能够被分享，帮助用户更有效的协作完成流程编辑过程，并且在任务出现异常中断时，也需要保证用户无需再次重新启动任务，等待漫长的时间。以此为系统良好的使用体验打下基础。

（5） RPA 任务执行计划需求

系统提供立即运行与定时计划运行两种模式，实现有人值守型机器人与无人值守型机器人两种类型。支持按照小时、日、周、月等多种方式对用户任务定时启动。还需要提供执行结果管理，帮助用户了解定时任务的执行情况，用户可以及时根据具体执行情况调整任务流程。

3.1.2 系统非功能性需求

RPA 云服务系统不仅需要满足以上的系统功能性需求，还需要满足系统的非功能性需求，涉及到用户界面设计、系统可维护性、系统可扩展性、系统可靠性等方面。RPA 云服务系统具有以下四点非功能性需求：

（1） 易用性

系统的易用性是为了保证用户能够轻松理解和使用系统中的重要功能，因此 RPA 云服务系统需要提供简便易用的用户界面与操作逻辑，保证系统的可理解性。使得用户在没有专业知识的支撑下依旧能够快速掌握使用方法。

（2） 可靠性

系统的可靠性是保证用户在使用系统的同时，能够按照用户的预期及时响应用户请求，以及在不同网络环境下能够稳定运行，这就要求系统具备较高的性能可靠性。这样即便是用户客户端出现故障，依旧不会影响用户任务的执行过程，保证用户的任务可以成功地执行完毕。

（3） 可扩展性

系统的可扩展性要求系统能够根据用户的新需求快速增加功能，以满足变化的需求。保证在系统的使用过程中，能够在不改变原有功能的前提下快速迭代出具备新特性的系统版本，逐步优化系统。

（4） 可维护与兼容性

系统的可维护性是为了保证系统能够正常运行的同时，在系统的结构、架构上的维护变得容易。保证在系统出现故障时能够快速修复并再次为用户提供服务。系统的兼容性要求主要体现在浏览器上，即系统能够支持多种浏览器，并且保证在不同的浏览器上都能良好的运行。同时，为了支持不同的操作系统，系统需要兼容安卓、IOS 等多种类型的操作系统，以保证系统能够实现真正的开箱即用，不限制于用户使用环境。

3.2 系统总体设计

本节根据系统需求分析，进行 RPA 云服务系统的整体架构设计与功能模块设计。系统用例描述如附录II所示。

3.2.1 总体架构设计

RPA 云服务系统的整体架构主要被设计为如下 5 层：表示层、核心功能层、系统管理层、容器云技术设施层、数据层。整体的系统架构图如图 3-1 所示。

（1） 数据层

在 RPA 云服务系统的数据层上，使用 MySQL 8.0 关系型数据库对用户信息、任务信息、执行日志、任务执行状态等进行存储，并将系统的配置信息在数据库中进行存储，以此为上层服务，提供数据基础。

此外，数据层在 MySQL 数据库的基础上实现了多租户模式下的数据隔离。通常，多租户的实现方式具有以下三种模式：独立数据库、共享数据库隔离数据架构、共享数据库共享数据架构。本文采用了共享数据库共享数据架构的模式，该方案的维护成本较低，允许每个数据库支持的用户数量最大，选用该种方案可

以拥有更低的成本且租户的数据隔离级别通常不高，因此采取此种方案最优。在租户的资源隔离方面，企业级使用场景中，RPA 任务与业务紧密相关，任务与任务之间对资源的占用情况差别较大，用户难以决策需要多少资源来完成这个任务，导致租户资源的浪费，因此资源的分配应由服务端决定。

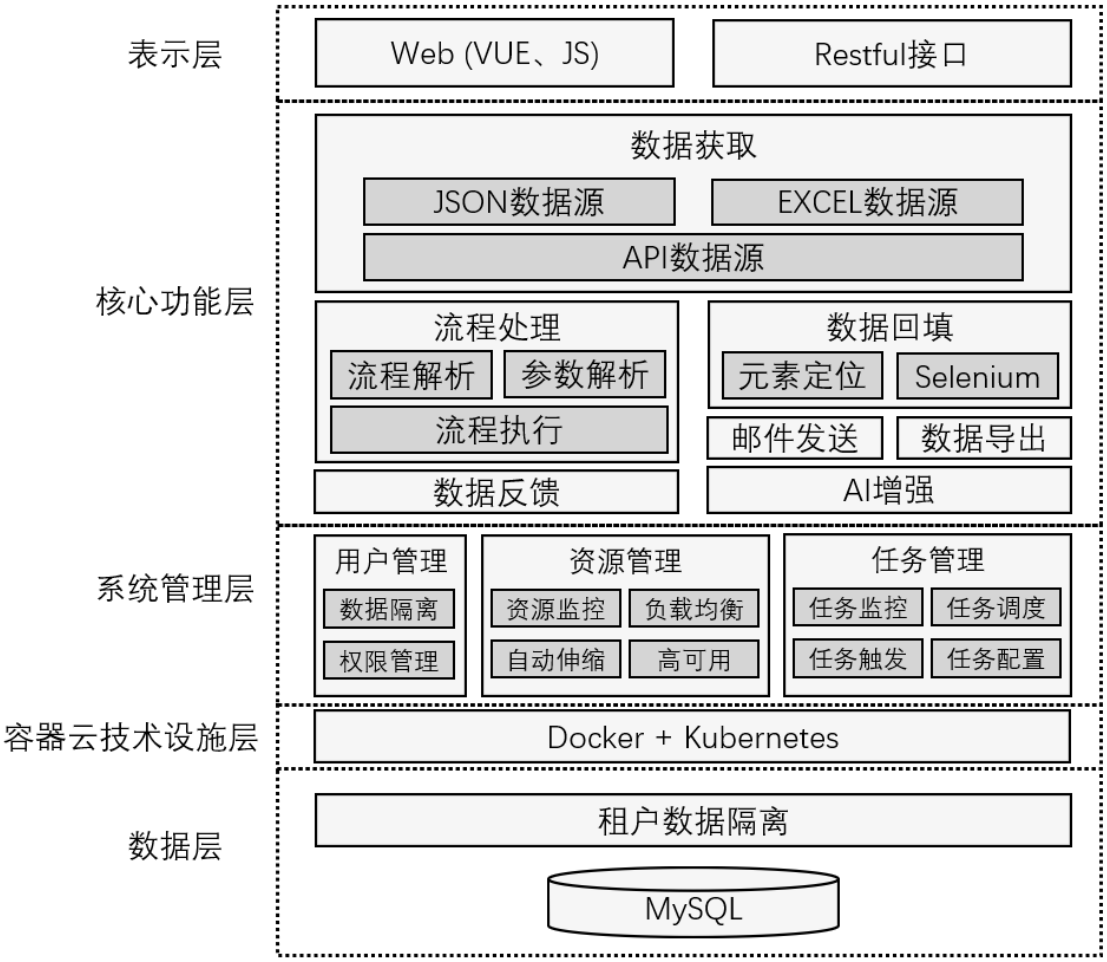


图 3-1 RPA 云服务系统架构设计图

（2） 容器云技术设施层

本文使用 Docker 容器化技术对系统进行封装，通过 Docker 在操作系统层面上的隔离技术，使得本文系统的启动过程在秒级，并且可以在一个操作系统下支持多个容器，可以极大的利用物理机提供的各种资源。同时，采用 Docker 容器化技术也可以避免了生产环境与开发环境不匹配的问题，具有极高的移植性，可以保证系统在任意一台服务器上轻松兼容。通过这种方式打包的系统，可以在任意云环境下快速部署。基于以上，本文采用 Kubernetes 对容器进行管理与部署，并利用 Kubernetes 搭建集群，为系统的运行提供充分的资源保障。

（3） 系统管理层

在系统管理层中，主要有以下三个部分：用户管理、资源管理、任务管理。通过这三大部分对整个系统服务提供稳定的支持。

用户管理中主要实现了数据隔离与权限管理两大部分。数据隔离的实现主要通过用户信息验证的方式，利用用户的 ID、账户信息等约束对用户使用的查询、任务执行等相关功能进行限定，保证用户与用户间的任务执行与存储是相互隔离的。而权限管理部分则是基于 Ruoyi-Cloud 中提供的授权与鉴权技术实现的，其中的权限包含：菜单权限，前端权限，后端权限和数据权限。

资源管理中主要包含资源监控、负载均衡、自动伸缩、高可用四部分。其中资源监控的核心目的是为了保证容器在基于 Kubernetes 实现的资源隔离上实时查看当前的系统状态，用于帮助在后期运行过程中出现资源不足时可以做到快速调整资源的目的。上述中涉及到的资源隔离则是使用的 Linux 提供的 Cgroup 技术实现的。负载均衡部分是使用基于 Kubernetes 提供的负载均衡服务，其主要是通过暴露 NodePort，绑定主机的某个端口，然后对 pod 的请求进行转发，实现负载均衡。自动伸缩部分采用的是 kubernetes 的水平伸缩技术（HPA），相比于垂直伸缩技术（VPA），HPA 使用者更多，技术更成熟，VPA 则暂未大规模的投入到生产环境中使用，其次，VPA 在资源不足时需要重启容器以扩容，这个过程会导致整个服务停止，HPA 则没有这种缺陷。最后，高可用技术则是利用 Kubernetes 对 Docker 的监控实现，当 Docker 服务宕机后，Kubernetes 会自动检测到掉线的 Docker 并将其再次拉起，保证服务的高可用。

任务管理中包含任务监控、任务调度、任务触发、任务配置四个部分。任务监控主要用于 RPA 任务执行监控，通过对任务执行状态的监测将任务执行信息反馈给用户，提高用户体验，利用 JAVA 的线程池对任务执行做管理，以获得更全面的任务执行信息。任务调度主要用于改善集群的负载均衡度，并且通过资源的高利用率来降低任务的执行时间，此部分使用本文提出的一种改进粒子群算法做任务调度优化。任务触发则是通过 Ruoyi-Cloud 提供的 quartz 作业调度框架实现的定时触发特性及通过 RestfulAPI 实现的立即触发特性。任务配置主要为用户提供任务编辑、任务管理等相关功能，该部分主要使用 JAVA 及数据库实现。

（4） 核心功能层

核心功能层主要包含了数据获取、流程处理、数据回填及其他辅助特性。数

据获取部分中 API 与 JSON 数据源通过基于 Alibaba 的 FastJSON 工具提供支持，而 Excel 数据源则通过基于 Apache 的 WorkBook 框架提供读取支持。流程处理部分利用 JAVA 实现，其中利用设计模式的模板方法模式、工厂模式等为后续维护、拓展提供支持。数据回填部分借助 Selenium 引擎实现网页自动化特性，利用 Xpath 对网页元素进行定位，实现对用户实际操作浏览器的模拟要求。其余辅助特性中的 AI 增强特性则利用百度公司旗下开源的 PaddlePaddle 作为基础，引入图片识别（OCR）、表格识别等特性，利用深度学习中 CV、NLP 两大部分对系统的功能做拓展。

（5） 表示层

在表示层中，RPA 云服务系统通过基于 VUE 与 Echarts 可视化工具对任务信息、执行状态信息等进行数据展示。其中 Restful 接口主要为核心功能层的 API，实现前后端交互过程。

3.2.2 功能模块设计

RPA 云服务系统的功能模块主要分为四个部分：RPA 工作区模块、执行计划模块、RPA 任务处理模块与系统模块。RPA 云服务系统的功能模块图如图 3-2 所示。

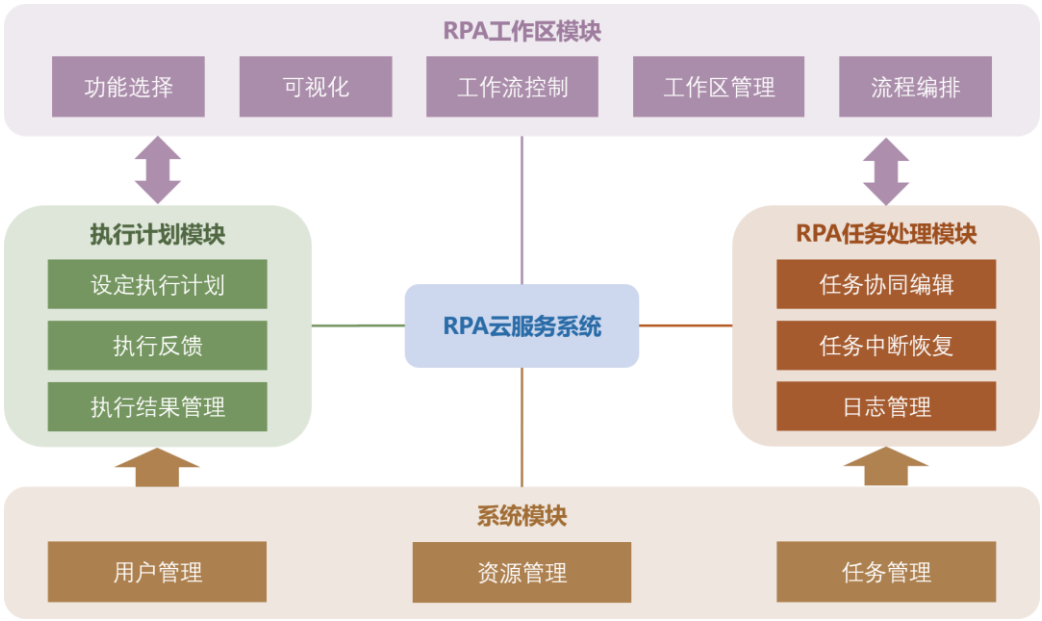


图 3-2 系统功能模块设计

（1） RPA 工作区模块

在 RPA 工作区模块中，用户可以管理已有的 RPA 工作流程、添加新的 RPA

工作流程，同时也可以针对已有的工作流程进行编辑、可视化等操作。在本模块中还提供了一个可视化的编辑器，用户可以在上面拖拽组件并设置参数，实现快速制定 RPA 工作流程，从而构建出符合自身需求的 RPA 流程。此外，本文的 RPA 云服务系统核心功能主要是为网页自动化特性提供服务，因此本文的系统在功能选择部分所提供的组件类型包括：数组读取类型、数据处理类型、条件判断类型、循环类型、网页自动化类型、AI 增强类型、其他功能型组件等七种大类。

在 RPA 工作区模块中，功能选择部分提供的所有组件共计七种类型：数据读取类型、数据处理类型、条件判断类型、循环类型、网页自动化类型、AI 增强类型与其他类型，如图 3-3 所示。

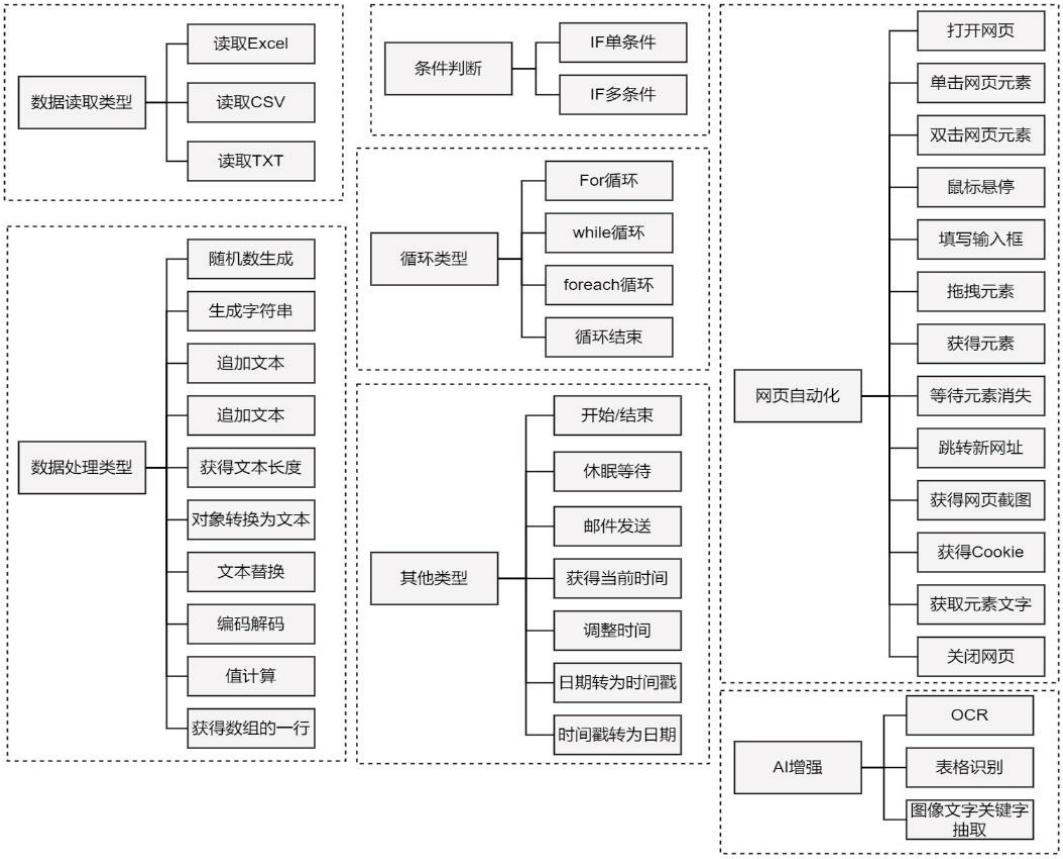


图 3-3 系统组件图

在 RPA 工作区模块，由于系统需要针对用户任务执行解析并运行的操作，因此，本文对此模块的实现划分为三个组成部分：流程解析器、任务执行器、功能组件库。上述模块中，功能组件库模块则包含图 3-3 中描述的所有功能组件，其中，AI 增强与网页自动化两种组件类型分别依赖于 Paddle 框架与 Selenium 执

行引擎。该模块主要实现结构如图 3-4 所示。

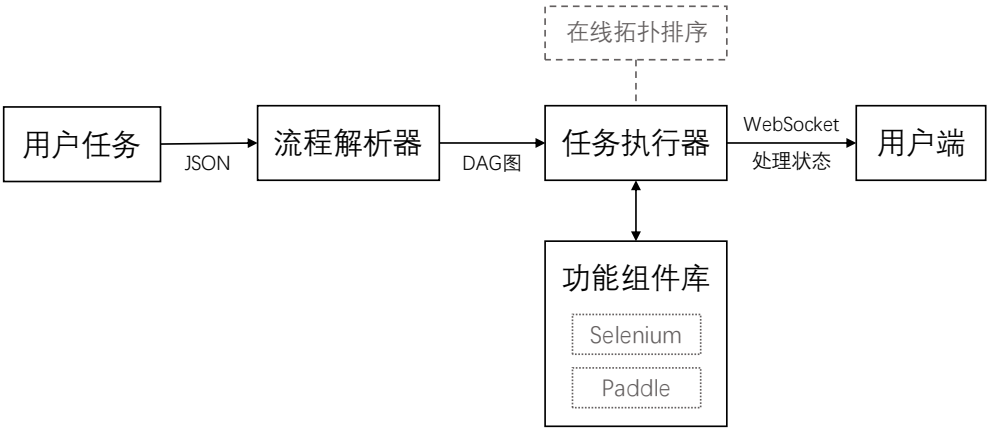


图 3-4 工作区模块实现结构图

图 3-4 中，用户任务以 JSON 的形式提交至流程解析器，流程解析器需要解析 JSON 信息，并将其转换为有向无环图(DAG)图。为了高效地完成这个解析过程，本文采用 Alibaba 提供的 FastJSON 框架对用户任务实现解析过程，将所有的边与节点以邻接表的方式构造出一个 DAG 图。若用户任务并不满足有向无环图的约束，流程解析器将停止工作，并告知使用者用户任务存在环路，需要修改任务流程。

随着流程解析成功，该任务转化为了一个由邻接表存储的 DAG 图后，任务执行器将利用该 DAG 图做拓扑排序，以按照用户的执行顺序尝试执行 RPA 任务。但是，传统的拓扑排序为离线处理，即针对 DAG 图一次解析为一个拓扑序列，本文系统中由于存在循环、条件判断两种组件，会导致用户组件执行序列中出现环路及分支，传统的拓扑排序已经无法满足本 RPA 系统的需求。由此，本文针对传统的拓扑排序进行了修改，将其改造为了一种在线拓扑排序算法，其算法流程如下。

1) 输入 DAG 图。
2) 在 DAG 图中选择所有图中入度为 0 的点加入到**在线拓扑队列**中。
3) 任务执行器取出队列首个节点，并将该节点的有向边删除。并执行节点对应的功能组件。
4) 检索 DAG 图中是否存在新的入度为 0 的点，加入至**在线拓扑队列**中。
5) 重复步骤 3 和 4。直到在线拓扑队列为空。

与传统的离线拓扑排序不同，本文在原算法基础上加入了在线拓扑队列，通

过此队列可以实现 DAG 图出现分支时，只执行其中一条分支路径上的功能组件需求。

此外，为了解决循环组件在拓扑序执行过程中存在环路的问题，在线拓扑队列每个位置中存放的元素有两种类型：节点和节点组。以收集异常信息流程为例，简化的任务示例图如图 3-5 所示。为了便于展示大致执行过程，图中将整个任务流程以系统解析完成后的 DAG 图的方式展现，并且仅包含循环节点类型。

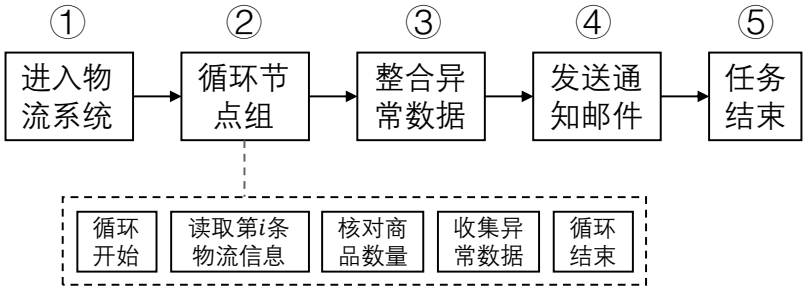


图 3-5 简化的收集异常信息任务示例图

任务执行器将根据图 3-5 中标记的①至⑤顺序逐步将每个元素加入至在线拓扑队列中。每当任务执行器执行节点对应的功能组件时，在线拓扑队列无需维护任务流程中存在的环路，只需提供对应的节点或节点组，任务执行器将自动根据不同类型的元素执行相应的处理过程。

通过上述方案，任务执行器即可通过在线拓扑排序算法，自动读取 DAG 图中的所有节点信息，利用系统的功能组件库，即可执行用户任务。

最后，任务执行器将根据每个组件的处理结果，将相应的处理状态与用户指定的中间数据，利用 WebSocket 技术推送至用户前端，实现实时的节点状态反馈过程。

（2） 执行计划模块

执行计划模块主要实现了无人值守型 RPA 机器人定时任务的设定、执行反馈及执行结果管理。它主要由设定执行计划、执行反馈和执行结果管理三个部分组成。通过设定执行计划，用户可以实现机器人定时启动的功能，执行反馈则是在机器人启动定时任务时提供用户友好的提示，或者在定时任务结束后通知用户，从而使用户可以了解机器人状态。此外，执行结果管理也是必不可少的，它能帮助用户整合定时任务产生的中间数据与最终结果，从而更好地把握机器人的整体活动。

（3） RPA 任务处理模块

RPA 任务处理模块是 RPA 机器人技术的重要组成部分，它具有任务协同编辑、任务中断恢复、执行机器人管理三大功能。任务协同编辑可以帮助用户对多个任务进行协同编辑，避免用户之间的信息障碍，快速构建一个满足需求的任务。任务中断恢复功能则可以在任务意外中断时自动恢复正在执行的任务，而执行机器人管理则是用于管理 RPA 机器人实际执行任务的，可以实现暂停任务的执行和状态查看等功能。通过这三大功能，RPA 任务处理模块可以让 RPA 机器人更加稳定可靠，而且还可以极大提高用户使用体验。

（4） 系统模块

系统模块是 RPA 中不可或缺的部分，包含有用户管理模块、资源管理模块和任务管理模块。其中，用户管理模块用于管理系统中的所有用户，包括权限、账户等重要信息，资源管理模块旨在对机器资源和系统服务资源进行有效的管理，而任务管理模块则可以对系统中的所有任务进行管理，用户可以针对任意一个任务进行删除或者停用操作，有效帮助对 RPA 系统的任务进行管理与控制。

3.3 RPA 任务负载微基准设计

为了更为深入地了解 RPA 任务负载特征，针对任务负载特征对系统平台进行优化，本文构造一个微型基准测试(Micro-Benchmark)以帮助测试系统实际负载情况。首先，本节介绍了测试任务背景及任务设计流程，然后介绍了任务负载数据集及测试平台。最后针对设计好的任务负载场景及任务进行测试，得出了具体的 RPA 任务负载情况，为实际负载场景提供了数据支撑。

3.3.1 任务负载设计

本文在制造业、财务、人力资源、物流及在线教育五大领域设计了一系列常见任务，共计 23 种。且此部分任务通常具有操作复杂、操作时间长、出错率较高的特点，较为适合 RPA 应用场景。

在制造业领域中，本文针对生产环节设计了如下五个任务：采购、入库、生产、销售及开票。由于目前办公自动化常利用 Excel 收集并处理数据，因此本文利用 Excel 作为输入源对上述五个任务进行设计。相关任务流程如图 3-6 所示：

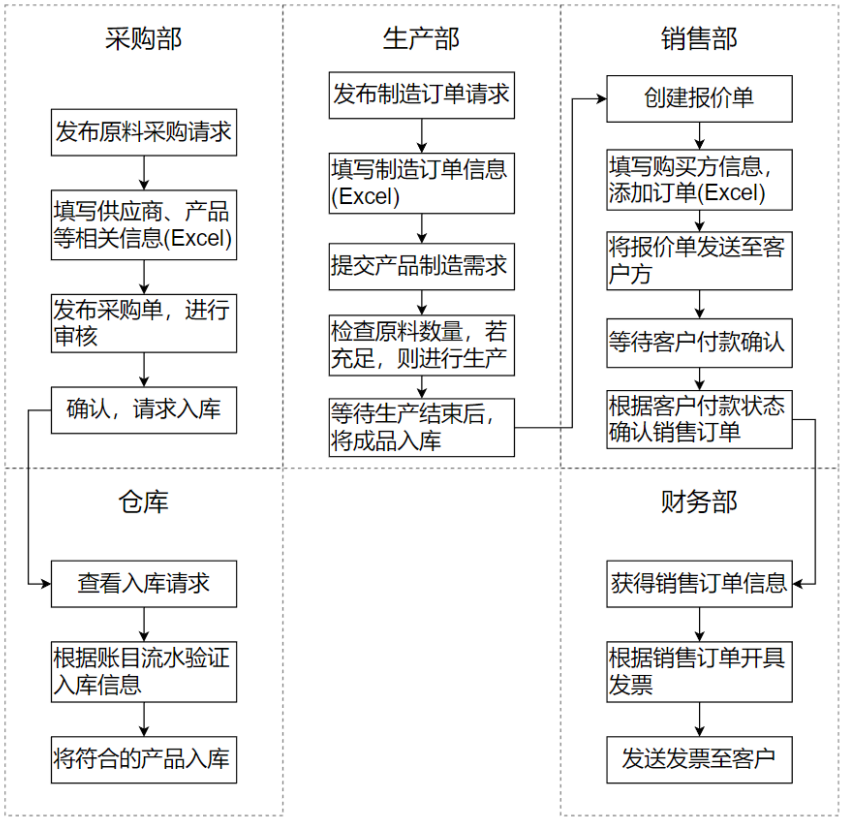


图 3-6 制造业任务设计图

财务领域中，本文设计了发票真伪检验、自动填写收据、自动催收账单、工资税计算及批量报销五种任务，其中，发票真伪检验与自动填写收据任务使用了 OCR 和 NLP 技术，对图片进行识别并自动解析表单。如图 3-7 所示。

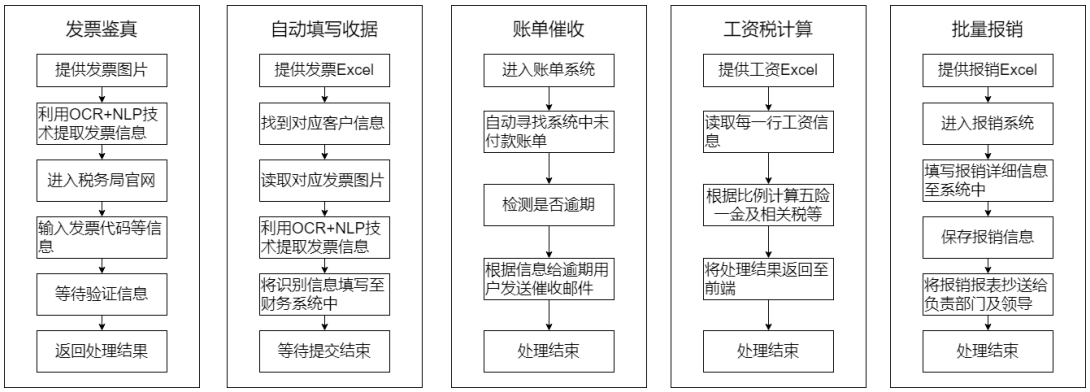


图 3-7 财务领域任务设计图

人力资源领域中，本文设计了自动录入人员信息、自动检测招聘申请、自动检测员工缺勤、员工设备报修处理与学历验证五种任务。其中自动录入人员信息以 OCR 与 NLP 技术自动分析入职者信息。如图 3-8 所示。

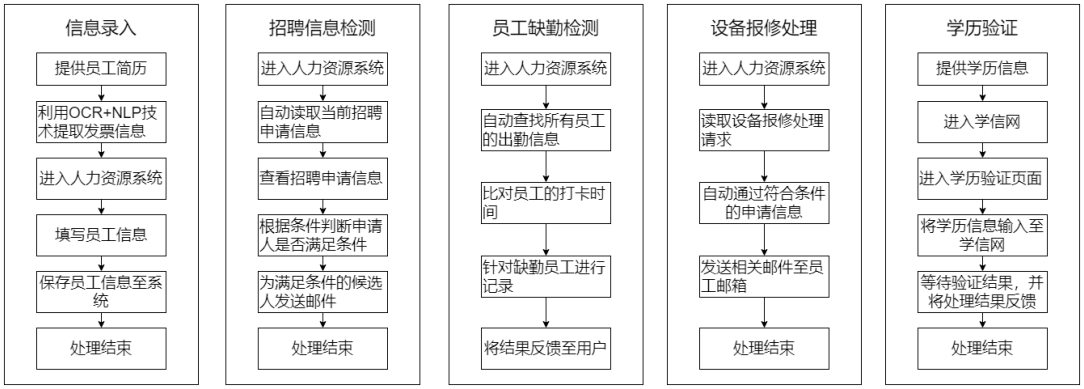


图 3-8 人力资源领域任务设计图

物流行业中，本文设计了驾驶证自动登记、货运状态同步、异常件告警、库存检测与驾驶证自动验证五种任务。其中，驾驶证自动登记通过 OCR 与 NLP 技术识别并登记。如图 3-9 所示。

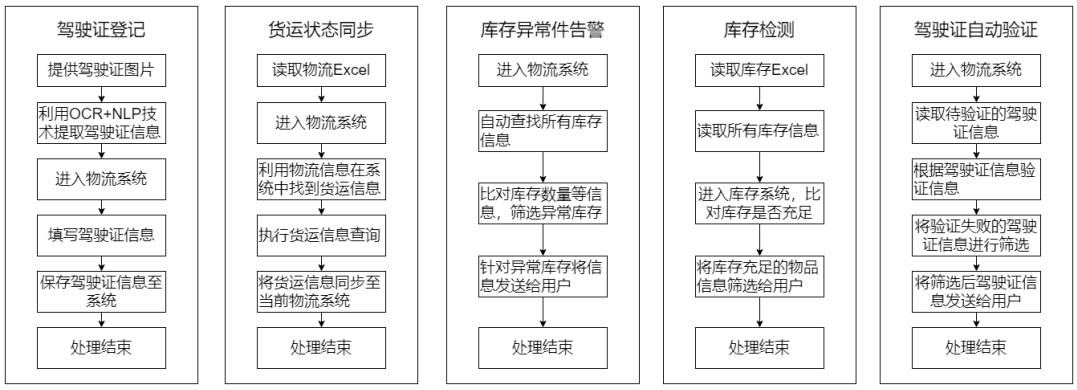


图 3-9 物流领域任务设计图

在线教育领域中，本文设计了课程自动注册、自动录入学生成绩与自动注册学生账户三种任务。如图 3-10 所示。

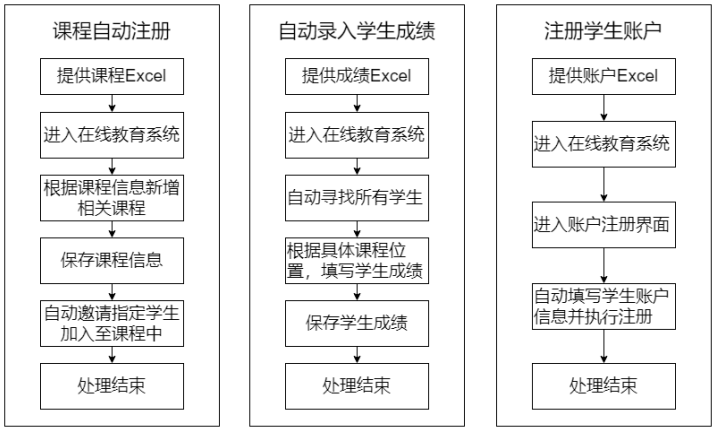


图 3-10 在线教育领域任务设计图

根据上述五大领域 23 种不同的任务设计图，本文最终得出的任务设计表如

下表 3-1 所示：

表 3-1 任务设计表

行业	场景
制造业	采购
	入库
	销售
	开票
	生产
财务	发票真伪检验(OCR+NLP)
	自动填写收据(OCR+NLP)
	自动催收账单
	工资税计算(Excel)
	批量报销
人力资源	自动录入人员信息(OCR+NLP)
	自动检测招聘申请
	自动检测员工缺勤
	员工设备报修处理
	学历验证
物流	驾驶证自动登记(OCR+NLP)
	货运状态同步
	异常件告警
	库存检测
	驾驶证自动验证
在线教育	课程自动注册
	自动录入学生成绩
	自动注册学生账户

为了任务负载探索的真实有效性，其中，制造业、财务、人力资源、物流四个场景基于 Odoo 13 开源平台进行测试。在线教育场景基于 MOODLE-3.11.11 开源平台进行测试。

由于本文的 RPA 云服务系统面向的场景均为实际场景，不同种类的数据集对平台并不会产生较大的影响，因此本章设计的所有任务负载中，所需的测试数据集均以自建的方式进行测试。其中，所有涉及到提供 Excel 数据源的任务均随机生成所需数据，且每个 Excel 文件中包含 10 条数据。以此作为 RPA 任务负载数据源。

3.3.2 负载评价指标

本文的 RPA 云服务系统面向网页自动化场景，并且所有的用户任务均在服务端以 SaaS(Software as a Service)的模式运行。因此，为了有效评价负载任务的

资源消耗情况，本文对任务负载的 CPU、内存、网络 IO 三大指标进行监测，以衡量服务器资源的消耗情况，设计一个任务负载集，如 3.3.3 节所示。

根据多领域不同任务的负载集合，本文使用了三个指标衡量在实际并发场景下的系统性能。分别如下所示：

（1） 任务执行代价：表示机器在任务执行过程中，每秒的预计执行代价累加值。该指标反映了当前该系统执行任务的能力，该值越低，其任务执行能力越强。

（2） 任务执行时间：表示从第一个任务提交到最后一个任务执行结束所消耗的时间。该指标反应了系统的任务执行效率。

（3） 负载均衡度：表示将所有任务分配至系统中后，集群资源负载请况。该指标反映了系统中任务调度算法的调度能力。

由于上述三个指标中，任务执行代价和任务执行时间两个指标与服务质量(QoS)直接相关，但负载均衡度为间接相关。本文作任务调度优化的目标是提升服务质量，因此在后续的系统测试指标对比中，本文以任务执行代价、任务执行时间、总优化目标调度代价进行对比。上述对比指标中，总优化目标调度代价包含了负载均衡度，并能够在整体任务调度层面展示调度结果，因此作了上述的替换。

3.3.3 任务负载集

为了进一步探索相关任务负载情况，本文针对表 3-1 设计好的任务以单任务单节点的方式执行，并收集对应任务的性能数据。具体的相关测试机器信息如表 3-2 所示。

表 3-2 硬件环境参数表

配置名称	参数
CPU	Intel(R) Xeon(R) CPU E5-2630 v2
硬盘	戴尔 PERC H710 600GB
操作系统	CentOS 7.1

基于以上硬件环境，本文针对每个任务在 CPU、内存、网络三个性能指标方面进行了实际性能的数据采集，下表 3-3 为 23 个 RPA 任务的性能数据表：

表 3-3 RPA 任务性能数据表

领域	场景	平均 CPU 利用率	内存占用	平均网络 IO	最高 CPU 利用率	执行时间
制造业	采购	7.469%	166.39MB	13.751Mbps	12.709%	31s
	入库	6.704%	219.39MB	31.686Mbps	9.802%	15s
	销售	6.277%	156.14MB	8.698Mbps	8.79%	60s
	开票	7.127%	164.98MB	14.308Mpbs	9.70%	32s
	生产	6.589%	151.14MB	9.953Mbps	10.21%	46s
财务	发票真伪检验	13.15%	1025.56MB	9.518Mbps	28.36%	63s
	自动填写收据	9.41%	837.43MB	10.192Mbps	28.42%	57s
	自动催收账单	6.67%	113.37MB	34.891Mbps	7.34%	14s
	工资税计算	2.56%	1MB	0Mbps	2.56%	1s
	批量报销	7.11%	142.07MB	10.172Mbps	9.91%	44s
人力资源	自动录入人员信息	11.10%	1701.99MB	3.949Mbps	51.65%	149s
	自动检测招聘申请	7.04%	121.19MB	14.857Mbps	10.92%	28s
	自动检测员工缺勤	7.31%	124.32MB	13.765Mbps	10.76%	29s
	员工设备报修处理	8.17%	117.75MB	33.72Mbps	9.52%	14s
	学历验证	10.62%	172.18MB	16.325Mbps	11.72%	30s
物流	驾驶证自动登记	12.40%	215.73MB	6.498Mbps	27.35%	74s
	货运状态同步	11.38%	252.16MB	11.53Mbps	14.21%	58s
	异常件告警	8.11%	120.75MB	7.602Mbps	11.55%	29s
	库存检测	7.12%	117.91MB	6.988Mbps	10.72%	27s
	驾驶证自动验证	11.40%	166.77MB	17.893Mbps	13.20%	25s
在线教育	课程自动注册	6.05%	190.16MB	1.915Mbps	9.24%	87s
	自动录入学生成绩	4.04%	124.59MB	1.382Mbps	6.59%	32s
	自动注册学生账户	6.55%	115.48MB	1.074Mbps	11.71%	189s

基于表 3-3 的实际测试数据结果，可以分析得出各领域下不同种类的任务负载特点，并根据 CPU 负载、内存负载、网络负载对不同类型的任务进行归类，分为 CPU 高密集、CPU 中密集、CPU 低密集，内存高密集、内存中密集、内存密集，网络高密集、网络中密集、网络低密集多种任务类型。具体分类如表 3-4 所示。

根据表 3-4 的结果可知，在 RPA 常见领域的不同种类任务中，任务负载是不均衡的状态。由于本文系统实际的任务执行过程均在服务端进行，若没有一个有效的任务调度算法，则会直接导致云端服务器出现资源负载不均衡的现象，影响用户任务的执行时间，降低服务质量。因此，针对此种负载，本文需要对任务调度进行优化，以保证任务执行效率，提高服务质量。本文将利用上述任务负载对本文的 RPA 云服务系统进行测试，并作任务调度优化。

表 3-4 RPA 任务负载特性表

领域	场景	CPU 密集	内存密集	网络密集	输入
制造业	采购	低	中	中	Excel
	入库	低	中	高	无
	销售	低	中	低	Excel
	开票	低	中	中	无
	生产	低	中	低	Excel
财务	发票真伪检验	高	高	低	图片
	自动填写收据	中	高	中	Excel
	自动催收账单	低	中	高	无
	工资税计算	低	低	低	Excel
	批量报销	低	中	中	Excel
人力资源	自动录入人员信息	高	高	低	图片
	自动检测招聘申请	低	中	中	无
	自动检测员工缺勤	低	中	中	无
	员工设备报修处理	低	中	高	无
	学历验证	中	中	中	字符串
物流	驾驶证自动登记	高	中	低	图片
	货运状态同步	中	中	中	Excel
	异常件告警	低	中	低	无
	库存检测	低	中	低	Excel
	驾驶证自动验证	中	中	中	无
在线教育	课程自动注册	低	中	低	Excel
	自动录入学生成绩	低	中	低	Excel
	自动注册学生账户	低	中	低	Excel

3.4 本章小结

本章首先整体介绍了 RPA 云服务系统的设计需求，包括功能性需求与非功能性需求，然后阐述了系统的总体设计与总体架构，对功能模块展开了一定的描述，最后对 RPA 云服务系统的任务负载探索做了相关介绍，在设计了 23 中任务场景后，针对任务进行了负载测试，得出了 5 大领域中 23 个任务的详细负载情况，并得出了 RPA 任务负载是不均衡的结论，针对 RPA 任务负载不均衡的现象，本文需要针对系统做任务调度优化。本章提供的微基准任务负载集可以有效地为后续实际并发任务调度实验提供数据基础，为后续实际并发任务场景下的调度优化提供支撑。

第4章　SN-PSO：一种改进的 RPA 云任务调度优化算法

基于 3.3 节的负载测试结果，本章针对负载不均衡的现象，采用改进粒子群算法 SN-PSO 针对云任务调度进行优化。首先，介绍本文面向的 RPA 云环境下的任务调度问题，并根据问题背景构建了多目标优化任务调度数学模型。然后，为了便于计算，通过参考多目标优化任务调度数学模型，对标准粒子群中的适应度函数进行了设计，并针对标准粒子群算法在求解调度目标时存在的粒子编码问题进行了离散化处理。最后，提出了基于滑动窗口的变异策略及基于非支配解集的优化策略，构建一种改进粒子群算法 SN-PSO。

4.1 问题描述与分析

在 RPA 云环境中，用户实际面对的是虚拟机资源，而云任务调度是为了将用户从虚拟机发送的任务分配到具体的任务执行节点上，该任务执行节点具有实际的物理资源[67]。因此，合理的任务调度策略可以降低总任务完成代价，尽可能做到物理资源的负载均衡，提高任务执行的效率，降低任务执行时间。

4.1.1 问题描述

在本文算法调度场景中，每一个 RPA 任务都以工作流的形式执行，每个任务包含若干个组件，并组成一个有向无环图(DAG)，以本文的 RPA 云服务系统中的采购流程为例，如图 4-1 所示。

图 4-1 采购流程的组件示例图

如图 4-1 所示，在本文的实际应用场景下，其面向场景主要为 Web 端，大多数组件的功能以模拟用户操作网页浏览器为主。因此，组件与组件之间的执行存在拓扑关系与依赖关系，本文不对完整的任务流拆分成多个子任务进行优化。综上所述，本文的调度算法以调度一个完整的 RPA 任务流程为任务调度目标，

通过对任务与集群间机器的分配，实现任务调度优化。

根据上述背景，可以定义如下的任务调度环境：

假设，当前 RPA 云服务系统集群环境中拥有 M 台机器，用集合$MP = \{m_1, m_2, \dots, m_M\}$表示。待处理任务序列数量为 N，用集合$TP = \{t_1, t_2, \dots, t_N\}$表示，则当前集群环境中：

（1） 对于任意一台机器m_i，该机器拥有 CPU、内存、网络带宽资源，分别使用O_c^i，O_m^i，O_n^i表示，即向量$O^i = (O_c^i, O_m^i, O_n^i)$。

（2） 云环境中任意一台机器m_i已经使用的 CPU、内存、网络资源表示为U_c^i，U_m^i，U_n^i，即向量$U^i = (U_c^i, U_m^i, U_n^i)$。此机器的剩余资源可以通过$O^i - U^i$计算得到，其剩余资源分别表示为$R_c^i$，$R_m^i$，$R_n^i$。

（3） 根据任务序列集合 TP 与机器集合 MP 组成一个具有 N 个维度的机器分配向量$X_i = (x_1, x_2, \dots, x_N)$，其中：

$$x_i = \begin{cases} 1，\ 任务i分配到第1\ 台机器上 \\ 2，\ 任务i分配到第2\ 台机器上 \\ \dots \\ M，\ 任务i分配到第M台机器上 \end{cases} \quad (4-1)$$

（4） 根据机器分配向量X_i对需要执行的任务模拟分配至对应机器上，并计算当前机器分配代价，记为F，利用任务调度算法找出使得 F 最小的机器分配向量X_i，即可完成一次最优调度。

根据以上步骤，任务调度算法每次接受到达的任务时，将利用当前云环境中的资源与任务资源进行比对和搜索，找出一种方案使得整体分配代价最优，此时将得到一个机器分配向量X_i，作为最优任务调度策略。

4.1.2 多目标优化任务调度数学模型

在云任务调度环境下，云计算提供商关注成本，用户关注服务质量。对云计算提供商而言，希望能够最大程度的利用每一台服务器的资源，为更多的用户提供服务。对用户而言，希望获得更好的服务体验，而一个好的服务体验取决于执行机器的资源是否充足，充足的资源能够确保任务被快速执行以降低总执行任务的时间。本文针对上述需求，构建了负载均衡度以代表云提供商集群资源使用情况，构建了任务执行代价以代表用户每次执行任务时所需理论代价值，构建多目

标优化数学模型，以寻求集群负载与执行代价间的平衡。

假定，当前可选调度方案一共具有 N 种，其任务调度指标的具体定义如下所示：

定义 4.1 任务执行代价

若第 j 台机器待分配的任务集为$TP' = \{t_1, t_2, \dots, t_k\}$。集合中对于任意一个任务$t_i$，记其执行所耗费的 CPU、内存、网络资源为$C_c^i$，$C_m^i$，$C_n^i$，其执行需要的时间为：$D^i$。则$TP'$所消耗的 CPU、内存、网络资源分别如下所示：

$$C_c = \sum_{i=1}^{k} C_c^i \tag{4-2}$$

$$C_m = \sum_{i=1}^{k} C_m^i \tag{4-3}$$

$$C_n = \sum_{i=1}^{k} C_n^i \tag{4-4}$$

那么该任务集在机器 j 上的执行代价为：

$$Exec_j = \sqrt{\left(\frac{C_c}{R_c^j}\right)^2 + \left(\frac{C_m}{R_m^j}\right)^2 + \left(\frac{C_n}{R_n^j}\right)^2} * \sum_{i=1}^{k} D^i \tag{4-5}$$

$$Exec = \sum_{j=1}^{M} Exec_j \tag{4-6}$$

定义 4.2 集群负载均衡度

沿用定义 4.1 的规定，第 j 台机器上任务集消耗的 CPU、内存、网络资源为：C_c，C_m，C_n。那么，对于第 j 台机器上的负载均衡度为：

$$Balance_j = \sqrt{\left(C_c + U_c^j\right)^2 + \left(C_m + U_m^j\right)^2 + \left(C_n + U_n^j\right)^2} \tag{4-7}$$

则，集群的负载均衡度为：

$$Balance = \sum_{j=1}^{M} Balance_j \tag{4-8}$$

定义 4.3 多目标优化数学模型

1) 集群负载均衡度最优，可以表示为：

$$F_1 = Min\{Balance_i\}\ (i = 1,2, \dots, N) \tag{4-9}$$

2) 集群执行代价最优，可以表示为：

$$F_2 = Min\{Exec_i\}(i = 1,2,\dots,N) \quad (4-10)$$

3) 单节点负载均衡度最优，可以表示为：

$$F_3 = Min\left\{\left(Max\{Balance_j\}\right)_i\right\}(j = 1,2,\dots,M)(i = 1,2,\dots,N) \quad (4-11)$$

4) 多目标优化问题数学模型描述可以表示为：

$$MinF = \{F_1, F_2, F_3\} \quad (4-12)$$

4.2 粒子群适应度函数设定与粒子编码策略

在上一节的问题描述中，负载均衡度与任务执行代价为理论场景下的定义，但是在实际生产环境中，除了机器资源与待分配任务所消耗资源外，还需要考虑已分配的任务所占用资源。尤其是在高并发场景下，多个任务分配至不同机器后，其资源使用率在一段时间内逐渐提高，难以立刻体现资源使用情况，因此需要考虑已分配的任务所占用的资源状态。本文针对适应度函数进行了调整，以避免上述存在的问题。

4.2.1 适应度函数

假定，第 j 台机器上已经分配的任务集$P' = \{t'_1, t'_2, \dots, t'_k\}$，集合中对于任意一个任务$t'_i$，记其执行所耗费的 CPU、内存、网络资源为$C'^i_c$，$C'^i_m$，$C'^i_n$。

在任务集P'上，所有已分配的任务占用的总 CPU、内存、网络资源分别为：

$$C'_c = \sum_{i=1}^{k} C'^i_c \quad (4-13)$$

$$C'_m = \sum_{i=1}^{k} C'^i_m \quad (4-14)$$

$$C'_n = \sum_{i=1}^{k} C'^i_n \quad (4-15)$$

结合 3.2 节的数学模型描述，需要优化的三个目标函数分别如下所示：

$$f_1(x) = \sum_{j=1}^{M}\left(\sqrt{\left(\frac{C_c + C'_c}{R^j_c}\right)^2 + \left(\frac{C_m + C'_m}{R^j_m}\right)^2 + \left(\frac{C_n + C'_n}{R^j_n}\right)^2} * \sum_{i=1}^{k} D^i\right) \quad (4-16)$$

式(4-16)中表示集群任务执行代价总和。

$$f_2(x) = \sum_{j=1}^{M}\sqrt{\left(C_c + C'_c + U^j_c\right)^2 + \left(C_m + C'_m + U^j_m\right)^2 + \left(C_n + C'_n + U^j_n\right)^2} \quad (4-17)$$

式(4-17)中表示集群负载均衡度总和。

$$f_3(x) = Max\left\{\sqrt{\left(C_c + C_c' + U_c^j\right)^2 + \left(C_m + C_m' + U_m^j\right)^2 + \left(C_n + C_n' + U_n^j\right)^2}\right\} \quad (4-18)$$
$$(j = 1,2,\dots,M)$$

式(4-18)中，$f_3(x)$表示的是集群中单台服务器的最高负载均衡度指标。

在求解多目标优化问题场景下，共计拥有 N 种任务调度方案，则粒子群算法的适应度函数如式(4-19)所示：

$$F(x) = Min\{w * f_1(x_i) + 0.5 * (1-w) * (f_2(x_i) + M * f_3(x_i))\} \quad (4-19)$$
$$(i = 1,2,\dots,N)$$

其中，w 为当前多目标优化问题场景下的调整因子，当 w=0 或者 w=1 时，求解问题将自动转化为单目标优化问题。

综上，粒子群算法使用式(4-19)求解函数最小值，即可得出在当前资源状态下的最优任务分配方案。

4.2.2 粒子编码策略

根据 4.1 节的问题描述，任务序列集合与机器集合可以构成一个机器分配向量，因此，粒子群算法中，任意粒子位置向量$X_i^t = (x_{i1}^t, x_{i2}^t, x_{i3}^t, \dots, x_{iN}^t)$代表着一个 N 维的机器分配向量，任意维度$x_{ij}^t$表示将第 j 个任务分配至第$x_{ij}^t$台服务器上。但是，标准粒子群中的位置向量中任意一个维度的值均为连续值，而机器分配向量要求为离散值。所以，目前的关键问题在于将粒子位置转化为一个离散且具有实际意义的机器编号，最终得到机器分配向量。本文针对连续位置信息做了如下离散化处理过程：

（1） 任意一个维度x_{ij}^t取绝对值，即$x_{ij}^t = |x_{ij}^t|$，得到一个新的位置向量X_i^t

（2） 利用公式(4-20)对位置向量进行离散化处理：

$$Dis(X_i^t) = \left(floor(x_{i1}^t), floor(x_{i2}^t), \dots, floor(x_{iN}^t)\right) \quad (4-20)$$

根据上述处理，即可得到一个具有离散值的机器分配向量。

以四维位置向量举例，某时刻粒子的位置为(0.3, 1.1, -0.4, 2.9)。经过离散化处理后得到的向量为(0, 1, 0, 2)，表 4-1 为对应的任务分配策略。

表 4-1 任务分配策略表

任务编号	1	2	3	4
机器编号	0	1	0	2

表 4-1 中，任务编号为 1 的任务被分配到了机器编号为 0 的虚拟机上，任务编号为 2 的任务被分配到了机器编号为 1 的虚拟机上，以此类推。

4.3 基于 SN-PSO 的 RPA 任务调度算法

基于以上的适应度函数与粒子编码策略，本文中的粒子群优化算法主要面向的是离散空间中的搜索问题。以二维搜索空间为例，如下图所示。

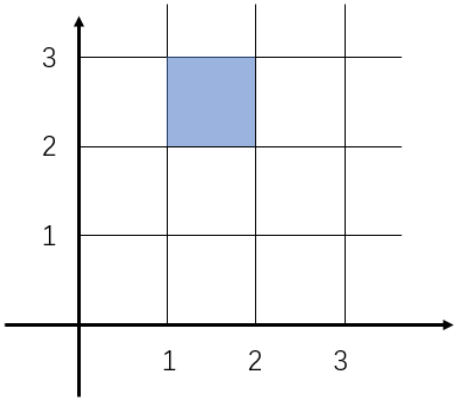


图 4-2 二维离散空间示意图

图中的阴影部分(不含边界)中任意一个点所代表的机器分配向量均为(1, 2)，因此粒子群的收敛精度问题在此搜索环境下重要性较低，同时，离散空间将放大粒子群算法易陷入局部最优解的缺陷。若算法陷入一个较差的局部最优解并无法跳出该状态，最终的调度效果将会较差，无法有效地利用集群资源，甚至延长任务响应时间，降低任务执行效率并影响 RPA 云服务提供商的服务质量。为了帮助粒子群算法跳出局部最优解，许多学者提出了不同的变异策略，用于改善粒子群的这个缺陷。以随机变异与基于聚集度变异两种常见的策略为例，上述两种变异策略具有以下问题：

（1）随机变异具有不可预测性，极有可能破坏向较优方向迭代的粒子位置，导致降低算法的搜索效果。

（2）基于聚集度变异策略可以有效检测到陷入局部最优的粒子，但是此策略会改变某一区域内所有粒子的位置，依旧可能破坏迭代良好的粒子位置。

综上，本章在离散空间下，利用滑动窗口检测每个粒子自身的迭代状态，并帮助粒子有效跳出局部最优解，避免变异策略对迭代良好的粒子造成影响。同时利用非支配解集增强种群多样性以提高算法搜索能力。因此，本章提出了一种基于滑动窗口与非支配解集的粒子群优化算法 SN-PSO。

4.3.1 基于滑动窗口的变异策略

在标准粒子群算法中，全局搜索与局部搜索能力的平衡是非常重要的。由于粒子群算法中粒子更新的方式，导致在粒子收敛过程中易于陷入局部最优状态，为了解决该缺陷，本文引入了变异策略。但是传统变异策略具有随机性与不可控性，极有可能破坏收敛方向更好的粒子位置，导致错过全局最优解。而基于粒子聚集状态的变异策略有概率会破坏收敛精度。因此，本文提出了一种基于滑动窗口的变异策略，使得每个个体都能专注于自身的适应值变化状态。在该方法中，滑动窗口与变异算子是重要的组成部分。

（1） 滑动窗口

在粒子群算法中，每个粒子 i 均维护第 i 个滑动窗口$Window_i$，每个滑动窗口都有起始观测点 Start 与终止观测点 End，且滑动窗口的最大观测区间为 WindowSize。每个滑动窗口观测的对象为每个粒子从第 Start 代至第 End 代的适应度值。当满足$End - Start < WindowSize$条件时，粒子群算法每经过一次迭代更新后，终止观测点 End 向后移动一个单位；当满足$End - Start = WindowSize$条件时，粒子群算法每经过一次迭代更新后，终止观测点与起始观测点都将向后移动一个单位，以保证观测空间为最大设定值：WindowSize。

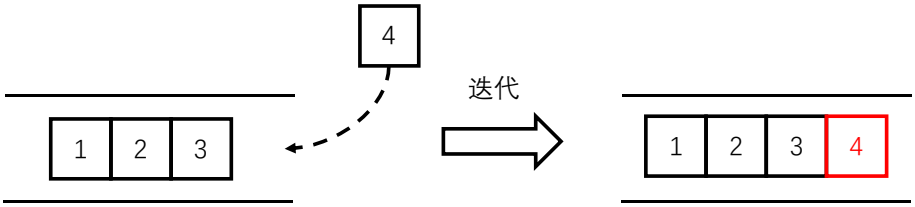


图 4-3 观测空间不足 5 的滑动窗口迭代示例图

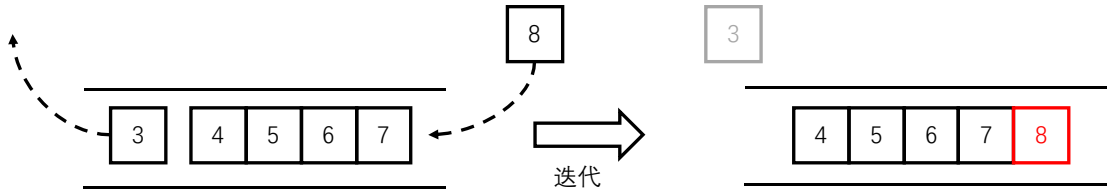


图 4-4 观测空间等于 5 的滑动窗口迭代示例图

在这种策略的影响下，每个粒子只需关注窗口期内的适应度值变化状态，并记某个粒子的滑动窗口中观测到的最小适应度值为 mFit，粒子个体最优适应度值为 pFit，当且仅当满足下列两个条件时，粒子执行变异操作：

- $End - Start = WindowSize$
- $mFit > pFit$

当变异操作执行结束后，粒子对应的滑动窗口将被重置，此时的起始观测点与终止观测点位置相同。同时，计算每个粒子的适应度值，记为 nFit，将当前粒子的个体最优适应度值P_i^t替换为 nFit。

（2） 变异算子

在滑动窗口观测到粒子需要执行变异操作后，对当前粒子对应的位置向量$X_i^t = (x_1, x_2, \dots, x_n)$计算一个具有 n 个元素的适应度集合$FP = \{fit_1, fit_2, \dots, fit_n\}$。其中，$fit_i$表示将位置向量$X_i^t$中第 i 维移除后，剩余 n-1 维向量计算得到的适应度值。

10维向量$\vec{X} = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]$

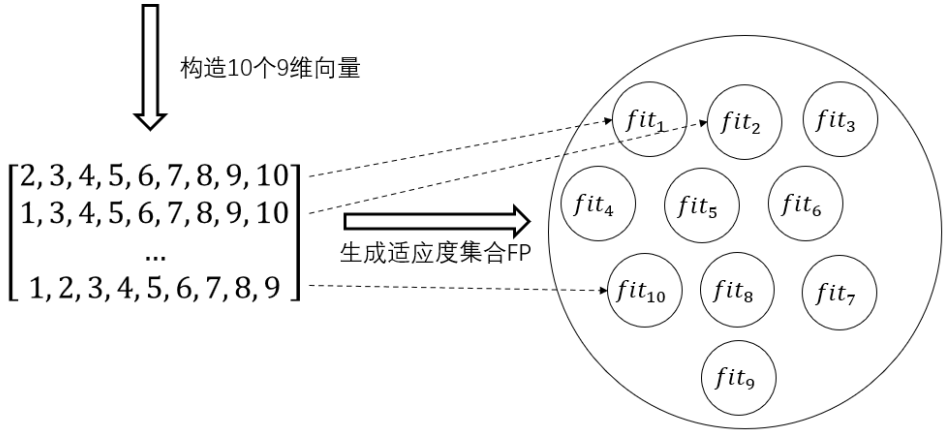


图 4-5 集合 FP 生成示例图

随后对 FP 从小到大排序，取前 50%的集合元素，记此部分对应向量维度的集合为 DP，可以近似认为，DP 集合中的维度较大程度地影响了适应度，使得适应度值过高。因此，这部分维度将被重新初始化为一个随机位置：

$$x_i = rand(lbound_i, rbound_i)\ (i \in DP) \qquad (4-21)$$

其中，$lbound_i$和$rbound_i$表示第 i 维的上边界和下边界。这种变异方式保证了子代中较优维度被保留，能够平衡个体的局部搜索能力与全局搜索能力。

（3） 伪代码

综上，基于滑动窗口的变异策略伪代码如算法 4-1 所示。

4.3.2 基于非支配解集的优化策略

标准粒子群中最优解维护只考虑了帕累托(Pareto)解集中的支配解情况，并未考虑到非支配解。本文所考虑的粒子群需要解决离散空间下的最优解问题，粒子群维护的种群最优解应当是一个解集。因此，本文对标准粒子群解集维护过程

算法 4-1: 基于滑动窗口的变异策略

输入: 粒子当前适应度值 nFit，粒子个体最优适应度值 pFit，粒子的滑动窗口 WP，粒子原始位置向量X

输出: 粒子位置向量X^*

1:将 nFit 加入到 WP 中, $X^* = X$

2: mFit= MIN(WP)　//从 WP 中获得一个最小适应度值

3: *if* mFit > pFit:

4:　　用 X 生成适应度集合 FP

5:　　sort(FP)　//对 FP 元素排序

6:　　i=0.5*len(FP)　//获得集合 FP 长度的一半

7:　　*while* i>=0:

8:　　$DP \leftarrow FP_i$对应维度　//将第 i 个元素对应维度加入 DP 集合

9:　　i--

10:　*for* i=1…N:

11:　　if $i \in DP$:

12:　　$X_i^* = rand(lbound_i, rbound_i)$　//粒子第 i 维位置随机赋值

13:　　清空 WP, 重置起始观测点与终止观测点

14:　　pFit = nFit　//替换粒子个体最优适应度值

15: *else* $mFit \leq pFit$:

16:　　移动 WP 的终止观测点

17:　　若窗口观测大小与最大观测区间相等，移动 WP 起始观测点

18: *return* X*

进行改进，并加入了随机选择种群最优解的优化策略，以获得离散空间下的最优解。

（1） 个体最优与群体最优解的判断规则

在标准粒子群中，如果粒子当前个体最优解P_i^t对应的适应度值为$f(P_i^t)$，且满足个体最优适应度值比种群最优适应度值低的条件时，即$f(P_i^t) < f(G^t)$，则将种群最优解G^t更新为P_i^t，反之，不对种群最优解更新。

但是，当适应度值满足$f(P_i^t) = f(G^t)$条件时，难以指定P_i^t与G^t这两种方案中的最优解，两者之间处于非支配关系。在本策略中，需要维护一个非支配解集 GSet，用于存放所有的非支配解。

在非支配解存放至 GSet 之前，需要对非支配解向量通过公式(4-20)做离散化操作，再判断该支配解是否已经存在于 GSet 中，若当前解不存在于 Gset 中，则

算法 4-2: 基于非支配解集的优化策略

输入: 粒子种群$P = \{Particle^i\}(i = 0, 1, \dots, N)$, 种群原最优解 gs, 种群最优适应度 gf, 迭代计数器 ItCnt, 非支配解集$GSet = \{Sol^i\}$,策略阈值 RN

输出: 种群最优解gs^*

1:$gs^* = gs$

2: *for* i=1…N:

3: *if* $Particle^i_{fitness} < gf$: //如果当前粒子适应度比种群最优适应度小

4: $ItCnt = 0$

5: $GSet = \{Particle^i_{solution}\}$ //GSet 初始化为仅含粒子i的解的集合

6: $gf = Particle^i_{fitness}$

7: $gs^* = Particle^i_{solution}$ //将种群最优解更新为粒子i的解

8: *elif* $Particle^i_{fitness} = gf$:

9: $D_{Solution} \leftarrow Dis(Particle^i_{solution})$ //通过公式(4-20)离散化

10: *if* $D_{Solution} \notin GSet$:

11: $GSet \leftarrow D_{Solution}$ //将$D_{Solution}$加入到 GSet 中

12: *if* $ItCnt \% RN = 0$:

13: $RSol \leftarrow Rand(GSet)$ //从非支配解集中随机选取一个解Sol^i

14: $gs^* = RSol$

15: *return* gs^*

加入至解集中，反之无需更新 Gset。

（2） 种群最优解的随机选择策略

由于粒子群具有收敛速度快的特点，在经过若干轮迭代后，粒子群算法便已经成功收敛，种群最优解将不会发生较大改变。综上，可以设定一个触发随机选择策略的阈值 RN，并且使得 RN 略大于成功收敛轮次，每当经过 RN 轮时，触发随机选择策略，将当前种群最优解随机更换为 GSet 中任意一个非支配解，这样既可以保证在 RN 轮内正常收敛，又可以增强粒子群的全局搜索能力。

为了避免粒子群算法被随机选择策略影响搜索方向，规定，在粒子群算法中维护一个迭代计数器 ItCnt，当且仅当满足$ItCnt\ mod\ RN = 0$的条件时，执行随机选择策略。并且当种群最优解发生更新时，重置迭代计数器 ItCnt 与非支配解集 Gset。此外，在搜索后期，一旦 SN-PSO 算法触发了变异策略，则意味着此轮搜索结束，并且种群最优解并没有被更新。因此，非支配解可以对粒子群搜索方向进行扰动，增强算法寻求更优解的能力。综上，本文将 RN 设置为 4.3.1 小节

中滑动窗口大小相同的值。

（3） 伪代码

综上，基于非支配解集的优化策略伪代码如算法 4-2 所示。

4.4 基于 SN-PSO 的任务调度总体流程

本小节主要介绍 RPA 云服务系统中，使用基于 SN-PSO 算法的任务调度总体流程。如图 4-6 所示。

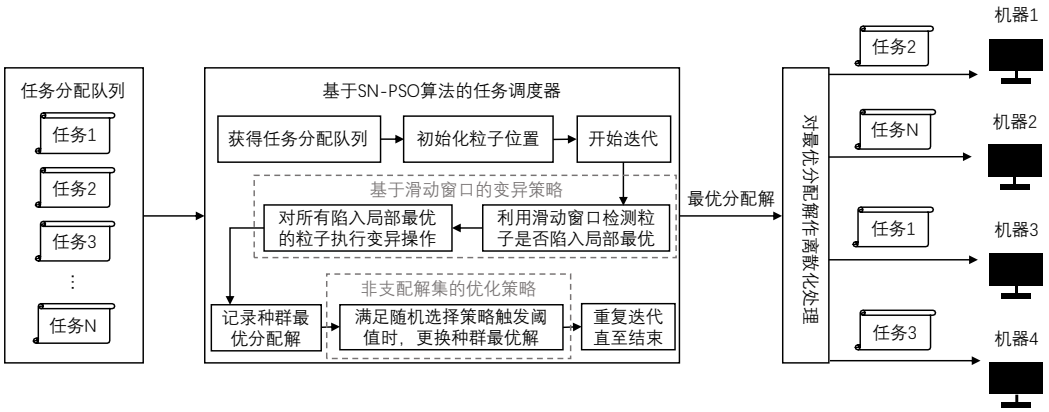


图 4-6 任务调度总体流程图

图 4-6 中，RPA 云服务系统中的任务调度器需要接收用户提交的 RPA 任务。在实际场景下，大量用户可能在短时间内会提交一个或多个任务，因此，本文将短时间内的所有任务加入到任务分配队列中，并将整个队列批次发送至任务调度器。任务调度器接受到任务分配队列后，将立刻启动 SN-PSO 任务调度算法，并执行初始化与迭代过程，利用基于滑动窗口的变异策略与基于非支配解集的优化策略对粒子群算法进行搜索优化，获得一种较好的种群最优分配解向量。最后对分配解向量作离散化处理，获得任务与实际执行机器间的对应关系，将任务推送至对应机器中，正式交予 RPA 云服务系统的执行器以执行用户任务。

4.5 本章小结

本章在基于粒子群算法的云任务调度优化问题上进行了分析与阐述。首先，介绍了云任务环境问题描述与多目标优化任务调度数学模型，明确了在本文系统中所处场景下的问题建模，基于本文的场景特性与实际调度环境构建了一个多目标数学模型。其次，介绍了粒子群适应度函数的具体设定与粒子群的粒子编码策略，明确了粒子群实际优化的目标函数并针对如何有效处理粒子搜索到的具体解进行了展开描述。最后，针对粒子群核心改进策略进行了阐述，主要包含基于滑

动窗口的变异策略及基于非支配解集的优化策略两个方案，并给出了相关的伪代码。

第5章　系统实现与测试

本章介绍 RPA 云服务系统的实现。首先，展示了 RPA 云服务系统各模块的实现截图，然后通过 RPA 云服务系统的效率对比实验，验证 RPA 云服务系统具备实际的应用价值。最后，对第四章提出的任务调度算法 SN-PSO 作实验评估与实验结果分析。

5.1 系统实现与测试

RPA 云服务系统的开发基于 Ruoyi 框架，采用了 Spring Cloud 开发框架进行微服务设计，基于 Windows 10 进行开发，CentOS 7 操作系统作为运行环境。前端使用 Vue 及 ElementUI 构建。本节将对 RPA 工作区模块、执行计划模块、RPA 任务处理模块及系统模块实现结果进行介绍与展示。

5.1.1 系统模块

用户需要通过用户登录才可进入到系统中使用系统模块，系统模块分为用户管理、资源管理、任务管理三大组成部分。

为了便于用户管理，用户管理中包含了系统登录模块，系统登录页面如图 5-1 所示。

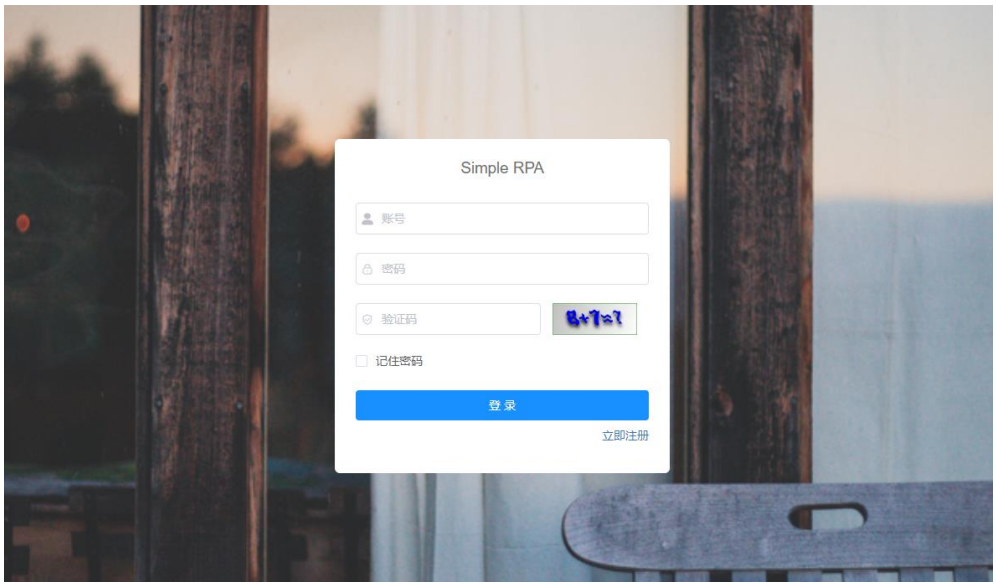


图 5-1 系统登录

本文为了完整展示所有的功能，以下所有的模块均以超级管理员的方式登入至系统中进行演示。普通用户无法使用系统管理功能，若普通用户直接访问系统

管理页面，系统则会拦截请求。

当用户登入至系统中后，平台的用户管理模块如图 5-2 所示。

图 5-2 用户管理页面

资源管理中包含资源监控及资源控制两个部分，具体页面如图 5-3 所示。图中，上半部分包含资源控制功能，用户可以点击停用按钮，使得所选虚拟机暂停接受 RPA 任务请求。同时，用户可以在每个卡片上看到虚拟机当前的 CPU、内存、网络资源使用率。图中下半部分监控了整个集群的资源使用情况及各节点资源使用情况的变化图。

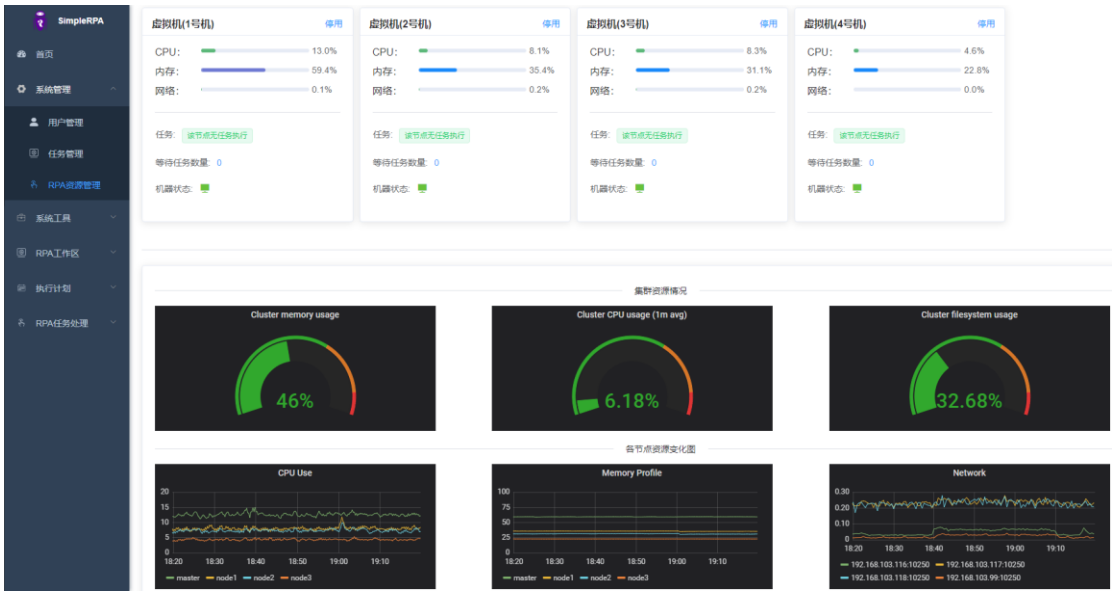


图 5-3 资源管理页面

任务管理可以控制本平台中所有用户的任务，包括编辑及删除。如图 5-4 所示。为了有效管理平台中的任务并监控任务不合理行为，超级管理员可以删除任意一个用户的任务，并且可以修改当前任务的用户归属。

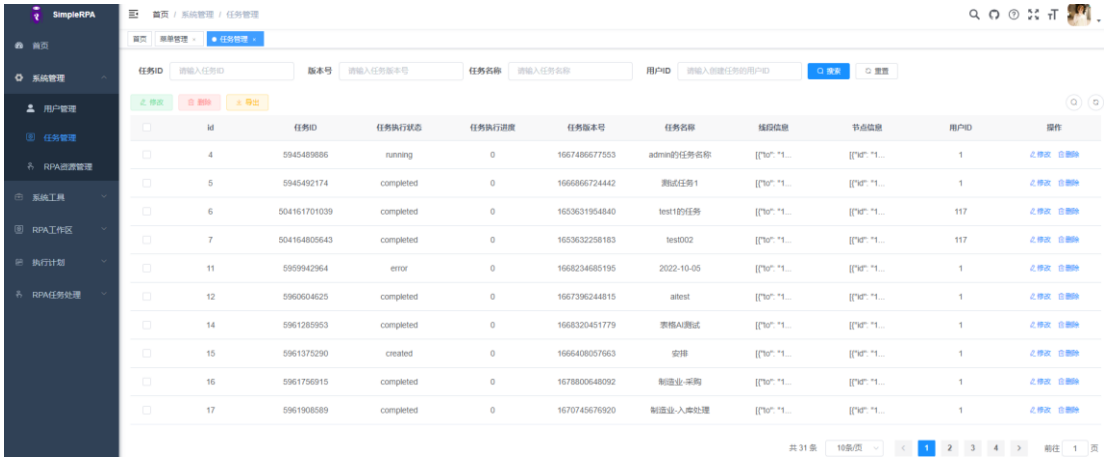


图 5-4 任务管理页面

5.1.2 RPA 工作区模块

RPA 工作区模块主要用于管理 RPA 任务、跳转至 RPA 任务编辑面板、RPA 任务可视化与流程编排三部分。

RPA 任务管理页面如图 5-5 所示。用户可点击新增按钮，为该账户创建一个新 RPA 流程，也可以修改指定任务的任务名称。此外，用户可以点击打开面板按钮，进入到对应 RPA 任务流程中，并对该任务进行编辑。在该页面还提供了任务流程可视化接口，用户点击可视化按钮，即可查看对应任务的执行可视化信息。

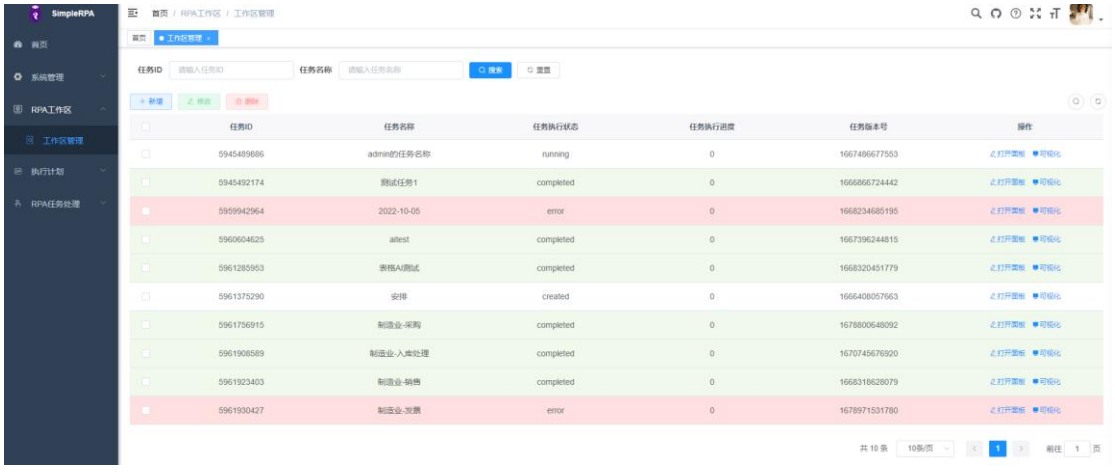


图 5-5 RPA 任务管理页面

RPA 任务可视化包含任务当前执行信息、历史任务执行信息及当前任务执行异常时出错组件信息。如图 5-6 所示。

图 5-6 RPA 任务可视化页面

用户可以通过 RPA 工作区管理页面中的打开面板功能，跳转至 RPA 任务流程编辑页面，如图 5-7 所示。系统集成 Easy-flow 框架，实现组件拖拽功能，提供了数据读取、数据处理、条件判断、循环组件、网页自动化组件、其他类型组件及 AI 增强组件七大类型，共计 42 个功能组件。用户可以根据自身需求，拖拽需要的组件，设计一个符合自身需求的任务流程，而无需受限于原系统设计好的任务流程。

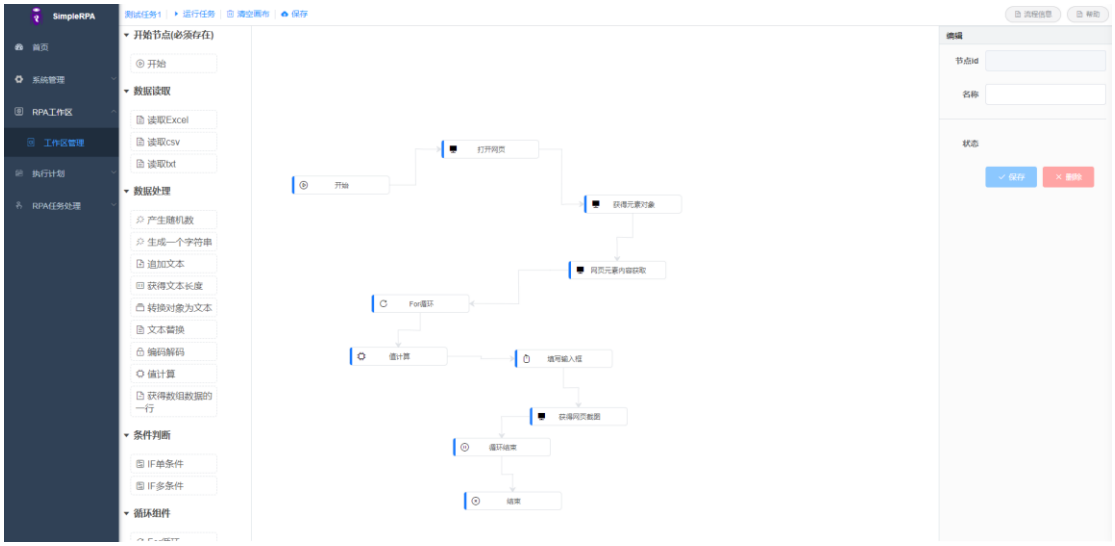


图 5-7 RPA 流程编辑页面

为了进一步说明 RPA 系统在实际场景下的应用，本文以表 3-3 中的制造业-销售任务和人力资源-自动录入人员信息为例，其中销售任务为一种典型的业务 RPA 任务，其对应的大致任务过程如图 3-6 所示，自动录入人员信息任务为带有 AI 功能的任务，其对应的大致任务过程如图 3-8 所示。

在销售任务中，所需要使用到的数据源如图 5-8 所示。经过 RPA 系统运行后，分别需要产生报价单与发票两种类型的数据信息，如图 5-9 与图 5-10 所示。

客户	发票地址	交货地址	到期时间	付款条款	产品名称	数量
Lumber	Lumber Inc, Lorraine Douglas	Lumber Inc, Lorraine Douglas	2022/11/20	立即付款	白板笔	1
					大书桌台	2
					桌上收纳盒	1

图 5-8 销售任务数据源

图 5-9 销售任务报价单图

图 5-10 销售任务客户发票图

在自动录入人员信息的任务中，所需要使用到的数据源如图 5-11 所示。由于需要具体的人员信息，因此本文截取了部分简历信息并隐藏了隐私信息。RPA

系统将根据此份简历自动识别，并填写入职人员信息。如图 5-12 所示。

图 5-11 自动录入人员信息任务数据源(截取)

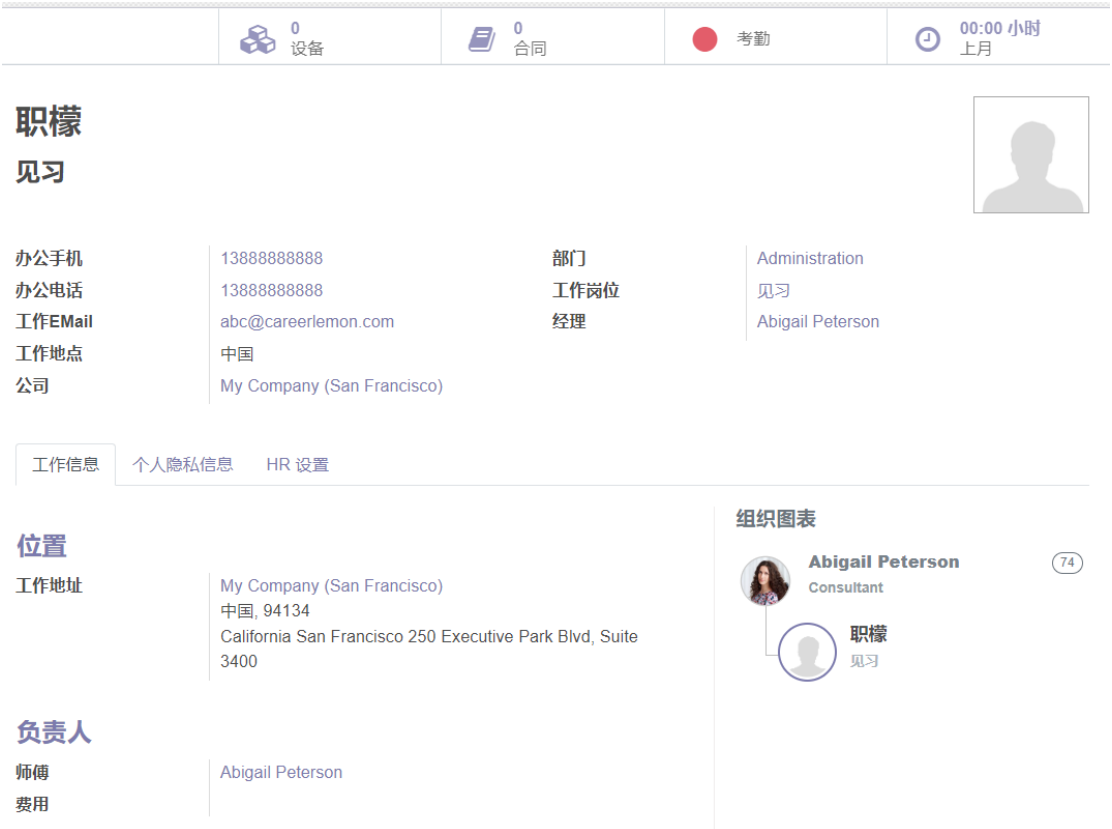


图 5-12 自动录入人员信息任务处理结果

综上，本文的 RPA 云服务系统在实际应用场景下具备一定的实际意义与应用价值。

5.1.3 执行计划模块

执行计划模块主要用于对 RPA 任务设定一个定时执行计划，系统会自动根

据用户指定的定时任务后台自动执行。用户可以使用设定执行计划功能设定定时触发规则，设定执行计划页面如图 5-13 所示，并在执行计划管理中查看已经设定完成的所有任务信息，如图 5-14 所示。

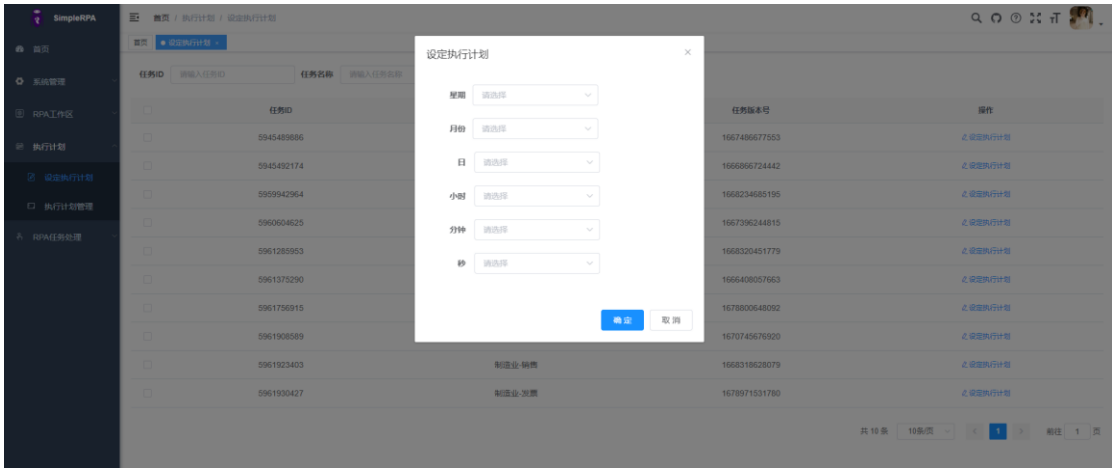


图 5-13 设定执行计划页面

图 5-14 执行计划管理页面

5.1.4 RPA 任务处理模块

RPA 任务处理模块主要提供协同编辑、任务中断恢复、日志管理功能。

其中，用户可以使用协同编辑功能导入已经存在的任务，导入后的任务包含创建信息、名称、协同编辑状态的信息，用户可以决定当前任务是否开启协同编辑状态。并且用户可以通过云协同编辑空间查看所有用户分享的任务信息，并进入任务进行编辑操作。如图 5-15 所示。

任务中断恢复功能针对当前用户下，最近一次执行失败的任务进行任务执行恢复操作。若用户最近一次任务执行成功，则不会在该页面中展示相关信息并让用户执行恢复操作。该页面如图 5-16 所示。

日志管理提供当前用户下所有任务历史执行日志信息，用户可以通过该日志查看历史执行过程中所有的出错信息，或者删除部分无用日志信息记录。该模块可以帮助用户快速了解到任务运行状态，一旦出现执行错误，能够及时修改任务

流程，使得任务恢复正常运行。该页面如图 5-17 所示。

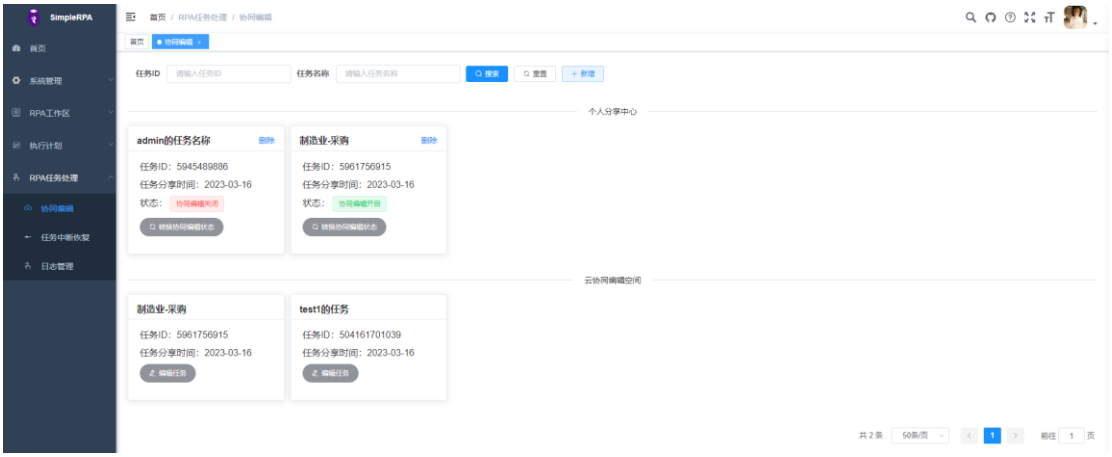


图 5-15 协同编辑页面

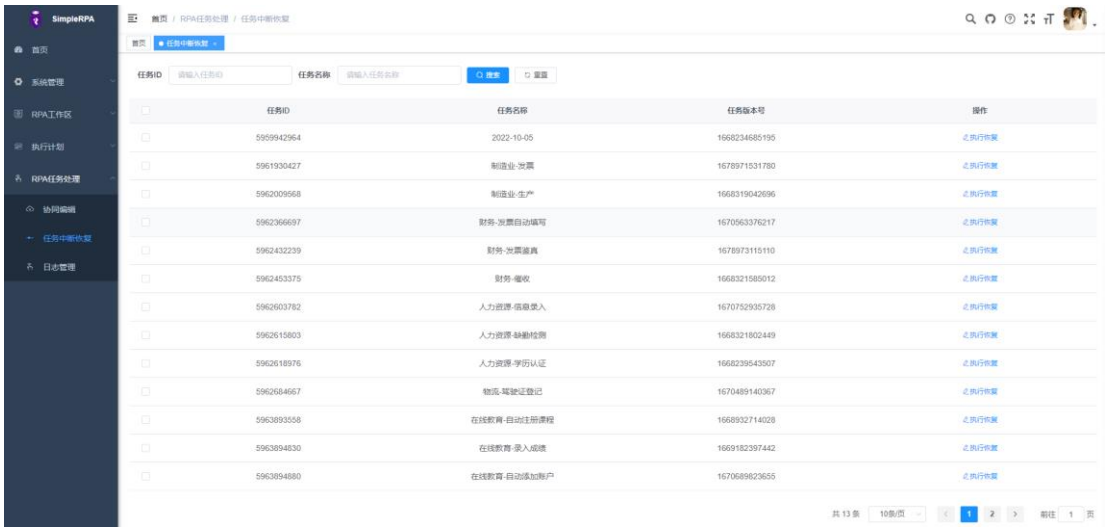


图 5-16 任务中断恢复页面

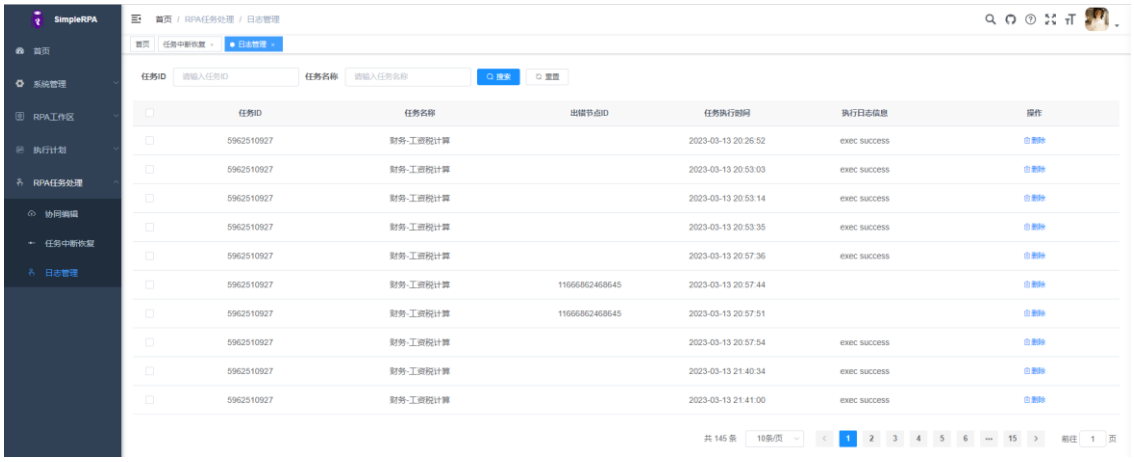


图 5-17 日志管理页面

5.2 基于 SN-PSO 的 RPA 云任务调度实验

为了验证本文提出的 SN-PSO 调度算法的有效性，并验证 SN-PSO 任务调度

算法在实际并发场景下的优化效果，本文选取了另外三种算法作为对比：CAPSO[47]，ChPSO[48]与 NSGA-III[68]。

本节将分两步对算法作对比测试。第一步通过对 Taherkhani 等人[69]使用的函数数据集选取静态函数展开静态函数测试，验证并分析算法优化策略的有效性，验证算法具备足够的寻优能力，并为后续实际并发场景测试实验提供理论支持。第二步将根据 3.3 节所提出的任务负载微基准，设计小规模并发与大规模并发任务实验，验证算法在实际并发场景下的有效性。

5.2.1 参数设置

为了保证算法对比的有效性，本文将对比算法按照作者文章中的推荐参数进行设置，为了对比的准确性，所有算法的种群大小与迭代次数分别设置为 50 和 2000，具体的调度算法参数如表 5-1 所示：

表 5-1 调度算法参数设置表

算法名称	参数名称	参数值
SN-PSO	学习因子	$c_1 = 1.49,\ c_2 = 1.49$
	惯性因子	0.729
	滑动窗口大小	30
	非支配解集阈值	30
NSGA-III	变异概率	0.1
	参考点关联个体数	2
CAPSO	学习因子	$c_{max} = 2.5,\ c_{min} = 0.5$
	惯性因子	$w_{max} = 0.9,\ w_{min} = 0.4$
ChPSO	学习因子	$c_{max} = 2.5,\ c_{min} = 1.5$
	惯性因子	$w_{max} = 0.8,\ w_{min} = 0.001$

5.2.2 静态函数测试

本文基于 Taherkhani 等人[69]使用的函数数据集对四个调度算法做静态函数测试，表 5-2 是实验平台上物理机的硬件环境表。

为了测试 SN-PSO 的算法性能，本文选取了 13 种静态函数做测试，其中包括单峰函数与多峰函数两种类型。所有选定的问题都是求函数最小值问题，通过多种类函数不同种类环境的测试验证本文算法的有效性。具体函数表如 5-3 所示。

表 5-2 硬件环境表

配置名称	参数
CPU	酷睿 i5-10500
内存	16GB DDR4
硬盘	三星 970 EVO 500G
操作系统	微软 Windows 10

表 5-3 静态函数测试表

函数名	函数公式	维度	搜索空间	最小值
Sphere	$f_1(x)=\sum_{i=1}^{D}x_i^2$	30	$[-100,100]^D$	0
Rotated hyper-ellipsoid	$f_2(x)=\sum_{i=1}^{D}\left(\sum_{j=1}^{i}x_j\right)^2$	30	$[-100,100]^D$	0
Step	$f_3(x)=\sum_{i=1}^{D}(x_i+0.5)^2$	30	$[-100,100]^D$	0
Branin	$f_4(x)=\left(x_2-\frac{5.1}{4\pi^2}x_1^2+\frac{5}{\pi}x_1-6\right)^2+10\left(1-\frac{1}{8\pi}\right)\cos(x_1)+10$	2	$-5\le x_1\le 10$ $0\le x_2\le 15$	$\frac{5}{4\pi}$
Rosenbrock	$f_5(x)=\sum_{i=1}^{D-1}[100(x_i^2-x_{i+1})^2+(x_i-1)^2]$	30	$[-5,10]^D$	0
McCormick	$f_6(x)=Sin(x_1+x_2)+(x_1-x_2)^2-1.5x_1+2.5x_2+1$	2	$-1.5\le x_1\le 4$ $-3\le x_2\le 4$	-1.9133
Beale	$f_7(x,y)=(1.5-x+xy)^2+(2.25-x+xy^2)^2+(2.625-x+xy^3)^2$	2	$[-4.5,4.5]^D$	0
Bukin N.6	$f_8(x,y)=100\sqrt{\lvert y-0.01x^2\rvert}+0.01\lvert x+10\rvert$	2	$-15\le x\le -5$ $-3\le y\le 3$	0
Schwefel	$f_9(x)=\sum_{i=1}^{D}-x_iSin\left(\sqrt{\lvert x_i\rvert}\right)$	30	$[-500,500]^D$	$-418.982D$
Rastrigin	$f_{10}(x)=\sum_{i=1}^{D}(x_i^2-10\cos(2\pi x_i)+10)$	30	$[-5.12,5.12]^D$	0
Noncontinuois Rastrigin	$f_{11}(x)=\sum_{i=1}^{D}(y_i^2-10\cos(2\pi y_i)+10)$ $y_i=\begin{cases}x_i & \lvert x_i\rvert<0.5\\ \frac{round(2x_i)}{2} & else\end{cases}$	30	$[-5.12,5.12]^D$	0
Ackley	$f_{12}(x)=-20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right)-\exp\left(\frac{1}{D}\sum_{i=1}^{D}\cos(2\pi x_i)+20\right)$	30	$[-32,32]^D$	0
Griewank	$f_{13}(x)=\frac{1}{4000}\sum_{i=1}^{D}x_i^2-\prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right)+1$	30	$[-600,600]^D$	0

根据表 5-3 的函数测试表，本文针对 SN-PSO、CAPSO、ChPSO、NSGA-III 算法对共计 13 个函数展开测试 10 次，四种算法均以寻找对应函数的最小值为目标进行搜索。根据 10 次静态函数测试结果，分别计算每个算法在各函数中测得适应度值的最小值、平均值和标准差。结果如表 5-4 所示。

表 5-4 测试函数结果表

函数	指标	CAPSO	ChPSO	NSGA-III	SN-PSO
$f_1(x)$	均值	3.56048	0.00236	35.74300	**5.209E-04**
	标准差	4.84222	0.00130	86.90800	**1.842E-04**
	最优	0.03753	8.741E-04	0.07940	**3.385E-04**
$f_2(x)$	均值	96.37603	0.21662	4627.04500	**0.07561**
	标准差	36.68424	0.06546	1398.70200	**0.03444**
	最优	74.51745	0.09599	2724.74900	**0.04075**
$f_3(x)$	均值	1.25607	0.00383	0.04504	**0.0010**
	标准差	2.28904	0.00625	0.01117	**8.535E-04**
	最优	0.04415	0.00102	0.02647	**3.033E-04**
$f_4(x)$	均值	**19.60211**	**19.60211**	19.62319	**19.60211**
	标准差	1.3958E-06	3.6174E-07	0.03795	**3.168E-07**
	最优	**19.60211**	**19.60211**	**19.60211**	**19.60211**
$f_5(x)$	均值	37.75070	32.34450	120.23050	**27.24949**
	标准差	25.61820	14.89600	77.70410	**1.81645**
	最优	**9.67060**	24.26210	28.75550	23.35712
$f_6(x)$	均值	-3.40000	-1.20000	-1.91280	**-1.10000**
	标准差	1.54059	0.40000	**0.00041**	0.30000
	最优	**-2.00000**	**-2.00000**	-1.91180	**-2.00000**
$f_7(x,y)$	均值	**0.00000**	**0.00000**	0.24830	**0.00000**
	标准差	**0.00000**	**0.00000**	0.45276	**0.00000**
	最优	**0.00000**	**0.00000**	2.4E-05	**0.00000**
$f_8(x,y)$	均值	**0.00000**	**0.00000**	1.09260	**0.00000**
	标准差	**0.00000**	**0.00000**	1.52270	**0.00000**
	最优	**0.00000**	**0.00000**	0.02530	**0.00000**
$f_9(x)$	均值	−643.58430	-6775.14210	-7894.17380	**-8907.89900**
	标准差	245.44200	794.68129	490.40910	**146.21520**
	最优	−729.10840	-7977.62460	-8560.64700	**-9016.33600**
$f_{10}(x)$	均值	78.46605	**10.02310**	88.26720	52.45215
	标准差	28.64623	**3.75885**	20.19120	13.80886
	最优	42.63644	**4.61976**	51.00850	19.98337
$f_{11}(x)$	均值	91.10759	20.33892	67.59120	**19.71306**
	标准差	14.00064	**5.88165**	24.27910	10.81781
	最优	61.02706	11.24865	24.31350	**8.26089**
$f_{12}(x)$	均值	**-1.3188E+09**	**-1.3188E+09**	-6.5856E+08	**-1.3188E+09**
	标准差	**0.00000**	**0.00000**	6.5856E+08	**0.00000**
	最优	**-1.3188E+09**	**-1.3188E+09**	-1.3181E+09	**-1.3188E+09**
$f_{13}(x)$	均值	1.19169	**2.591E-04**	105.55730	0.39952
	标准差	0.47990	**1.236E-04**	43.59240	0.45593
	最优	0.19518	**1.532E-04**	24.31350	0.00119

如表 5-4 的测试结果所示，SN-PSO 在 Taherkhani 等人所使用的静态函数测试中的最优值与均值都有较好的结果，具有一定的优势。且 SN-PSO 的 10 次测试中标准差较小，表现出了 SN-PSO 算法的搜索稳定性。但是本文提出的 SN-

PSO 算法在$f_{10}(x)$、$f_{13}(x)$两个函数上的结果相较于其他四种算法较差，这两个函数的图像特征都具有大量且密集的局部最优解，以$f_{10}(x)$ Rastrigin 函数举例，Rastrigin 函数图像如图 5-18 所示。

考虑到 Rastrigin 函数图像的特点，本文提出的 SN-PSO 算法中基于滑动窗口的变异策略中使用的是随机变异算子，在陷入局部最优解时，无法有效的变异到更优位置，在这两种函数上难以保证搜索方向为最优方向，因此算法搜索结果较差。但是本文实际调度场景下所设计的多目标优化函数并没有如此密集的局部最优点，因此 SN-PSO 算法的搜索过程在面向的 RPA 云任务调度场景下并不会被此问题干扰，依旧可以表现出较好的效果。

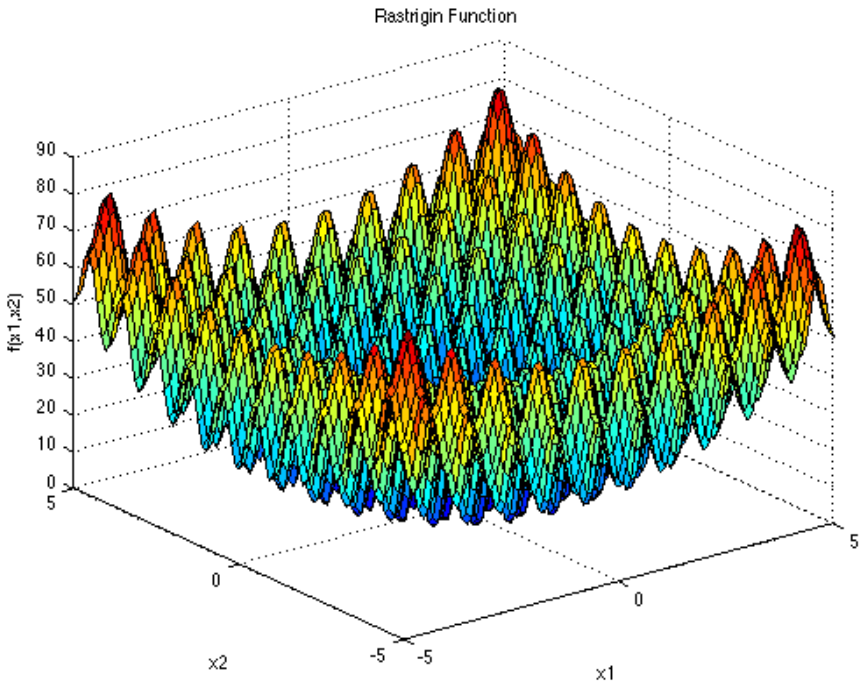


图 5-18 Rastrigin 函数图像

5.2.3 并发任务实验

为了有效模拟实际生产环境下的并发任务实验，本文搭建了具有四台机器的 kubernetes 集群用于测试。集群中每一台机器的硬件参数均相同，其参数信息如表 5-5 所示。

表 5-5 硬件环境表

配置名称	参数
CPU	Intel(R) Xeon(R) CPU E5-2630 v2
硬盘	戴尔 PERC H710 600GB
操作系统	CentOS 7.1

此外，为了模拟真实集群环境，本文构造了一种不同资源类型的集群环境，针对集群上的每台机器上的 CPU、内存进行了调整，组成了一个资源异构集群，具体的集群机器参数如表 5-6 所示。

表 5-6 集群参数设置表

机器编号	CPU 核心	内存/GB	带宽/Mbps
1	16	10	1000
2	16	8	1000
3	8	8	1000
4	4	8	1000

此外，为了验证算法在低并发状态与高并发状态下的调度具备有效性，本文针对 3.3 节中的负载测试结果，将表 3-4 中共计 23 个 RPA 任务打包成一组，每一组称之为一个并发批次。并规定，小规模场景下并发批次从 1 至 5 进行测试，大规模场景下的并发批次从 50 至 90 进行测试，并取 5 次算法调度结果中最优方案执行任务分配。具体并发任务数量对照表如表 5-7 与表 5-8 所示。

表 5-7 小规模并发任务数量对照表

并发批次	AI 任务数量	普通任务数量	总计数量
1	4	19	23
2	8	38	46
3	12	57	69
4	16	76	92
5	20	95	115

表 5-8 大规模并发任务数量对照表

并发批次	AI 任务数量	普通任务数量	总计数量
50	200	950	1150
60	240	1140	1380
70	280	1330	1610
80	320	1520	1840
90	360	1710	2070

（1） 小规模并发任务实验对比

在小规模并发实验场景下，表 5-9 所示为四个算法的目标函数中公式(4-19)计算得到总适应度值的对比结果，即任务调度总代价值，表 5-10 所示为四个算

法的目标函数中公式(4-17)计算得到的适应度值对比结果。图 5-19 所示为四个算法在实际并发任务执行时所消耗的时间。

表 5-9 总任务调度适应度值对比表

并发批次	CAPSO	ChPSO	NSGA-III	SN-PSO
1	580.18	602.78	666.52	**579.59**
2	1568.87	1627.11	1592.31	**1568.28**
3	3030.45	3097.4	3043.86	**2991.71**
4	4889.4	5000.1	4937.58	**4865.45**
5	7335.62	7410.78	7456.9	**7237.2**

表 5-10 执行代价对比表

并发批次	CAPSO	ChPSO	NSGA-III	SN-PSO
1	1145	1185.61	1113.42	**1105.49**
2	4404.55	4663.96	4464.01	**4450.16**
3	10071.48	10403.82	**9719.43**	9914.46
4	17594.78	18367.19	17795.39	**17587.71**
5	28451.42	28943.26	28481.01	**27802.41**

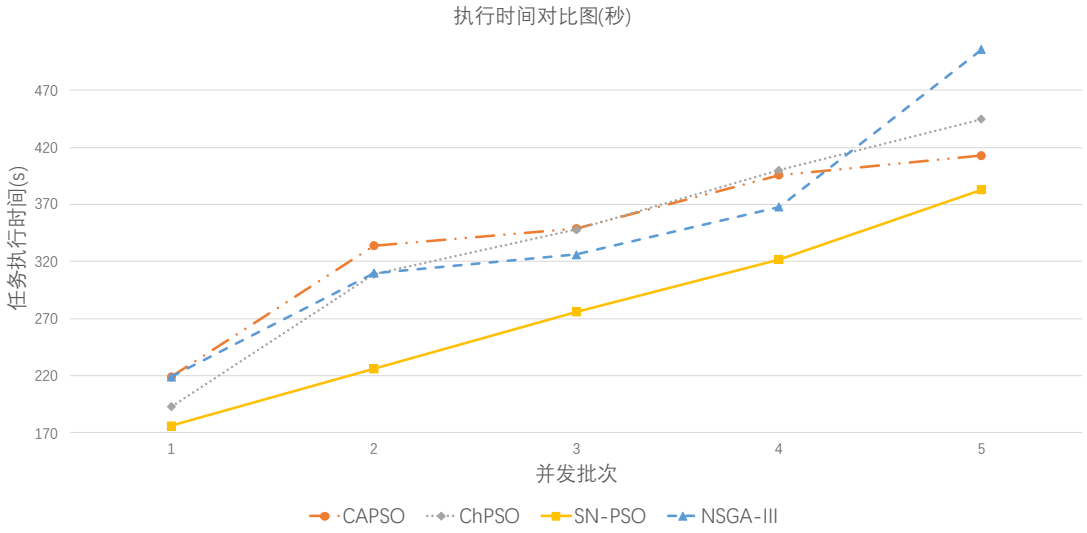


图 5-19 小规模并发任务执行时间对比图

根据表 5-9 的实验结果，本文提出的 SN-PSO 算法在总任务调度适应度值均优于其他三种对比算法。在表 5-10 的实验结果中，本文提出的 SN-PSO 算法在任务执行代价均优于其他三种对比算法。在任务执行时间方面，图 5-19 的结果表明 SN-PSO 算法整体上优于其他三种算法。图 5-19 中，当并发批次为 1 时，本文对比的四种算法调度结果差别不大，随后四种算法的执行时间与适应度值差距逐渐明显。这表明 SN-PSO 引入的优化方案中，确保了算法能够及时跳出局部最优解，并提高群体多样性，增强寻优能力。综上，在小规模并发任务实验中，

SN-PSO 算法更优。

（2） 大规模并发任务实验对比

表 5-11、表 5-12 与图 5-20 为大规模并发任务实验下，算法计算得到的任务调度适应度值、任务执行代价值与任务执行时间对比结果。

表 5-11 总任务调度适应度值对比表

并发批次	CAPSO	ChPSO	NSGA-III	SN-PSO
50	615300.82	587495.29	586685.05	**585297.56**
60	874495.76	852519.67	851316.83	**849824.87**
70	1191089.31	1157597.07	1157567.34	**1155371.21**
80	1551977.29	1506936.33	1508156.68	**1503406.08**
90	1953808.72	1905097.05	1905165.65	**1902619.88**

表 5-12 执行代价对比表

并发批次	CAPSO	ChPSO	NSGA-III	SN-PSO
50	2984481.87	2857781.92	**2844372.73**	2853566.68
60	4265707.92	4167851.15	4161571.17	**4153854.75**
70	5830988.06	5678026.69	5677601.12	**5662164.46**
80	7615340.48	7407478.15	7413709.06	**7389188.57**
90	9606113.03	9382752.75	9382609.42	**9369298.15**

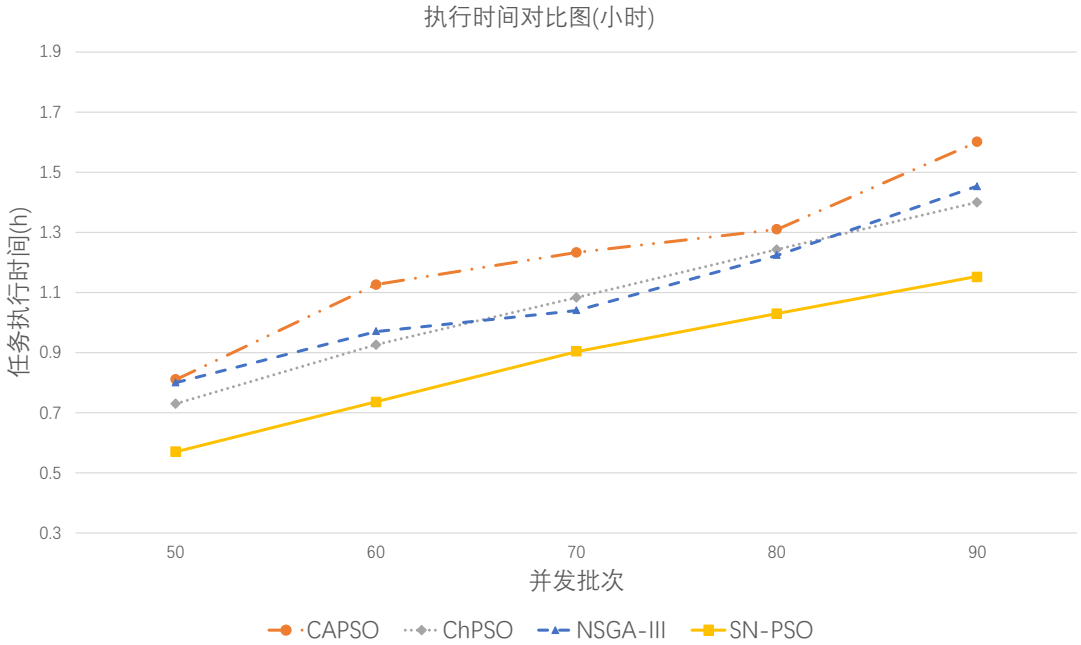


图 5-20 大规模并发任务执行时间对比图

根据表 5-11、表 5-12 与图 5-20 的实验对比结果，在大规模并发任务执行场景下，通过 SN-PSO 调度算法所求得的任务调度适应度值、任务执行代价及任务执行时间均优于其他三种算法。

综上，不论在小规模并发还是大规模并发实验场景下，引入基于滑动窗口的

变异策略可以有效地检测粒子是否陷入局部最优解，同时帮助粒子跳出局部最优状态；引入基于非支配解集优化策略能够有效的增强粒子的全局搜索能力，增强了种群的多样性，相比于其他算法能够具备更好的搜索能力，辅助算法搜索到较优位置。通过两种策略的优化效果，SN-PSO 能够搜索到一种更优的任务调度方案，以帮助系统提高任务执行效率，改善 RPA 云服务系统的服务质量。SN-PSO 算法相比较于 NSGA-III、CAPSO、ChPSO，是一种更优的调度算法。

（3） CPU 密集型实验对比

为了进一步探究不均匀负载下的算法效果，本文设置了 CPU 密集型实验对上述算法进行实验对比。为了更好的模拟真实场景下的 CPU 密集型任务实验，本节将表 3-4 中，发票真伪检验、自动填写收据、自动录入人员信息和驾驶证自动登记共 4 种涉及到 AI 功能的任务单独抽取出来，作为 CPU 密集型任务集，剩余 19 种任务作为 CPU 非密集型任务集。并且按照 80%的 CPU 密集型任务与 20%的 CPU 非密集型任务进行实验设定，具体的实验任务配置如表 5-13 所示。

表 5-13 CPU 密集型实验任务配置表

总任务数量	CPU 密集型任务数量	CPU 非密集型任务数量
20	16	4
40	32	8
60	48	12
80	64	16
100	80	20

表 5-13 中，所有的 CPU 密集型任务均从包含 4 种 AI 任务的 CPU 密集型任务集进行随机抽取，而 CPU 非密集型任务从剩余的 19 种任务种进行随机抽取。

通过上述方式，执行 CPU 密集型任务并发实验，具体的任务调度总适应度值、任务执行代价与任务执行时间分别如表 5-14、表 5-15 和图 5-21 所示。

表 5-14 总任务调度适应度值对比表

任务数量	CAPSO	ChPSO	NSGA-III	SN-PSO
20	839.86	**835.56**	859.29	836.27
40	2482.47	**2471.57**	2577.68	2472.47
60	5449.74	5451.84	5559.94	**5448.60**
80	9149.24	9160.78	9310.40	**9131.64**
100	13230.53	13205.63	13437.92	**13164.6**

表 5-15 执行代价对比表

任务数量	CAPSO	ChPSO	NSGA-III	SN-PSO
20	**1911.76**	1915.60	2044.53	1919.49
40	8237.87	8191.37	8663.92	**8187.50**
60	**20973.22**	20992.34	21578.80	20977.16
80	37428.32	37418.49	38222.82	**37382.10**
100	55993.32	55892.05	57498.96	**55509.34**

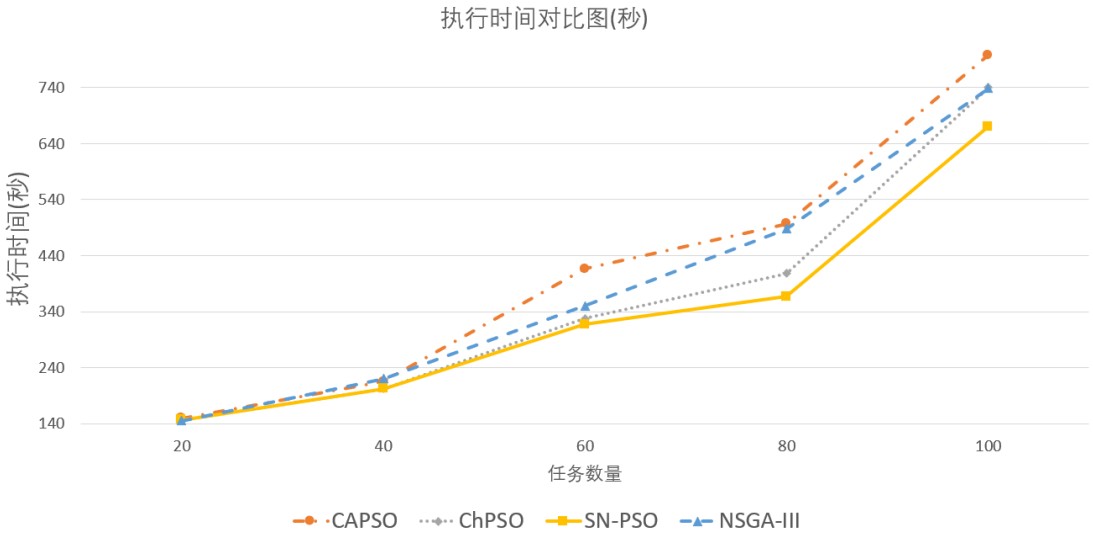


图 5-21 CPU 密集型任务执行时间对比图

根据表 5-14 与 5-15 可知，在并发任务数量为 20 与 40 时，SN-PSO 与其余三种对比算法差异不大，SN-PSO 算法得出的调度代价值与表中最优值差距非常小，可以忽略不计。当任务数量在 60 以上时，SN-PSO 算法可以获得较好的调度方案，不论在任务调度总代价还是任务执行代价指标上，均具备一定优势。在图 5-21 的执行时间对比图中，当并发任务数量为 20 和 40 时，几种算法的执行时间差异非常小，当并发任务数量达到 60 以上时，SN-PSO 算法相较于其他三种算法具备一定优势。

虽然在并发任务数量较低时，SN-PSO 得到的任务分配方案并不能在理论代价上具备明显优势，但是任务执行时间并不会因此受到影响。综上，SN-PSO 算法能够在 CPU 密集型实验场景下获得较好结果，相较于 NSGA-III、CAPSO、ChPSO，是一种更优的任务调度算法。

5.3 本章小结

本章分别介绍了 RPA 云服务系统的实现效果，并通过静态函数测试及生产环境的真实并发测试对系统中任务调度算法进行了验证。实验验证了本文提出的

SN-PSO 是有效的，且相比较于 NSGA-III、CAPSO 及 ChPSO 而言，SN-PSO 的调度结果更优。

第 6 章 总结与展望

6.1 总结

RPA 是目前新兴的研究方向与商业发展趋势，近些年，随着 RPA 热度的不断提高，越来越多的研究者开始对 RPA 展开研究，逐渐成为了一大热点。并且随着云计算与云环境的普及，RPA 与云的结合也将越来越紧密。因此针对云环境下的 RPA 云任务调度显得愈发重要，调度算法也成为云调度过程中不可或缺的一部分。本文针对 RPA 云服务系统，构建了一个基于 kubernetes 集群的云环境，并利用 selenium 等相关框架对基于网页自动化的 RPA 系统进行了编码与实现。通过 RPA 云服务系统，本文针对 5 大场景下的 23 个不同类型的任务进行了任务负载测试，得出了 RPA 云任务负载表，利用粒子群优化算法，在 RPA 云任务环境下进行建模，并使用多目标模型对 RPA 任务调度进行优化。针对粒子群中易于陷入局部最优解的问题，本文提出了一种基于滑动窗口的变异策略，用于检测粒子是否陷入局部最优解并帮助粒子跳出该状态，还提出了一种基于非支配解集的优化策略，提高了粒子群的种群多样性，增强了粒子群的搜索能力。

本文主要的工作内容：

（1） 设计了一种基于网页自动化的 RPA 云服务系统，并基于 RPA 云服务场景下设计了一个任务负载微基准方案，通过实际运行并测试，得到了多种任务场景下的资源消耗与执行时间的情况，为 RPA 云服务系统的任务调度优化提供了有效的数据支撑。

（2） 提出了一种基于滑动窗口与非支配解集的改进粒子群算法 SN-PSO。SN-PSO 通过滑动窗口检测出陷入局部最优的粒子，并调整粒子位置中贡献较低的维度。该方案避免了随机变异导致的不确定性，减少了粒子的无效搜索行为。通过非支配解集的优化策略提高了种群的多样性，增强了粒子全局寻优能力。为 RPA 云任务调度提供保障。

（3） 实现了 RPA 云服务系统并针对主要工作进行实验评估。针对 RPA 云服务系统的实现做了功能测试与效率对比实验，对 RPA 云服务系统中使用的改进粒子群算法进行了实验评估与分析，实验表明在实际使用场景下，SN-PSO 能够有效地降低并发任务总执行时间，提高系统的运行效率。

6.2 展望

本文针对 RPA 云服务系统进行了设计与实现，并针对任务调度进行了基于改进粒子群 SN-PSO 的 RPA 云任务调度优化。后期将对 RPA 云服务系统及粒子群算法中的问题继续展开相关研究，主要包含以下几点内容：

（1） 针对目前 RPA 云服务系统未实现的功能进行填充。在本文系统中，主要的核心功能是网页自动化。但是仍然有许多场景中的任务难以完成，例如针对 DOM 难解析的标签处理等，后期可以针对系统缺乏的功能进行补充。

（2） 针对变异策略进行改进。本文主要采用了基于滑动窗口的变异策略对粒子群算法进行优化，但是本文采用的变异算子是随机变异的方式，因此难以保证变异目标为一个较优的方向，因此后期将继续针对变异算子进行优化，考虑合理的变异位置，进一步增强粒子群算法的搜索能力。

（3） 系统高可用问题的研究。本文主要采用的 kubernetes 集群对 RPA 云服务系统进行搭建，但是原有方式只能保证最基础的系统高可用，一旦出现系统崩溃等特殊情况，将影响正在执行的任务。未来将考虑针对系统高可用方面进行研究，针对突发情况做容灾处理，进一步保证系统的稳定性。

致谢

在本文的完成之际，经历颇多，有着不少的感慨。曾想过致谢或许是毕业论文最简单的部分，可是当我提笔时，却忘记了该如何下笔。

2020 年是一个魔幻的时期，从最开始的疫情到硕士扩招，从选择二战到考研调剂，有过不甘心，但是也逐渐慢慢接受。那时的我常常会思考，到底何种人生选择才是正确的？2021 年在恍惚之间过去，感受到的是工程项目的紧迫，回首看到的是与实验室的同学们共坐天台，静赏晚霞的时光，遇到的是与我在一起的那个女孩。紧张的实习春招揭开了 2022 年序幕，它像一杯味道丰富的拿铁，初尝时，是费劲千辛万苦，拿下美团 offer 却因招聘寒冬停止流程的苦涩，品味时，是秋招长跑下逐渐斩获三家公司录取通知的绵长与醇厚，回味时，是年末与家人一起商讨毕业选择时的甘甜与快乐。转眼间，来到了 2023 年，毕业论文的撰写与修订带给我的不仅仅是辛酸、疲倦，也有快乐、满足。硕士的故事，还远远不止于此。

在三年的硕士生涯中，总有良师益友伴我左右。我想感谢陈梅老师对我学业上与生活上无微不至的关心与照顾，在我最困难的时候，给我最亲切的疏导；我想感谢李晖老师严谨而认真地给予我完成本篇论文所需要的知识积累，也非常感谢他能够在我误入错误的方向时予以纠正。我还想感谢实验室所有的好伙伴能够在闲时互相交流，敞开心扉，在忙时互相鼓励，共同进步。感谢万利航、孙晨航在观山湖区的相互陪伴，能够在紧张的项目开展之余增添美好的回忆；感谢李子鹏、贺鸿琨在实验室一起分享论文思路，攻坚克难；感谢叶旭芳和刘春，在最需要放松的时期，去黄果树瀑布游山玩水，感受世间的悠然；感谢闫慧明、范丹丹能够一起畅聊未来与人生，朋友间的几次交流感悟到了人生百态，令我难以忘怀；感谢余志旺在学术上的成就，让我亲眼看到了一位博士的诞生；感谢我的舍友蒋周杰带我品遍贵阳美食，一起共度生活时光，一块出行娱乐，听我诉说苦楚；感谢我本科的挚友乔治，正所谓高山流水遇知音，他成为了我硕士三年重要的精神支柱。当然，还要感谢实验室所有的师兄师姐、师弟师妹诸多陪伴与帮助。最后，我还要感谢女朋友陶佳丽一直的陪伴，让我见证了爱情的快乐与艰辛；感谢父母的支持与理解，他们铸就了我不断求学的动力。今当远离，万千感谢，不知所言。

行文至此，硕士生涯即将结束，经历了太多，感慨太多。如今，我终于明白了 2020 年的那个答案，其实哪有什么正确的人生选择，我们所有人之间的缘分就是人生最正确的选择，Follow Your Heart，无需羡慕。

此刻的分别，或许是下个十年再也难见的悲痛，我更希望我们所有人在若干年后依旧能够拥有今天的凝聚力与羁绊，不忘再见，不忘初心。

参考文献

[1] 沈港. 基于RPA的自动化办公系统的设计与实现[D].东华大学,2021.

[2] 刘功朝. 基于SaaS模式的企业办公系统设计与实现[D].山东大学,2017.

[3] Syed R, Suriadi S, Adams M, et al. Robotic process automation: contemporary themes and challenges[J]. Computers in Industry, 2020, 115: 103162.

[4] Leno V, Dumas M, La R M, et al. Automated discovery of data transformations for robotic process automation[J]. arXiv preprint arXiv:2001.01007, 2020.

[5] Moffitt K C, Rozario A M, Vasarhelyi M A. Robotic process automation for auditing[J]. Journal of emerging technologies in accounting, 2018, 15(1): 1-10.

[6] Cearley, D.; Jones, N.; Smith, D.; Burke, B.; Chandrasekaran, A.; Lu, C.; Panetta, K. Gartner Top 10 Strategic Technology Trends for 2020—Smarter with Gartner; Gartner: Stamford, CT, USA, 2019.

[7] 周栋萌. RPA的发展研究——以阿里云RPA为例[J].中国管理信息化,2021,24(14):78-79.

[8] Madakam S, Holmukhe R M, Jaiswal D K. The future digital work force: robotic process automation (RPA)[J]. JISTEM-Journal of Information Systems and Technology Management, 2019, 16.

[9] Asatiani, Aleksandre, and Esko P. "Turning robotic process automation into commercial success–Case OpusCapita." Journal of Information Technology Teaching Cases 6.2 (2016): 67-74.

[10] Wewerka J, Reichert M. Towards quantifying the effects of robotic process automation[C]//2020 IEEE 24th International Enterprise Distributed Object Computing Workshop (EDOCW). IEEE, 2020: 11-19.

[11] Ansari, Wasique A, et al. "A review on robotic process automation-the future of business organizations." 2nd International Conference on Advances in Science & Technology (ICAST). 2019.

[12] Aguirre S, Rodriguez A. Automation of a business process using robotic process automation (RPA): A case study[C]//Workshop on engineering applications. Springer, Cham, 2017: 65-71.

[13] Leno V, Deviatykh S, Polyvyanyy A, et al. Robidium: Automated Synthesis of Robotic Process Automation Scripts from UI Logs[C]//BPM (PhD/Demos). 2020: 102-106.

[14] Leno V, Augusto A, Dumas M, et al. Discovering executable routine specifications from user interaction logs[J]. arXiv preprint arXiv:2106.13446, 2021.

[15] Wewerka J, Reichert M. Robotic Process Automation--A Systematic Literature Review and Assessment Framework[J]. arXiv preprint arXiv:2012.11951, 2020.

[16] Choi D, R'bigui H, Cho C. Candidate digital tasks selection methodology for automation with robotic process automation[J]. Sustainability, 2021, 13(16): 8980.

[17] Agostinelli S, Marrella A, Mecella M. Towards intelligent robotic process automation for BPMers[J]. arXiv preprint arXiv:2001.00804, 2020.

[18] Matoussi W, Hamrouni T. A new temporal locality-based workload prediction approach for SaaS services in a cloud environment[J]. Journal of King Saud University-Computer and Information Sciences, 2022, 34(7): 3973-3987.

[19] Eisa, Mona et al. "Trends and Directions in Cloud Service Selection." 2016 IEEE Symposium on Service-Oriented System Engineering (SOSE) (2016): 423-432.

[20] Madni S H H, Abd Latiff M S, Coulibaly Y. Resource scheduling for infrastructure as a service (IaaS) in cloud computing: Challenges and opportunities[J]. Journal of Network and Computer Applications, 2016, 68: 173-200.

[21] Enríquez, José G, et al. "Robotic process automation: a scientific and industrial systematic mapping study." IEEE Access 8 (2020): 39113-39129.

[22] C. Flechsig, J. Lohmer and R. Lasch, "Realizing the full potential of robotic process automation through a combination with BPM" in Logistics Management, Cham, Switzerland:Springer, pp. 104-119, 2019.

[23] Lacity L and Willcocks M, "Robotic process automation: The next transformation lever for shared services", 2012.

[24] 张红. 一种业务流程自动化生成方法研究[D].北京邮电大学,2016.

[25] 姜明煜. 基于UiPath的机器人流程自动化框架设计及应用[D].山东科技大学,2020.

[26] 祁伟,吕湛,殷蓓. 基于RPA的供电服务虚拟座席设计[J].自动化应用,2020(12):163-165.

[27] Lacity, Mary, Leslie P. Willcocks, and Andrew C. "Robotic process automation at Telefonica O2." (2015).

[28] Vajgel B, Corrêa P L P, De Sousa T T, et al. Development of Intelligent Robotic Process

Automation: A Utility Case Study in Brazil[J]. IEEE Access, 2021, 9: 71222-71235.

[29] 张晓军. 部署机器人流程自动化的安全问题[J].网络安全和信息化,2021(10):115-116.

[30] Dong R, Huang Z, Lam I I, et al. Webrobot: Web robotic process automation using interactive programming-by-demonstration[C]//Proceedings of the 43rd ACM SIGPLAN International Conference on Programming Language Design and Implementation. 2022: 152-167.

[31] Ibrahim I M. Task scheduling algorithms in cloud computing: A review[J]. Turkish Journal of Computer and Mathematics Education (TURCOMAT), 2021, 12(4): 1041-1053.

[32] Hasan D A, Hussan B K, Zeebaree S R M, et al. The impact of test case generation methods on the software performance: A review[J]. International Journal of Science and Business, 2021, 5(6): 33-44.

[33] Gao Y, Guan H, Qi Z, et al. A multi-objective ant colony system algorithm for virtual machine placement in cloud computing[J]. Journal of computer and system sciences, 2013, 79(8): 1230-1242.

[34] Chen X, Cheng L, Liu C, et al. A WOA-based optimization approach for task scheduling in cloud computing systems[J]. IEEE Systems journal, 2020, 14(3): 3117-3128.

[35] Farid M, Latip R, Hussin M, et al. A survey on QoS requirements based on particle swarm optimization scheduling techniques for workflow scheduling in cloud computing[J]. Symmetry, 2020, 12(4): 551.

[36] Jing W, Zhao C, Miao Q, et al. QoS-DPSO: QoS-aware task scheduling for cloud computing system[J]. Journal of Network and Systems Management, 2021, 29: 1-29.

[37] Sahana S K. Ba-PSO: A Balanced PSO to solve multi-objective grid scheduling problem[J]. Applied Intelligence, 2022, 52(4): 4015-4027.

[38] 王国豪,李庆华,刘安丰. 多目标最优化云工作流调度进化遗传算法[J].计算机科学,2018,45(05):31-37+48.

[39] Shami T M, El-Saleh A A, Alswaitti M, et al. Particle Swarm Optimization: A Comprehensive Survey[J]. IEEE Access, 2022, 10: 10031-10061.

[40] Wang P, Lei Y, Agbedanu P R, et al. Makespan-driven workflow scheduling in clouds using immune-based PSO algorithm[J]. IEEE access, 2020, 8: 29281-29290.

[41] Moon Y J, Yu H C, Gil J M, et al. A slave ants based ant colony optimization algorithm for

task scheduling in cloud computing environments[J]. Human-centric Computing and Information Sciences, 2017, 7(1): 1-10.

[42] Nabi S, Ahmad M, Ibrahim M, et al. AdPSO: adaptive PSO-based task scheduling approach for cloud computing[J]. Sensors, 2022, 22(3): 920.

[43] Kchaou H, Kechaou Z, Alimi A M. A PSO task scheduling and IT2FCM fuzzy data placement strategy for scientific cloud workflows[J]. Journal of Computational Science, 2022, 64: 101840.

[44] Ebrahimian H, Barmayoon S, Mohammadi M, et al. The price prediction for the energy market based on a new method[J]. Economic research-Ekonomska istraživanja, 2018, 31(1): 313-337.

[45] Kennedy J, Eberhart R. Particle swarm optimization[C]//Proceedings of ICNN'95-international conference on neural networks. IEEE, 1995, 4: 1942-1948.

[46] Rodriguez M A, Buyya R. Deadline based resource provisioningand scheduling algorithm for scientific workflows on clouds[J]. IEEE transactions on cloud computing, 2014, 2(2): 222-235.

[47] Duan Y, Chen N, Chang L, et al. CAPSO: Chaos Adaptive Particle Swarm Optimization Algorithm[J]. IEEE Access, 2022, 10: 29393-29405.

[48] Cheng M, Han Y. Application of a modified CES production function model based on improved PSO algorithm[J]. Applied Mathematics and Computation, 2020, 387: 125178.

[49] 张晓丽. 自适应CPSO算法在云计算任务调度中的应用[J].计算机技术与发展,2016,26(08):161-165.

[50] 马钰,杨迪,王鹏. 基于改进粒子群算法的云计算调度策略[J].长春理工大学学报(自然科学版),2022,45(05):80-86.

[51] Manasrah A M, Ba Ali H. Workflow scheduling using hybrid GA-PSO algorithm in cloud computing[J]. Wireless Communications and Mobile Computing, 2018, 2018: 1-16.

[52] 马学森,谈杰,陈树友,储昭坤,石雷. 云计算多目标任务调度的优化粒子群算法研究[J].电子测量与仪器学报,2020,34(08):133-143.

[53] Dordaie N, Navimipour N J. A hybrid particle swarm optimization and hill climbing algorithm for task scheduling in the cloud environments[J]. ICT Express, 2018, 4(4): 199-202.

[54] Laskar N M, Guha K, Chatterjee I, et al. HWPSO: A new hybrid whale-particle swarm optimization algorithm and its application in electronic design optimization problems[J]. Applied Intelligence, 2019, 49: 265-291.

[55] Suyono H, Subekti E, Purnomo H, et al. Economic Dispatch of 500 kV Java-Bali Power System using Hybrid Particle Swarm-Ant Colony Optimization Method[C]//2020 12th International Conference on Electrical Engineering (ICEENG). IEEE, 2020: 5-10.

[56] Alkayal E S, Jennings N R, Abulkhair M F. Survey of task scheduling in cloud computing based on particle swarm optimization[C]//2017 International Conference on Electrical and Computing Technologies and Applications (ICECTA). IEEE, 2017: 1-6.

[57] Verma, Amandeep, and Sakshi K. "A hybrid multi-objective particle swarm optimization for scientific workflow scheduling." Parallel Computing 62 (2017): 1-19.

[58] Saeedi S, Khorsand R, Bidgoli S G, et al. Improved many-objective particle swarm optimization algorithm for scientific workflow scheduling in cloud computing[J]. Computers & Industrial Engineering, 2020, 147: 106649.

[59] Kumar M, Sharma S C. PSO-based novel resource scheduling technique to improve QoS parameters in cloud computing[J]. Neural Computing and Applications, 2020, 32: 12103-12126.

[60] Agostinelli S, Lupia M, Marrella A, et al. Reactive synthesis of software robots in RPA from user interface logs[J]. Computers in Industry, 2022, 142: 103721.

[61] Ruiz R C, Ramírez A J, Cuaresma M J E, et al. Hybridizing humans and robots: An RPA horizon envisaged from the trenches[J]. Computers in Industry, 2022, 138: 103615.

[62] Schwarzkopf M. Operating system support for warehouse-scale computing[D]. University of Cambridge, 2018.

[63] Houssein E H, Gad A G, Wazery Y M, et al. Task scheduling in cloud computing based on meta-heuristics: review, taxonomy, open challenges, and future trends[J]. Swarm and Evolutionary Computation, 2021, 62: 100841.

[64] Karaboga D. An idea based on honey bee swarm for numerical optimization[R]. Technical report-tr06, Erciyes university, engineering faculty, computer engineering department, 2005.

[65] Shi Y, Eberhart R. A modified particle swarm optimizer[C]//1998 IEEE international conference on evolutionary computation proceedings. IEEE world congress on computational intelligence (Cat. No. 98TH8360). IEEE, 1998: 69-73.

[66] Clerc M, Kennedy J. The particle swarm-explosion, stability, and convergence in a

multidimensional complex space[J]. IEEE transactions on Evolutionary Computation, 2002, 6(1): 58-73.

[67] 杨戈,赵鑫,黄静. 面向云计算的任务调度算法综述[J].计算机系统应用,2020,29(03):11-19.

[68] Deb K, Jain H. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: solving problems with box constraints[J]. IEEE transactions on evolutionary computation, 2013, 18(4): 577-601.

[69] Taherkhani, Mojtaba, and Reza S. "A novel stability-based adaptive inertia weight for particle swarm optimization." Applied Soft Computing 38 (2016): 281-295.

附录Ⅰ 硕士期间研究成果

一、参与的科研项目：

1. 参与实验室横向课题两项

二、研究生期间获奖情况：

1. 第十二届蓝桥杯全国软件和信息技术专业人才大赛研究生组二等奖
2. 2020-2021 年度贵州大学硕士三等奖学金
3. 2021-2022 年度第三届全国大学生算法设计与编程挑战赛(秋季赛) 银奖
4. 第十九届中国研究生数学建模竞赛 优秀奖

附录 II 系统用例描述

根据以上功能性需求约束，RPA 云服务系统用例图如图 7-1 所示。本小节将具体介绍相关主要用例说明。

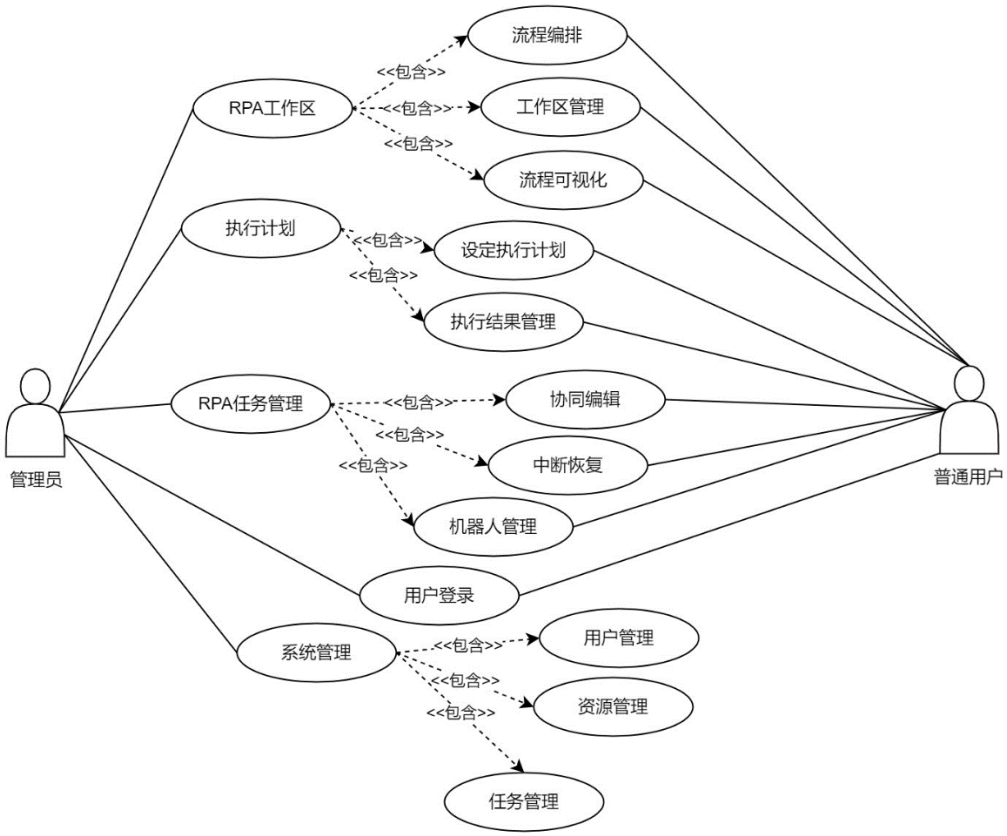


图 7-1 系统用例图

RPA 云服务系统中将用户划分为管理员与普通用户两类角色。所有功能需要登录后才可以被使用，管理员与普通用户均拥有任务编排、任务执行、任务状态监控等功能。同时管理员拥有修改用户权限与用户管理的功能。

用户登录的用例描述说明表如表 7-1 所示：

表 7-1 用户登录用例说明表

描述项	说明
用例名称	用户登录
用例描述	通过不同用户权限提供不同种类的功能
活动者	管理员与普通用户
优先级	高

描述项	说明
前置条件	进入系统登录页面
后置条件	登录验证成功后，进入 RPA 云服务系统首页
基本操作流	1. 用户进入系统登录界面 2. 在登录界面中输入用户名与密码 3. 登录验证成功，进入 RPA 云服务系统并显示系统首页
异常事件流	1. 登录验证的用户名与密码错误，登录失败

考虑到 RPA 云服务系统的后期运行维护需求，管理员需要拥有对用户进行管理并修改用户权限的功能，因此用户管理模块允许管理员对所有已经注册的用户进行查看、添加、删除、更新等操作，此功能普通用户无法使用。基于以上，用户管理相关用例描述说明表如表 7-2 所示。

表 7-2 用户管理用例说明表

描述项	说明
用例名称	用户管理
用例描述	拥有对账户增加、修改、删除、更新的功能
活动者	管理员
优先级	低
前置条件	成功登录至 RPA 云服务系统中
后置条件	对用户信息完成实际修改
基本操作流	1. 管理员进入系统首页 2. 点击用户管理选项卡 3. 点击添加按钮，并输入相关用户信息，点击确定 4. 选择已有用户并点击删除按钮，点击确定 5. 选择已有用户并点击修改按钮，输入需要修改的信息，点击确定 6. 选择已有用户并修改权限用户相关权限信息 7. 完成用户管理操作，点击关闭当前页面
异常事件流	1. 添加的用户已经存在于数据库中，弹出提示信息 2. 更新的用户邮箱或者用户名与其他账户重复，弹出提示框 3. 用户的权限没有选中信息，弹出提示框

RPA 云服务系统中，管理员还拥有资源管理与任务管理两个部分，管理员可以针对当前系统状态，关闭或者开启相关系统服务，以调整系统占有的资源，同时也可以针对系统中所有的任务进行删除与禁用两种操作，有效控制用户任务。基于以上，资源管理用例说明如表 7-3 所示，任务管理用例说明表如表 7-4 所示。

表 7-3 资源管理用例说明表

描述项	说明
用例名称	资源管理
用例描述	对系统服务资源执行启用与暂停命令
活动者	管理员
优先级	低
前置条件	成功登录至 RPA 云服务系统中
后置条件	对服务资源进行调整
基本操作流	1. 管理员进入系统首页 2. 点击资源管理选项卡 3. 对目前已暂停使用的服务资源点击启用按钮 4. 对目前正在运行的服务资源点击暂停按钮 5. 完成资源管理操作，点击关闭当前页面
异常事件流	1. 启用正在运行的服务资源或者暂停已经停止服务的资源，执行操作失败

表 7-4 任务管理用例说明表

描述项	说明
用例名称	任务管理
用例描述	对系统中所有的任务执行删除或者禁用操作
活动者	管理员
优先级	低
前置条件	成功登录至 RPA 云服务系统中
后置条件	改变任务状态
基本操作流	1. 管理员进入系统首页 2. 点击任务管理选项卡 3. 对已有任务点击删除或者修改按钮 4. 完成任务管理操作，点击关闭当前页面
异常事件流	1. 对已经禁用的任务执行禁用指令，执行失败

RPA 云服务系统提供对 RPA 任务流程编排的功能，普通用户及管理员均可在系统上按照自身需求编排任务工作流，并将相关任务工作流程保存至系统中。

系统将根据用户设定和操作方式决定是否立即执行当前任务工作流或定时执行当前任务工作流。任务流程编排功能用例说明表如表 7-5 所示。

表 7-5 任务流程编排功能用例说明表

描述项	说明
用例名称	任务流程编排
用例描述	用户根据自身需求，使用任务流程编排功能对任务进行设计
活动者	管理员与普通用户
优先级	高
前置条件	1. 成功登录至系统首页 2. 系统中已经存在任务流程或者新建一个任务流程
后置条件	将设计好的任务流程保存至系统中
基本操作流	1. 用户登录成功，进入系统首页 2. 在导航栏中选择 RPA 流程管理选项卡，进入 RPA 流程管理界面 3. 点击新增按钮或选择已有任务流程点击编辑按钮，进入任务流程编排页面 4. 在页面左侧的任务功能组件选择区拖拽至任务流程画布上，并按照用户自身需求对任务组件连线形成任务执行图或在页面右侧对组件参数进行编辑 5. 当用户编辑好后，点击保存按钮将当前任务工作流保存至系统中，或者点击运行按钮，执行当前任务 6. 完成任务流程编排工作后，关闭当前页面回到首页
异常事件流	1. 用户对组件连线出现自环时，连接失败 2. 用户拖拽组件未至指定画布时，组件放置失败 3. 用户连线选择的终点位置上没有组件时，连线失败

RPA 云服务系统存储了用户所有编辑完成的任务工作流程信息，用户可以选择删除任务流程、新增任务流程或编辑现有任务流程。工作区管理的相关功能

用例说明表如表 7-6 所示。

表 7-6 工作区管理用例说明表

描述项	说明
用例名称	工作区管理
用例描述	对具体 RPA 任务流程做增加、修改、删除操作
活动者	管理员与普通用户
优先级	高
前置条件	成功登录至 RPA 云服务系统中
后置条件	对实际 RPA 任务流程执行修改
基本操作流	1. 用户进入系统首页 2. 点击工作区管理选项卡 3. 点击添加按钮，并输入任务名称信息，点击确定 4. 选择已有 RPA 任务并点击删除按钮，点击确定 5. 选择已有 RPA 任务进行编辑，在跳转后的页面中修改任务流程并保存 6. 完成工作区管理操作，点击关闭当前页面
异常事件流	1. 未选择任务并点击删除按钮，执行失败

在 RPA 工作区中，系统还提供了可视化功能，用户可以利用可视化页面查看当前任务的执行状态、历史执行情况等信息。可视化用例说明表如表 7-7 所示。

表 7-7 流程可视化用例说明表

描述项	说明
用例名称	流程可视化
用例描述	对具体 RPA 任务流程通过图标的方式展现任务执行状态、历史执行信息等
活动者	管理员与普通用户
优先级	高
前置条件	成功登录至 RPA 云服务系统中
后置条件	显示 RPA 任务可视化结果
基本操作流	1. 用户进入系统首页 2. 点击工作区管理选项卡 3. 选择已有的 RPA 任务，并点击流程可视化

描述项	说明
	按钮 4. 查看完毕后，点击关闭当前可视化界面

RPA 云服务系统提供了定时任务执行的功能，可以通过设定任务执行计划，实现无人值守的任务运行模式。执行计划用例说明表如表 7-8 所示。

表 7-8 执行计划用例说明表

描述项	说明
用例名称	执行计划
用例描述	设定 RPA 任务的执行模式
活动者	管理员与普通用户
优先级	高
前置条件	成功登录至 RPA 云服务系统中
后置条件	设定执行计划，并等待触发任务的执行
基本操作流	1. 用户进入系统首页 2. 点击执行计划选项卡 3. 选择已有的 RPA 任务，根据需求，设定任务的定时执行计划 4. 选择已有 RPA 任务并查看执行结果的情况与历史执行数据 5. 完成操作，点击关闭当前页面

RPA 云服务系统提供任务管理的功能，可以将任务调整至协同编辑状态，与其他用户共同设计任务流程，同时可以针对因异常或错误导致的中断任务做执行恢复，也可以针对正在运行的任务执行中断执行等操作。RPA 任务管理相关的用例说明表如表 7-9 所示。

表 7-9 RPA 任务管理用例说明表

描述项	说明
用例名称	RPA 任务管理
用例描述	对已有 RPA 任务启动协同编辑、中断任务恢复执行与暂停任务的功能
活动者	管理员与普通用户
优先级	高
前置条件	成功登录至 RPA 云服务系统中
后置条件	修改 RPA 任务状态

描述项	说明
基本操作流	1. 用户进入系统首页 2. 点击 RPA 任务管理选项卡 3. 选择已有的 RPA 任务并调整 RPA 任务为协同编辑状态，并发布到协同编辑板块 4. 选择最近一次执行出现中断的 RPA 任务并点击中断恢复按钮，恢复 RPA 任务的运行 5. 选择正在运行的 RPA 任务，点击暂停任务执行的按钮，停止此次执行 6. 完成相关操作，点击关闭当前页面
异常事件流	1. 目标任务没有出现过执行中断的情况，中断恢复执行失败 2. 对未运行的 RPA 任务执行暂停操作，暂停任务失败

图版

表版

原 创 性 声 明

本人郑重声明：所呈交的学位论文，是本人在导师的指导下，独立进行研究所取得的成果。除文中已经注明引用的内容外，本论文不包含任何其他个人或集体已经发表或撰写过的科研成果。对本文的研究在做出重要贡献的个人和集体，均已在文中以明确方式标明。本人完全意识到本声明的法律责任由本人承担。

论文作者签名：_____ 日 期：20 年 月

关于学位论文使用授权的声明

本人完全了解贵州大学有关保留、使用学位论文的规定，同意学校保留或向国家有关部门或机构送交论文的复印件和电子版，允许论文被查阅和借阅；本人授权贵州大学可以将本学位论文的全部或部分内容编入有关数据库进行检索，可以采用影印、缩印或其他复制手段保存论文和汇编本学位论文。

(保密论文在解密后应遵守此规定)

论文作者签名：_____导师签名：_____日 期：20 年 月