大学生论文检测系统

文本复制检测报告单(全文标明引文)

№:ADBD2023R_20230601163944472809101337 检测时间：2023-06-01 16:39:44

篇名： 多算子协同评测的资源调度平台模拟评估系统

作者： 魏宇 (11912920;计算机科学与工程系;计算机科学与技术)

指导教师： 宋轩

检测机构： 南方科技大学

提交论文IP： 110.***.***.***

文件名： sustechthesis_1_3_4.pdf

检测系统： 大学生论文检测系统

检测类型： 大学生论文

检测范围： 中国学术期刊网络出版总库
中国博士学位论文全文数据库/中国优秀硕士学位论文全文数据库
中国重要会议论文全文数据库
中国重要报纸全文数据库
中国专利全文数据库
图书资源
优先出版文献库
大学生论文联合比对库
互联网资源(包含贴吧等论坛资源)
英文数据库(涵盖期刊、博硕、会议的英文数据以及德国Springer、英国Taylor&Francis 期刊数据库等)
港澳台学术文献库
互联网文档资源
源代码库
CNKI大成编客-原创作品库
机构自建比对库

时间范围： 1900-01-01至2023-06-01

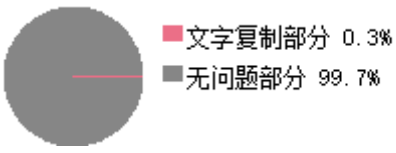
检测结果

去除本人文献复制比： 0.3% **跨语言检测结果：** -

去除引用文献复制比： **0.3%** **总文字复制比：** **0.3%**

单篇最大文字复制比： **0.3%**

重复字数：	[38]	总段落数：	[2]
总字数：	[14918]	疑似段落数：	[1]
单篇最大重复字数：	[38]	前部重合字数：	[0]
疑似段落最大重合字数：	[38]	后部重合字数：	[38]
疑似段落最小重合字数：	[38]		



指标： 疑似剽窃观点 疑似剽窃文字表述 疑似整体剽窃 过度引用

相似表格：0 相似公式：没有数据 疑似文字的图片：0

0%	(0)	0% (0)	多算子协同评测的资源调度平台模拟评估系统_第1部分 （总10643字）
0.9%	(38)	0.9% (38)	多算子协同评测的资源调度平台模拟评估系统_第2部分 （总4275字）

（注释： 无问题部分 文字复制部分 引用部分）

指导教师审查结果

指导教师：宋轩
审阅结果：
审阅意见：指导老师未填写审阅意见

1. 多算子协同评测的资源调度平台模拟评估系统_第1部分 总字数：**10643**

相似文献列表

去除本人文献复制比：0%(0) 文字复制比：0%(0) 疑似剽窃观点（0）

原文内容

分类号编号

U D C 密级

本科生毕业设计（论文）

题目： 多算子协同评测的资源调度平台模拟评估系统

姓名： 魏宇

学号： 11912920

系别： 计算机科学与工程系

专业： 计算机科学与技术

指导教师： 宋轩副教授

2023 年 5 月 8 日

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（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。

2. 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。

3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭
他人研究成果和伪造相关数据等行为。

4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本
人承担相应的法律责任。

作者签名:

年月日多算子协同评测的资源调度平台模拟评估系统魏宇

（计算机系指导教师：宋轩）

[摘要]：随着物联网技术的普及和发展，视觉人工智能系统变得越来越
重要。其适用于各种应用，特别是在城市治理领域有着不可或缺的功能：

例如行人行为识别，人流统计等等。但是人工智能系统被分配的任务的复杂性和强实时性以及容器资源的有限性使得合理的资源调度成为我们必须面对的问题。研究人员探索了许多优化算法，例如遗传算法、演化算法，以合理地将人工智能系统任务分配给每一时刻以及每个容器，得出性能优秀的资源调度算法。然而，在开发此类资源调度算法中存在的许多问题需要开发相关算法测试评估平台解决。我们的目标就是创建合

理的资源调度算法测试评估平台。该平台将通过考量算法对多种资源如CPU,GPU 的占用优化实现多算子协同评测，能够测试相关资源调度算法

的准确性以及优化能力。在实现过程中，我们以 SpringBoot 为后端框架，Vue.js 为前端框架，实现了一个基础的算法测试平台。在此基础上，我们正确分析了所需测试算法的特点，并且根据算法工作者需求合理地对算法测试平台性能进行了优化，并为用户提供了可供测试的测试集以及算

法优化结果可视化的功能，进一步方便算法工作者们开发新的资源调度
算法。

[关键词]：性能评估；资源调度；算法测试平台

[ABSTRACT]: It is obvious nowadays that visual AI systems are becoming more and more important for a variety of applications, such as behavior recognition, traffic calculation and so on, particularly in the sphere of city governance. Thus it has become a must for algorithm developers to develop an effective visual AI system for city governance. Then task scheduling is the most important part of this analysis system, given the complex characteristics of the inspection tasks and limited resources. An efficient task scheduling algorithm is critical for overcoming these challenges and optimizing system performance. Researchers explored many optimization and machine learning techniques, such as genetic algorithms, neural networks, and reinforcement learning, to create a customized task scheduling algorithm. And our goal is to create the algorithm testing platform.This platform will be able to test the accuracy the optimal result for every algorithm used to solve given problems. In the process of realizing it, we use SpringBoot as our back-end framework and Vue.js as front-end framework to create a basic algorithm testing platform. In addition, we analyzed the characteristics of these algorithms to be tested and improve the performance of the algorithm testing platform based on algorithm developer’s requirements. Furthermore, we provide the users with different test cases and the function of visualize optimization result of algorithms to make algorithm developer’s work easier.

[Key words]: performance evaluation, task scheduling, algorithm testing platform

目录

IV

1. 前言

1.1 算法需求背景

随着物联网技术的普及和发展，视觉人工智能系统变得越来越重要。其适用于各种应用，特别是在城市治理领域有着不可或缺的功能：例如行人行为识别，人流统计等等。

a) 交通流监控 b) 行为检测

c) 监控摄像头图 1 人工智能系统执行任务但是人工智能系统被分配的任务具有的复杂特征，如周期性，相关性和多目标限制以及计算资源的有限性使得合理的资源调度成为我们必须面对的问题。

1.2 算法问题建模

算法工作者们为了解决这一资源调度问题，对其进行了建模：

a) 任务定义

一个任务指一个周期性运行一个特定算法的进程, 即 x = [cx, f

x]。其中 x 表示

一个任务，cx表示提供此任务的摄像头，f

X

表示此任务需要运行的算法。并且对于每一个任务 x，它都拥有三个属性： 1

• t = t(cx, f

x) 表示此任务运行一次所需要的时间，其由提供任务的摄像头和所要运行的算法唯一确定。

• T = T(cx, f

x) 表示算法的切换周期，其亦由提供任务的摄像头和所要运行的算法唯一确定。

• 负载：即每个任务所需要的资源，共有四个属性：CPU 数，即 CPU 算力

uc

pu(x), GPU 数，即 GPU 算力 u

gpu(x), 内存占用 M

cpu(x) 以及显存占用

Mg

pu(x)。这些属性由任务的功能算子集合以及功能模型集合决定，而任务的功能算子集合以及功能模型集合由任务本身运行的算法决定。

b) 目标计算我们评估一个资源调度算法是否合理主要计算下面两个目标函数：

• 总调度所需时间：表示该调度算法运行所有任务需要的总时间，表示为

tr

unning 。

• 资源占用率：表示完成给定任务集的调度所需的容器数量和资源占用率。

根据调度算法的结果，可以确定每个容器上每个时刻运行的计算任务，从

而确定该容器上的资源负载。定义了指定维度的资源，就可以计算出该维度资源的平均使用率。由于我们在此处以 CPU，GPU，显存，CPU 内存四大属性作为我们的资源占用率计算标准，而这四大属性由任务的功能算子集合以及功能模型决定，所以我们最终的评估标准为多算子协同评测，将更好地反馈算法的各方面性能。

1.3 测试平台开发背景

在了解资源调度问题模型后，我们参考了相关类型的问题以及解决这些问题的算法，并通过分析论证以及与算法工作者们合作了解到：解决该问题的算法一般为启发式算法和演化算法，深度学习由于算力需求过大很难解决该类问题。同时算法工作者们给出了算法开发中的一些困难，希望我们去帮助解决：

• 目前没有用于测试资源调度算法性能的数据集，算法工作者们常常手工构造数据集，但仍然面对构造数据集时间长，规模小，无准确性保证的问题。 2

• 算法测试需求的算力很大，算法工作者在本地常常难以运行代码，希望能够有统一的测试平台具备可观算力帮助运行算法。

• 算法测试环境没有保障，算法工作者们面对需要长时间运行的算法很难长时间

维持算法测试环境，保证算法测试结果准确。

• 算法优化结果没有一个很直观的显示，算法工作者们常常没有头绪进一步优化。

面对这些问题，我们决定创建合理的资源调度算法测试评估平台。该平台将通过考量算法对多种资源如 CPU, GPU 的占用优化实现多算子协同评测，能够测试相关资源调度算法的准确性以及优化能力。同时我们针对算法工作者们工作中遇到的困难对算法测试平台进一步优化并提供相关功能：我们将构造合理并具有不同规模的数据集放置于平台供算法工作者们测试算法，同时将平台部署至服务器以为用户提供

高算力且稳定的开发环境，最后我们将提供合理的算法优化可视化结果，以帮助算法

工作获取优化思路。

1.4 测试平台设计框架以及优化

在实现过程中，我们以 SpringBoot 为后端框架，Vue.js 为前端框架，实现了一个基础的算法测试平台。在此基础上，我们正确分析了所需测试算法以及用户的特点，使用相关组件提升平台服务性能，最终提高算法开发者们的用户体验：

• 高并发能力优化：使用 RabbitMQ 组件，保证高并发性能。

• 数据库响应优化：使用 Redis 组件作为数据库缓存，优化数据库查询效率。

• 安全运行环境优化：手工搭建沙盒提供安全运行环境。

1.5 预期结果

我们最终的目标就是实现一个多算子协同评测的资源调度算法测试平台，该平台不仅能够提供大量合理数据集，还能够正确评测并且通过可视化方式反馈用户以

算法优化结果。除此以外，该平台还应当部署在高算力服务器上，并且具有强高并发

能力，快数据库响应速度，安全的代码测评环境，以提供算法工作者们更加优质的服务。 3

2. 相关工作

为了达成合理的资源调度，从而保证人工智能系统的负载利用率能够达到最优，许多资源调度的优化算法以及算法评估方法都被开发用来解决这样的资源调度问题。

在这里我们对一些典型的优化算法进行简要的介绍, 通过对这些算法有进一步了解，我们将对我们进行测试的算法也有更深一步的了解，从而针对需要测试的算法对算法测试平台进一步优化。

2.1 CVRP（Capacitated Vehicle Routing Problem）,TSP（Traveling Salesman

Problem）类组合优化问题由于此问题的提出相对较新，现有研究中并没有针对此类问题的单独求解。但是该问题如果只考虑单个时刻的分配，与 CVRP，TSP 类组合优化问题较为类似。CVRP

此类问题大多的解决方案多为：启发式算法，演化算法，或者两者的相结合。其中例子可见 GA[1],TS[2]以及文化基因算法[3]。强化式学习亦能够成为该问题的解决方案之一。例如 Google Brain 开发的通用的基于循环神经网络 + 策略梯度的强化学习

模型[4]。其中亦有很多的优化策略以及相关工作例如 2-opt（2-optimization）[5], Sym-

NCO（Symmetric Neural Combinatorial Optimization）[6],POMO(Policy Optimization with

Multiple Optima)[

7]等等。但是考虑到本问题如果加入时间维度，强化学习由于需要相

当多的算力去训练其深度神经网络，在问题维度拓宽的情况下不适合去使用。因此解决此类问题的算法多为非深度学习算法，即上文提及到的启发式算法和演化算法。

2.2 装箱问题

装箱问题[8]是另一个与此问题相似的场景，其中启发式算法[9]是解决此类问题的

非常好的方案之一，例如。强化学习亦能够在解决该问题中发挥很好的作用，例如AC（Access Control）架构[10]，注意力机制与强化学习结合的方法[11]等等。但是同样的，我们的问题在装箱问题基础之上也拓宽了一个时间维度，因此深度学习方法对于我们的问题仍然不够适用。

2.3 总结

通过对于这些解决模型问题的算法的了解，我们从而能够对资源调度问题有一

个更深层次的了解，即解决该资源调度问题大多使用非深度学习算法，例如启发式算 4

法和演化算法。同时，这些非深度学习算法需要大量的算力支持每次的演化或者迭代。因此，我们需要相应地搭建针对于这些算法的算法测试平台，即能够提供大量算力，保证程序长时间运行稳定，提供程序多次提交并且程序提交不丢失的测试平台。

3. 方法与设计

在前言中我们对我们测试的算法所要解决的问题进行了建模：即城市计算资源的调度。而在相关工作中我们了解到，我们测试的这些资源调度算法多为非深度学习算法，一般来说为演化算法和启发式算法。因此，为了方便算法开发工作者们对这些算法进行研究，我们搭建了专门用于测试该类算法的测试平台。而经过分析可知，我们需要搭建的算法测试平台有以下几个特点:

• 算力充足：以保证通常情况下需要大量算力的算法能够正常运行。

• 测试集合理且充分：为了保证算法的正确性以及合理度量算法的效率，我们需要搭建合理的相关测试集来对算法的这些特质进行考量。

• 高负载并发能力强：由于我们所要测试的算法大多需要数分钟以上的时间去运

行，期间收到的许多算法测试请求都要能够完整保留并正确响应结果。

• 数据库响应速度快：在高并发情况下数据库响应速度是一个至关重要的因素，我们需要提高数据库的相关性能以提高高并发能力。

• 算法运行环境安全稳定：为了帮助算法开发工作者们有一个稳定的开发环境以提高开发效率，我们为算法开发者们提供的平台要足够稳定，同时为了防止恶意代码入侵，我们对测试平台的安全性也要进行相应的维护。

考虑到如上几个特点，我们设计的算法测试平台主要结构如图 2 所示。其中前端主要

采用 Vue.js 实现良好的用户交互体验，能够使用户实时地得到自己算法结果的测试

反馈，并在此基础上加以改进。后端主要采用 SpringBoot 作为我们整体的项目框架，同时加入 Redis, Rabbit MQ，Mybatis 等等组件，为我们提供良好的系统支持，保证系统运行快捷便利的同时具有相当的安全性。同时，针对以上我们提出的算法测试平台 5

表 1 数据集分布

编号容器数量巡检任务数周期任务数

A 1 4 6

B 1 9 22

C 1 15 38

D 2 75 10890

E 17 750 110833

F 2 81 225

G 4 290 290

H 6 162 450

所应具有的特点，我们做出了相应的系统优化，我们将在下文对这些系统优化结合系统核心流程进行阐述。

3.1 测试集设计

由于此次模型情景较为新颖，目前开源的数据集大多不适合作为我们此次问题的测试集。因此我们考虑自己设计测试集来帮助测试算法。我们本次测试集的设计主要参考现实生活中的任务分布以及资源分布，而在真实环境下任务的周期，运行时间，占用的各项资源以及容器的资源大小都可以近似为随机分布。于是我们在与阿里巴巴 City Brain 部门合作中获得了一些可用于参考的常见任务周期，运行时间，占用资源，容器资源等参数，并使得这些数据在一定范围内随机波动。同时考虑到测试集对于算法的优劣的体现程度，以及测试集对算法测试平台造成的负载压力，我们设计了具有不同大小以及不同特点的测试集以对算法的综合能力进行测评。

如表 1 所示，其中 A,B,C 为小规模的测试集，目的在于对算法正确性的验证。而

D,E 为通过参考阿里巴巴提供的相关参数以及运行日志之后手动生成的大规模测试

集，目的在于对算法的优化能力进行进一步的考量。然而，我们合作的算法开发工

作者在测试之后提出了在容器以及任务较多时，难以求出真正可行的最优解，于是F,G,H 为我们设计的已知最优解的测试集，这样已知最优解的测试集可以为我们在算法演化过程中出现的各种情况提供合理的对照和启发。容器数量指可供调度的资源总数，巡检任务数指不考虑周期性带来的重复任务的任务总数，而周期任务数则为在该测试集情景下需要运行所有任务的数量，周期性任务重复计算。

为了保证阐述更加清晰，我们选取 E 测试集中的一部分数据进行分析。 6

表 2 任务数据集

任务运算模块周期运行时长摄像头 id 函数 id

fg0_f0_cam0 op_gm_m9，op_d_c22，op_foo10_c22，op_cm_m9_c22，op_util_m9_c22_p0 20 3 cam0 foo10

fg0_f1_cam0 op_gm_m9，op_d_c22，op_foo11_c22，op_cm_m9_c22，op_util_m9_c22_p0 30 9 cam0 foo11

fg0_f2_cam0 op_gm_m15，op_d_c22，op_foo27_c22，op_cm_m15_c22，op_util_m15_c22_p0 35 6 cam0 foo27

fg0_f3_cam0 op_gm_m15，op_d_c22，op_foo20_c22，op_cm_m15_c22，op_util_m15_c22_p0 40 8 cam0 foo20

fg0_f0_cam1 op_gm_m9，op_d_c86，op_foo10_c86，op_cm_m9_c86，op_util_m9_c86_p0 30 6 cam1 foo10

fg0_f1_cam1 op_gm_m9，op_d_c86，op_foo11_c86，op_cm_m9_c86，op_util_m9_c86_p0 30 4 cam1 foo11

fg0_f2_cam1 op_gm_m15，op_d_c86，op_foo27_c86，op_cm_m15_c86，op_util_m15_c86_p0 35 4 cam1 foo27

fg0_f3_cam1 op_gm_m15，op_d_c86，op_foo20_c86，op_cm_m15_c86，op_util_m15_c86_p0 25 10 cam1 foo20

如上表 2 所示，每一个巡检任务有着自己的周期，运行时长，被分配给的摄像头以及所要运行的函数。其中表 2 第二列表示每一个任务需要运行的运算模块，每一

个运算模块对应着下图表 4 中的第一列，而每一个运算模块需要占用相应的资源如CPU 数，GPU 数，显存来完成任务。

表 3 容器数据集数量 GPU 数显存 CPU 数 CPU 内存

17 100 15360000 1200 50331648

如上表 3 所示，此为该测试集情境下我们所拥有的容器的参数。

表 4 负载数据集

运算模块 GPU 数显存 CPU 数 CPU 内存

op_util_m33_c0_p0 0.0026950000 0.0000000000 0.0005500000 0.0000000000

op_util_m15_c0_p0 0.2062500000 0.0000000000 0.0062500000 0.0000000000

op_util_m15_c0_p1 0.2062500000 0.0000000000 0.0062500000 0.0000000000

op_util_m14_c0_p0 0.0851250000 0.0000000000 0.0025000000 0.0000000000

op_util_m16_c0_p0 0.4281250000 0.0000000000 0.0087500000 0.0000000000

op_util_m12_c0_p0 1.2900000000 0.0000000000 0.0200000000 0.0000000000

op_util_m36_c0_p0 0.0062500000 0.0000000000 0.0625000000 0.0000000000

op_util_m35_c0_p0 0.0967200000 0.0000000000 0.0026000000 0.0000000000

op_util_m9_c0_p0 0.0504900000 0.0000000000 0.0034000000 0.0000000000

op_util_m0_c1_p0 0.4216000000 0.0000000000 0.1700000000 0.0000000000

op_util_m8_c1_p0 0.6688000000 0.0000000000 0.2200000000 0.0000000000

op_util_m17_c1_p0 6.7700000000 0.0000000000 0.2000000000 0.0000000000

op_util_m16_c1_p0 0.4281250000 0.0000000000 0.0087500000 0.0000000000

op_util_m7_c1_p0 0.5400000000 0.0000000000 0.0500000000 0.0000000000

如上表 4 所示，此为该测试集情境下每一个运算模块所需要占用的各个资源数。

通过表 2 表 3 表 4 显示的信息，我们可以进一步了解模型情景设计，从而大致理解我们所需要测试的资源调度算法就是将不同的任务以及其运算模块在不同时间分 7

配给不同的容器，并且希望能够达成容器的按时间平均资源占用率最小化，同时尽量地缩小容器之间或者不同时间资源占用率的方差。

综上，我们开发的测试平台通过于开发人员合作，设计出了相对合理且完备的测试集以供各位算法开发工作者们测试自己的算法性能，节省了算法工作者们自己寻找并设计测试集的工作，并通过测试集的整合集思广益以设计

出更合理的测试集。

3.2 系统核心流程

图 2 代码测试前后端流程图在本次测试平台的搭建中，系统最为核心的部分就是接收用户提交的算法代码，并在服务器上选取相应测试集运行用户提交的代码，最后给予用户算法运算结果的

反馈。我们可以看到，在用户提交代码之后，代码会通过前端调用接口首先进入我们后端的 CRUD 模块处理完相应逻辑之后将提交记录暂时存储至 Redis 数据库中，与此同时，CRUD 模块会向 RabbitMQ 发送消息记录处理该次代码提交，然后 RabbitMQ

进行消息的分发。而分发的终端则为我们的代码测评服务器。代码测评服务器在通过消息队列获取到用户提交的代码之后，会在服务器搭建的沙盒环境下保证安全稳定

地运行代码。在最后，沙盒中代码运行的结果将会通过消息队列返回给 CRUD 模块

中的消费者，代码结果在 Redis 数据库中进行暂时存储。同时通过调用 python 脚本，

用户提交的代码结果会最终进行可视化并反馈给前端。

8

其中 CRUD 模块包含对于消息队列的生产者与消费者实现类，从而及时向评测

机通知代码提交信息与获得代码提交结果。其余则是依据后端逻辑正常创建的的不

同 Controller，Service，Mapper 等实现类或接口，分功能用于处理网页后端中的大部分逻辑。

而评测机模块则同样拥有消息队列的生产者与消费者实现类，用于向 CRUD 模

块发送代码提交结果与接收代码提交信息。最为重要的是，在评测机的消费者实现类

中我们通过 C++ 实现的沙盒来对代码进行安全的运行与评测。

可以看到，我们在测试平台的搭建中使用了如 RabbitMQ, Redis, 沙盒等组件，这些核心组件在我们系统的流程中发挥了重要的作用：使我们的测试平台具有更强的高并发性能，数据库响应能力以及安全稳定环境。这些都是我们在搭建测试平台时由于任务的特殊性所需要的特质。接下来我们将阐述这些组件如何对我们的平台搭建进行了优化与完善。

3.3 高并发能力优化

图 3 rabbitMQ 结构在测试平台中提供高并发能力支持的组件主要是 RabbitMQ[12].RabbitMQ 是一个在 AMQP（高级消息队列协议）基础上完成的可复用的企业消息系统，是当前最主流的消息中间件之一。考虑到 RabbitMQ 是一个对比 Kafka[13]，RocketMQ[14]等消息队列相对轻量级的消息队列，并且可以通过 Exchange 模块进行灵活的路由配置。除此之外，对于性能方面，即便不如 Kafka 等能够处理海量消息的消息队列组件，对于我们代码测评平台来说，每秒钟处理几万到十几万条消息已经足够能够满足代码测 9

评的需求。[15][16]因此我们选择 RabbitMQ 作为我们的消息中间件来处理复杂的代码评测逻辑.

在用户通过我们的平台交互页面提交代码后，通过我们的后端处理逻辑，管理代码提交的模块（即 producer）会生成相应的包通过我们以 RabbitMQ 组件实现的消息队列到达最后的代码测评模块（即通过 @RabbitListener 生成的 consumer）.

作为总结，RabbitMQ 作为一个非常强大的消息中间件，在我们的平台搭建过程中发挥了多重作用。一方面，RabbitMQ 帮助我们实现了应用解耦，防止我们调用一

个接口失败导致整个过程失败。另一方面，RabbitMQ 帮助我们实现了异步处理，由于代码评测需要消耗相当一段时间去运算，尤其是对于我们需要评测的启发式算法和演化算法，都需要很长时间去迭代。

---

2. 多算子协同评测的资源调度平台模拟评估系统_第2部分

总字数：**4275**

相似文献列表

去除本人文献复制比：0.9%(38)　文字复制比：0.9%(38)　疑似剽窃观点（0）

1	111180127_孙佳宇_通信 孙佳宇 - 《大学生论文联合比对库》- 2015-05-21	0.9%（38） 是否引证：否

原文内容

那么我们的 RabbitMQ 能够使得应用间并发处理消息，减少处理时间，更快地给用户提供反馈。

3.4 数据库响应优化

为了保证我们的算法测试平台在高负载的情况下依然能够高效运行，我们采用

了 Redis[17]数据库来对整个项目的数据存储进行一个合理的优化。Redis 是一个开源

的基于内存的键值对存储系统，是跨平台的非关系型数据库，其可以存储字符串，哈希，列表，集合和有序集合等类型。Redis 由于其完全基于内存，并且采用单线程多路 I/O 复用模型，使得其相比 MySQL[18]的读写性能和并发性能非常高.[19]

图 4 redis 使用机制 10

由于其优秀的读写性能和并发性能，如图 4 所示，我们在本次项目开发中作为我们持久性数据库 MySQL 的缓存数据库以提高整体的数据库访问速度。例如我们在本次项目中对于每次代码评测都要频繁地刷新数据库中关于该次评测的相关信息，当用户提交代码并且查看代码评测记录较为频繁时，我们会选择将较为近期的代码评测记录存储到 Redis 数据库中以满足用户的频繁查询。而当 Redis 数据库中的信息存在达到一定时间后就会从 Redis 数据库中删去，并且加入到持久化存储的 MySQL 数据库中，从而保证 Redis 数据库不占用过多内存的同时能够高效地为我们提供代码提交记录的查询操作。

另一方面，由于用户本身提交的代码本身作为高容量的数据类型，并不是 MySQL

善于处理的数据类型，于是将用户近期提交的代码存储入 Redis 能够进一步地提高数据库的查询速度，保证后端每个 API 调用时数据的返回速度。

3.5 安全运行代码

图 5 沙盒的基本思想理念如图 5 所示，沙盒 (Sandbox)[20]作为一个抽象化的远程代码运行环境，一般在服 1 1

务器或者虚拟机上被单独划分出一份空间以避免影响整体的代码运行甚至污染服务

器虚拟机。[21]而在本次项目的搭建中，我们使用 C++ 这类偏底层语言进行沙箱的构

建，其中重要的思想如下:

• 调用限制：为了保证用户提交的代码不调用危险的系统命令，我们在运行代码中涉及到文件，网络，用户，进程等操作前检查系统调用的合法性，防止不合法地调用内核态命令导致环境崩溃。其中 ptrace 由于效率较低，会让代码的运行时间增加很多。我们在此使用 seccomp 命令，主动地在代码中加载策略。通过加载动态链接库的方式在用户的 main 函数前执行自己的代码。同时采用白名单模式，使用 seccomp 参数指定可以执行的命令，防止部分危险的系统调用。

• 资源限制：为了保证用户提交的代码不占用过多的系统资源，我们可以采取多种指令来限制各项资源的使用。例如我们可以通过 setitimer 来限制程序的运行时间。同样的，我们可以通过调用 setrlimit 函数来限制进程的资源占用，并且

通过 RLIMITAS 来来限制进程最大内存地址空间，RLIM IT

FSIZE 来限制进程最大输出或者写文件的大小。

• 代码编译运行：在检查过一系列限制之后我们则可以使用 exec 系函数执行命令，并且通过父进程监控子进程，来控制代码的运行时间，并且子进程继承父进程一系列限制设定，从而保证新进程在安全的环境下继续运行。

3.6 前端搭建

综上，后端的基本框架基本已经阐述完毕，其中用户提交代码并且获得反馈这一

重点功能的具体实现流程。除此以外，我们还结合 Vue.js 以及 python 数据可视化工具开发了高效可用的用户前端交互页面。此前端交互页面不仅保证了用户的重点功能体验，并且提供了竞赛，论坛，公告等一系列交互机制以供算法开发者们交换算法

优化心得。同时数据可视化工具的使用使得算法优化结果能够更加直观地反馈给每

一个用户，从而对算法的进一步优化得到启发，提高算法开发的效率。 12

4. 实现以及结果

下面我们将对我们的平台实现进行一定的展示以及对与我们对算法的分析进行

一定的讨论。

4.1 前端演示

图 6 主页面展示图 7 测试页面展示我们搭建的算法测试平台名为 AlgoTest, 用户可以在页面右侧进行注册以及登陆操作，同时针对管理员端，我们有着特殊的操作权限能够发布公告并且对于测试集进行一定的调整和测试，保证测试集能够实时地进行更新以对算法进行进一步的优化测试。在记录板块用户则可以实时调取自己已经提交的代码记录并获取对算法的一

系列分析。同时我们实现了讨论版，竞赛等一系列附加功能，帮助用户之间相互讨论并且及时反馈问题。 13

图 8 测试集描述展示图 9 代码提交展示

4.2 算法测试结果分析演示

对于算法的评估方式，我们已于前言中具体介绍，即计算在该算法规划下容器按时间平均资源利用率，而这个数值在算法演化完毕并认定为正确解后后将会被系统统一按公式计算并在记录中给用户以反馈，具体可见图 10.

然而单一的数值虽然作为评估算法以及比较不同算法性能的标准足矣，但是却不能给算法工作者们很好的启发以及直观的感受。于是我们算法测试平台除了为用户提供安全稳定的测试环境以外，还将对算法运行的日志进行打印以及对算法的优图 10 查询记录展示 14

图 11 优化结果演示图表横坐标：时间（s）纵坐标：资源占用率

化效果进行图表绘制，从而能够使得算法工作者们获得更加直观的优化结果演示，启

发算法工作者们对算法进行进一步的优化与提升。

由于我们测试的算法多为启发式算法以及演化算法或者两者之间的结合，其中算法的每一次迭代都能够得到一个新的问题的解，为了对比优化迭代的效果，我们采取对初始解以及最终解按时间资源占用率的变化曲线图进行绘制，以直观得出在每个时间段算法迭代实现了多少的资源占用率优化。我们在问题描述中要求用户输出初始解以及最终解，从而绘制对于各种启发式以及演化算法适用的变化曲线图。此处

我们采取最基础的遗传算法迭代 100 次得到的结果作为演示。如图 11 所示，其中蓝

色曲线为初始解曲线, 即 First Generation，橙色曲线为最终解曲线, 即 Last Generation。

横轴的 time 代表测试集情景设定中的时间，而纵轴 result 则是通过前言中提及的定义计算得出的平均容器资源占用率。由图 11 中可见在负载达到局部最大值时末代相

对于初代优化的趋势明显，同时在不同时间段内负载存在着较大的差距，显示出有进

一步优化的可能性。

同样的，我们采取最基础的遗传算法迭代 100 次得到的结果作为演示。图 12 则

为运行任务数量随时间的变化趋势曲线图，横轴的 time 表示测试集情景设定中的时间，而纵轴 task numbers 则为该时间下所有容器被安排运行的任务总数，可以通过用

户提交的最终解统计得出。可以看出任务数量存在着一定的周期性变化规律，这与每

一个任务周期性运行的性质是相吻合的。因此，通过观察一个周期内任务数量随时间的变化趋势以及对比不同周期时间段的任务数量差异，我们能够了解大致任务安排 15

图 12 任务运行演示图表横坐标：时间（s）纵坐标：运行任务数量

的情况。从而有时当用户不能够对与算法的优化结果获得很好的启发时，通过单位时

间内运行的任务数量能够对算法的优化结果有一定直观的了解，从而去修复代码中

可能存在的逻辑缺陷抑或是可以进行优化的目标。

4.3 应用部署

图 13 部署演示我们项目后端的部署主要需要 Java,Springboot,MySQL,Redis,C++,Rabbit MQ，python

等依赖。同时，为了方便部署，我们提供了基于 Docker 的本地快捷部署方式，可通过 docker compose 快速搭建后端服务器，从而方便随时随地进行平台测试。 16

图 14 部分部署参数

5. 结论

综上所述，我们的项目成功搭建了一个针对城市监控计算资源进行合理调度的

资源调度算法进行合理测试以及反馈的平台。在搭建这一平台的过程中我们采用了

Redis, Rabbit MQ, Springboot 等组件，同时使用 python 对于算法测试结果进行合理的

可视化，从而帮助进一步的算法优化。总而言之，该平台将在资源调度算法测试中发挥出重要的作用，提供算法工作者们更多便捷。 17

参考文献

[1] MAZIDI A, FAKHRAHMAD M, SADREDDINI M H. A meta-heuristic approach to

CVRP problem: local search optimization based on GA and ant colony[J]., 2016.

[2] AUGERAT P, BELENGUER J M, BENAVENT E, et al. Separating capacity con-

straints in the CVRP using tabu search[J]. European Journal of Operational Research, 1998, 106(2-3): 546-557.

[3] CATTARUZZA D, ABSI N, FEILLET D, et al. A memetic algorithm for the multi trip

vehicle routing problem[J]. European Journal of Operational Research, 2014, 236(3): 833-848.
[4] BELLO I, PHAM H, LE Q V, et al. Neural combinatorial optimization with reinforcement learning[J]. arXiv preprint arXiv:1611.09940, 2016.
[5] D O COSTA P R, RHUGGENAATH J, ZHANG Y, et al. Learning 2-opt heuristics for the traveling salesman problem via deep reinforcement learning[C]//Asian Conference on Machine Learning. 2020: 465-480.
[6] KIM M, PARK J, PARK J. Sym-nco: Leveraging symmetricity for neural combinatorial optimization[J]. arXiv preprint arXiv:2205.13209, 2022.
[7] KWON Y D, CHOO J, KIM B, et al. Pomo: Policy optimization with multiple optima for reinforcement learning[J]. Advances in Neural Information Processing Systems, 2020, 33: 21188-21198.
[8] LIU D, TAN K C, HUANG S, et al. On solving multiobjective bin packing problems using evolutionary particle swarm optimization[J]. European Journal of Operational Research, 2008, 190(2): 357-382.
[9] CRAINIC T G, PERBOLI G, TADEI R. TS2PACK: A two-level tabu search for the three-dimensional bin packing problem[J]. European Journal of Operational Research, 2009, 195(3): 744-760.
[10] ZHAO H, SHE Q, ZHU C, et al. Online 3D bin packing with constrained deep reinforcement learning[C]//Proceedings of the AAAI Conference on Artificial Intelligence: vol. 35: 1. 2021: 741-749.
[11] ZHANG J, ZI B, GE X. Attend2Pack: bin packing through deep reinforcement learning with attention[J]. arXiv preprint arXiv:2107.04333, 2021.
[12] DOSSOT D. RabbitMQ essentials[M]. Packt Publishing Ltd, 2014.
[13] KREPS J, NARKHEDE N, RAO J, et al. Kafka: A distributed messaging system for log processing[C]//Proceedings of the NetDB: vol. 11: 2011. 2011: 1-7. 18
[14] 历启鹏. Apache RocketMQ 流计算处理[J]. 软件和集成电路, 2018(8): 76-77.
[15] DOBBELAERE P, ESMAILI K S. Kafka versus RabbitMQ: A Comparative Study of Two Industry Reference Publish/Subscribe Implementations: Industry Paper[C/OL]//DEBS '17: Proceedings of the 11th ACM International Conference on Distributed and Event-Based Systems. Barcelona, Spain: Association for Computing Machinery,2017: 227-238. https://doi.org/10.1145/3093742.3093908. DOI: 10.1145/3093742.3 093908.
[16] T S, K S N. A study on Modern Messaging Systems- Kafka, RabbitMQ and NATS Streaming[Z]. 2019. arXiv: 1912.03715 [cs.DC].
[17] CARLSON J. Redis in action[M]. Simon, 2013.
[18] MYSQL A. MySQL[Z]. 2001.
[19] PAKSULA M. Persisting objects in redis key-value database[J]. University of Helsinki, Department of Computer Science, 2010, 27.
[20] SPREITZENBARTH M, FREILING F, ECHTLER F, et al. Mobile-sandbox: having a deeper look into android applications[C]//Proceedings of the 28th annual ACM symposium on applied computing. 2013: 1808-1815.
[21] GONG L, MUELLER M, PRAFULLCHANDRA H, et al. Going beyond the sandbox: An overview of the new security architecture in the Java development kit 1.2.[C]//USENIX Symposium on Internet Technologies and Systems: vol. 2. 1997. 19

致谢

大四是我人生中十分特殊的一年。在大四学期开学不久，我在国庆回家期间被检查出胃部出现了一定状况，从此日日在医院奔波接受治疗，而进一步的医治可能需要等到暑假期间调养治疗。也因此大四的学业工作都受到了一定程度的影响，不能够像我想象中一样去完美地为我这四年的学习画上一个句号。

但是我还是要感谢在本科学习期间一直为我提供精神与学业上的支持的人。他们的关怀带着我渡过了人生中对我来说最为艰难的一段时间，让我坚持下来完成学业。这些帮助是我无论如何都感激不尽的。

首先我要感谢我的导师宋轩老师。从大一到大四，他都对我有着无微不至的关怀，从生活到学业，老师带我领略了本科生活中一系列的精彩，为我在学习上提供衷心的意见。我还要感谢范子佩老师以及张凌宇老师。范子佩老师带着我探索和研究双曲超图神经网络的应用，让我熟悉了科研的流程，掌握了深度学习所需要的技能。

而张凌宇老师带领着我完成了本科论文的工作，让我对于工业方面的代码构建有了更深层次的了解。同时还要感谢实验室的很多学长，如李永康学长，像朋友一样对我进行着鼓励和引导，从而完成一系列的科研工作。正是有了这些老师和学长的帮助，

我才能够成功的完成一系列科研任务，并且收获了卡耐基梅隆大学 ECE 专业的硕士

offer，从而能够进一步地深造。

其次我要感谢我身边很多对我鼓舞和支持的同学和朋友们，他们在我生病期间也对我一直关怀照顾，主动分担我的很多工作。我深深地知道这样的朋友是多么难能可贵，即便在此不便提及你们的姓名，但你们在大学四年中和我一起创造出来的快乐，对我一系列的支持与包容我都永生难忘。

最重要的是，我要感谢我的父母，因为我的病情，我看到父母因忧愁日渐衰老疲惫，但是他们仍然时时刻刻对着我露出笑容，鼓舞着我即便再难再累也要坚持下去。

这种情感可能难以言喻，但却会永远镌刻在我的心间。

再次，感谢我身边一直支持着，关怀着我的人，我会带着你们的支持再次扬帆起航。 20

说明： 1.总文字复制比：被检测论文总重合字数在总字数中所占的比例

2.去除引用文献复制比：去除系统识别为引用的文献后，计算出来的重合字数在总字数中所占的比例

3.去除本人文献复制比：去除作者本人文献后，计算出来的重合字数在总字数中所占的比例

4.单篇最大文字复制比：被检测文献与所有相似文献比对后，重合字数占总字数的比例最大的那一篇文献的文字复制比

5.指标是由系统根据《学术论文不端行为的界定标准》自动生成的

6.红色文字表示文字复制部分;绿色文字表示引用部分;棕灰色文字表示作者本人已发表文献部分

7.本报告单仅对您所选择比对资源范围内检测结果负责

amlc@cnki.net

check.cnki.net

http://check.cnki.net/