云环境下协同作业的密码服务优化调度算法

曹晓刚[1,2,3]，李凤华[1,2,3]，耿魁[1,3]，李子孚[1,3]，寇文龙[1,3]
（1.中国科学院信息工程研究所，北京 100085；2.中国科学院大学网络空间安全学院，北京 100049；
3.网络空间安全防御重点实验室，北京 100085）

摘 要：针对云环境下密码按需服务中多个计算作业协同服务的需求，提出了多密码作业协同服务的调度算法，能够充分应对密码算法种类多、并发需求高、作业随机交叉和作业负载瞬时激增等云环境下的新挑战。考虑每个密码计算作业之间的依赖关系、密码作业的完成时间需求以及密码计算单元的最大算力，以最小化能耗、迁移成本和瞬时激增负载的适应度为优化目标，将多密码作业协同服务调度问题建模为多目标优化的作业流调度问题，并提出“选择–排序”两阶段调度算法，在选择阶段，采用改进NSGA-III算法为密码计算作业选择合适的计算单元，在排序阶段，根据作业紧迫程度决定执行顺序。仿真结果表明，所提调度算法在能耗、迁移成本和对瞬时激增的作业负载的适应度方面优于传统调度算法。

关键词：云计算；密码按需服务；作业流调度；NSGA-III
中图分类号：TN92
文献标志码：A
DOI：10.11959/j.issn.1000–436x.2024083

Cryptographic service optimization scheduling algorithm for collaborative jobs in cloud environment

CAO Xiaogang[1,2,3], LI Fenghua[1,2,3], GENG Kui[1,3], LI Zifu[1,3], KOU Wenlong[1,3]
1. Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100085, China
2. School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
3. Key Laboratory of Cyberspace Security Defense, Beijing 100085, China

Abstract: In response to the demand for collaborative computation of multi-cryptographic jobs in cryptographic on-demand services within a cloud environment, a multi-cryptographic job collaborative scheduling algorithm was proposed. This algorithm effectively addressed new challenges in cloud environments, such as a variety of cryptographic algorithm types, high concurrency demands, random cross-job interactions, and sudden increases in workloads. Considering the dependencies among jobs, makespan of jobs and computational power of computing units, the scheduling problem for multi-cryptographic job collaborative service was modeled as a multi-objective optimization workflow scheduling problem. A two-stage “select-sort” scheduling algorithm was proposed. In the selection stage, the improved NSGA-III algorithm was employed to select computing units for cryptographic computing jobs, and in the sorting stage, the execution order was determined based on the urgency of jobs. Simulation results demonstrate that the proposed algorithm outperforms traditional scheduling algorithms in terms of energy consumption, migration costs, and adaptability to transient surges in loads.

Keywords: cloud computing, cryptographic on-demand service, workflow scheduling, NSGA-III

收稿日期：2024–02–02；**修回日期**：2024–03–11
通信作者：寇文龙，kouwenlong@iie.ac.cn
基金项目：国家重点研发计划基金资助项目（No.2021YFB3100300）；国家自然科学基金资助项目（No.62202463）；中国科学院青年创新促进会人才基金资助项目（No.2023168）
Foundation Items：The National Key Research and Development Program of China (No.2021YFB3100300), The National Natural Science Foundation of China (No.62202463), Youth Innovation Promotion Association, Chinese Academy of Sciences (No.2023168)

0　引言

云计算以其高灵活性、高可扩展性和高性价比的特性，已经成为当今信息技术领域中不可或缺的重要基础设施。随着大量数据上云，如何使用密码技术保护云上数据安全成为业界关注的焦点之一。Bleikertz 等[1]在 2013 年提出密码即服务（CaaS, Cryptography-as-a-service）的概念，探索以服务方式交付密码功能的可能性。随后加密即服务（EaaS, encryption as a service）[2]和密钥管理即服务（KMaaS, key management as a service）[3]等新的概念相继被提出。我国也在2019年发布《云密码服务技术白皮书》[4]探索云密码架构、场景及发展趋势。根据市场研究公司 FMI（Future Market Insights）报道，全球云加密市场收入在2023年达到31亿美元，预计将于2033年达到456亿美元[5]。

云密码服务把密码能力变成云中的一种资源服务，用户可以按需定制自己的密码服务方案，以“租用密码服务”的方式保障自己的数据安全。然而随着企业规模不断扩大，业务需求不断增加，单一的密码算法已经难以应对日益复杂的业务对数据安全提出的需求，用户通常需要将多种加密算法组合使用，以保证数据安全。在云计算场景中，密码计算服务面临多种场景需求，如传输加密、传输解密、存储加密和存储解密等，不同类型场景又可能存在多种密码算法协同计算的需求。除此之外，密码算法有多种实现方式，如国密算法和国际算法，同时，在传输时面临一对一加密传输或一对多加密多播场景，存储时面临多种密码算法协同加密存储或多密钥多备份存储等，这为云环境下密码资源分配和任务调度带来巨大挑战。除此之外，现行数据保护技术大多采用密码设备旁路调用的方式对数据进行加解密，但这种加解密方式数据回路远、调用时间长、传输效率低，因此本文采用“不走回头路”的密码服务方式，将密码服务串行在业务流程中，提高密码服务效率。

用户为了实现复杂的密码计算任务，需要定制多密码任务协同计算的密码服务，比如用户上传数据并多备份加密存储，如图1所示。用户首先向调度系统发送密码计算资源定制需求，如需要哪些密码算法、每种密码算法的算力需求、各密码算法之间的协同关系和服务时间等。调度系统根据定制加密需求中涉及的计算服务系统、加密存储系统和密码计算设备集群的运行状态，为密码计算作业分配指定的密码计算设备，如①~⑧所示。完成资源分配过程后，该作业流上后续的密码计算任务均由已分配的密码计算设备提供密码运算能力。在数据加解密的过程中，计算服务器接收用户上传的使用会话密钥加密的密文数据 $ENC_{k\text{-}sess}(D)$，将该密文数据发送给密码计算设备进行一系列密码计算后，再将使用不同存储密钥加密的密文数据存储在不同的存储服务器中，除此之外，还需要使用哈希、签名等密码算法保证存储文件的不可篡改性。

图1场景涉及密码设备之间多密码任务的协同计算，如图2所示。该密码计算作业由5个密码计算任务协同计算完成，首先密码计算单元（CCU, cryptographic computing unit）CCU-0从使用会话密钥加密的密文解密得到明文 D，并将解密后的明文发送给密码计算单元 CCU-2。接下来 CCU-2 和 CCU-0 并行运算，分别使用不同的存储密钥 k1-stg、k2-stg 对明文加密，同时 CCU-2 还计算明文的哈希值，并对哈希值签名，保证文件的不可篡改性。最后将加密后的不同密文保存到分布式加密存储系统中。

密码服务属于业务服务的伴生服务，需要保证密码运算效率以降低密码服务对正常业务服务的影响，因此，密码服务调度对性能要求较高，而云环境中不免出现瞬时激增的负载，从而对密码计算资源带来运算压力，如果负载增加过大，会严重影响服务质量。除此之外，在云计算场景中，密码服务还面临着密码任务协同关系复杂、并发数量高、密码计算设备的计算能力和支持的算法类型差异大等特点，当前的密码计算资源管理和任务调度方案难以应对上述需求。在低碳转型和环保意识不断增强的今天，如何在保证服务质量的前提下降低能耗也是优化调度要解决的问题之一。

针对云环境下复杂的密码服务调度需求，本文提出了高效的多密码任务协同服务的优化调度算法。首先将多密码任务协同调度问题建模为多目标优化的作业流调度问题，通过求解该问题，生成优化调度策略，实现降低能耗、提高抗瞬时激增负载的目标，保障云密码服务的高效可用。然后在调度阶段，提出了基于紧迫程度的优先级调度算法，保证关键任务优先完成。本文的主要贡献如下。

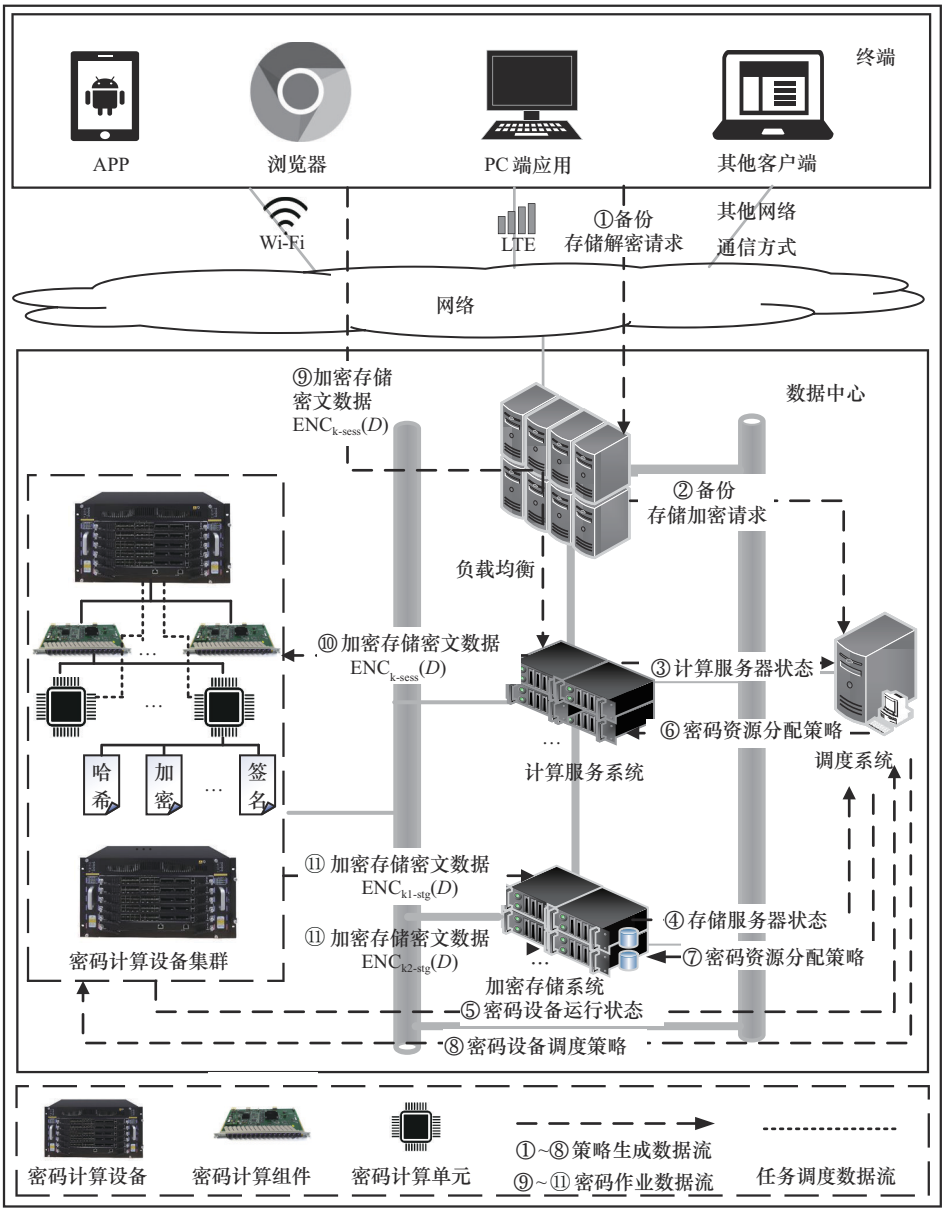

图1　文件加密上传多备份加密存储场景

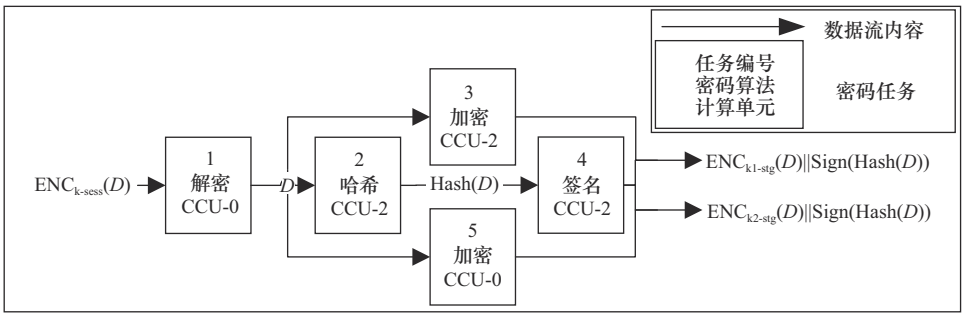

图2　文件加密上传加密存储密码算法协同关系

1) 将面向云计算的多密码计算任务协同调度问题建模为作业流调度模型。该模型综合考虑计算任务的算法算力需求及密码计算任务间协同场景，以降低能耗、降低迁移成本（MC, migration cost）及提高抗瞬时激增负载能力为目标，生成优化调度策略。针对密码计算单元多核并行、多密码任务随

机交叉的特点，使用排队论预测任务等待时间，优化调度策略效果。

2) 提出了瞬时激增负载适应度指标，来评估调度算法对云环境中瞬时激增负载的应对能力。采用关键任务均衡度来衡量调度策略对瞬时激增负载的适应度，并与基于紧迫程度的排序算法相结合，提高在云计算环境下对瞬时激增任务负载的应对能力。

3) 提出了“分配–排序”两阶段的调度算法 DRFF-NSGA-III-PS（double random first fit based NSGA-III selection and priority based sort）。在资源分配阶段，以能耗、迁移成本及关键任务平衡度为目标，生成调度策略，为了提高收敛速度，提出了 DRFF-NSGA-III 算法；在初始化种群阶段，采用双随机首次适配（DRFF, double random first fit）算法，保证初始化种群的个体质量；在任务排序阶段，根据任务紧迫程度决定执行顺序，保证关键作业和即将超时的任务优先完成。

4) 在 OMNet++模拟环境中实现了调度方案，并验证了本文算法的可行性和优越性。

1　相关工作

作为提高云计算环境下设备利用率的重要手段，调度算法是学术界聚焦的重点领域。任务调度可分为单任务调度和多个任务协同的作业流调度。本节回顾和分析了云计算环境中任务调度和调度问题中智能优化算法的相关工作。

1.1　任务调度

李建鹏等[6]提出了二级密码服务调度架构，根据密码算法类型将任务分配给指定的二级子系统，每个子系统根据任务优先级决定任务处理顺序。孙德洋等[7]根据处理速率对计算节点分级，并根据优先级及等待时间为任务分配计算节点。寇文龙等[8]提出了基于优化熵值法密码设备的服务能力评估系数，并基于该系数动态生成密码作业调度策略和设备动态重构策略。Narayanan 等[9]采样并分析了训练深度学习模型在不同加速器上的性能差异性，并使用基于轮的调度机制来确保训练深度学习模型时资源分配的最优性。Xiao 等[10]通过将作业打包的方式减少碎片化资源使用，以提高集群利用率。然而上述的调度方案大都没考虑需要多个任务协同服务时的调度问题。

需要多个任务协同完成指定目标的问题通常被建模为作业流调度问题。关川江等[11]根据密码运算相互依赖的特点提出了关联数据本地化的多密码作业调度算法，将相互依赖的任务分配在同一个计算节点上，降低了传输成本。Hussain 等[12]针对数据中心地理位置分散且不同位置电价不同的问题，提出了考虑截止日期约束的能量感知作业流调度（DEWS, deadline-constrained energy-aware workflow scheduling）算法，以实现降低能耗的目标。Li 等[13]针对电价随地理位置和时间不同的特点，采用自适应局部搜索方法，优化了搜索过程中的任务队列。Sun 等[14]通过查找满足用户所需的截止日期和预算约束的调度策略，并利用规划成功率来显示所提算法的有效性。Chen 等[15]在对作业流调度问题建模时同时考虑抢占和非抢占 2 种场景，并提出了基于空闲时间块的方法为任务搜索可用的资源。然而上述方案中并未考虑云环境下负载激增对服务质量带来的影响。

1.2　智能优化算法

任务调度问题是个典型的 NP-Hard 问题，因此，很多人工智能算法被提出用来寻找次优解。Xie 等[16]针对云作业流调度问题提出了自适应解码偏向随机密钥遗传算法，采用了限定值范围、基于级别和启发式的群体初始化等技术来提高进化效率。Anwar 等[17]提出混合生物启发式多目标优化，来优化科学作业流程的多目标调度。Choudhary 等[18]将引力搜索算法和异构最早完成时间相结合，来实现调度问题中任务完成时间和成本的最小化的目标。Verma 等[19]针对基础设施即服务（IaaS, infrastructure as a service）环境下多目标优化作业流调度问题提出了基于非优势排序的混合多目标粒子群优化算法，以实现任务完成时间和成本的平衡。Mohammadzadeh 等[20]提出了海鸥优化算法和蚱蜢优化算法相结合的混合多目标优化算法，解决了多云环境中的作业流调度问题，并采用膝点法从帕累托前沿中选取调度策略。You[21]提出了崩塌分层优势资源公平排队（collapsed H-DRFQ, collapsed hierarchical dominant resource fair queueing）算法对传统公平排队调度算法进行优化，以适应网络功能虚拟化场景下的新挑战。Calzarossa 等[22]在建模作业流调度问题时考虑到性能的不确定性，提出了多目标约束优化问

题的概率表述，以任务完成时间和成本为目标，结合蒙特卡罗方法和遗传算法确定最终的调度策略。Belgacem等[23]将异构最早结束时间和蚁群算法相结合，解决云计算平台上作业流最大完成时间和虚拟机使用成本之间的权衡关系。

上述方案大都忽略了以下2个问题：1) 云计算环境下可能存在的瞬时激增负载对作业调度服务质量带来的挑战；2) 在预测任务完成时间时，直接使用平均算力计算并不符合密码服务场景，因为密码计算设备支持的配置多、算法类型多、算法差异大，用平均算力预测执行时间或排队时间会带来较大误差。

2　系统模型

数据中心与密码任务调度相关部分包括负载均衡、计算服务系统、加密存储系统、密码计算设备集群和调度系统5个部分，其中，负载均衡为数据中心对外数据出入接口，完成数据分流功能；计算服务系统是云计算环境中完成计算相关任务系统的统称，如视频编解码、神经网络训练、网站解析响应等；加密存储系统是存储用户数据系统的统称，包括数据库系统、文件系统等；密码计算设备集群是完成密码算法运算的设备集群；调度系统为生成密码任务调度策略的策略配置中心。

在云计算场景中，负载均衡、计算服务系统或加密存储系统将需要密码运算的作业发送到设备集群，根据调度策略分配给指定密码计算设备完成密码运算后，再将处理后的数据发送给指定业务设备。以加密上传多备份加密存储为例，运行在调度系统上的调度算法根据作业请求和设备运行状态生成调度策略，并将调度策略发送给密码计算设备（CCD, cryptographic computing device），密码计算设备接收到密码作业后，按照调度策略分配给部署在密码计算组件（CCM, cryptographic computing module）上指定的CCU执行，计算单元上运行的优先级排序算法根据任务优先级决定执行顺序，最后将完成的加密作业发送给指定的接收端。

针对云环境下复杂的密码计算任务调度需求，本文将多密码任务组合服务调度问题建模为作业流调度问题，下面给出调度流程中涉及的内容及定义。

2.1　模型描述

根据任务之间的协同关系，以降低瞬时激增负载对服务质量影响和降低能耗为目的，优化密码计算任务的配置，从而达到优化目标，本文将协同任务调度算法建模为多目标优化的作业流调度模型，通过求解多目标优化问题得到的调度策略就是优化后的调度策略。

定义1　密码计算任务（CCT, cryptography computing task）：指需要基于密码学算法来完成的计算任务，如对数据加密、解密、计算哈希等。CCT由四元组 $T=(A,W,C,K)$ 表示，其中，A 为所需要的密码算法类型，W 为待处理的数据大小，C 为所需密码算法的算力大小，K 为密钥空间大小，包括非对称密钥和对称密钥2种，本文将保存这2种密钥的存储区域统称为密钥空间。

定义2　密码计算作业（CCJ, cryptography computing job）：指由一组密码计算任务协同完成的密码运算。CCJ可以用三元组 $J=(G,D,S)$ 描述，其中，有向无环图（DAG, directed acyclic graph）$G=(T,E)$ 表示组成作业的任务之间的依赖关系，DAG的点 $T=\{T_1,\cdots,T_{N_T}\}$ 表示一系列任务节点集合，N_T 表示任务数量，每一个节点 T_i 对应一个任务，DAG的边 $E_{i,j}=\{(T_i,T_j)|T_i,T_j\in T\}$ 表示不同密码计算任务之间的数据依赖关系集合，值代表传输数据量的大小；D 表示密码计算作业的最晚完成时间；S 表示该密码作业的平均发包速率，本文使用作业包相继到达的时间间隔的分布来表示，实际计算时，使用上一个调度周期的作业包的时间间隔来拟合当前周期作业包的时间间隔分布。组合密码服务作业流如图3所示。此外，本文定义任务 T_i 的前驱节点集和后继节点集分别为

$$\mathrm{PREV}(T_i)=\{T_j|E_{j,i}\in E\} \tag{1}$$

$$\mathrm{SUCC}(T_i)=\{T_j|E_{i,j}\in E\} \tag{2}$$

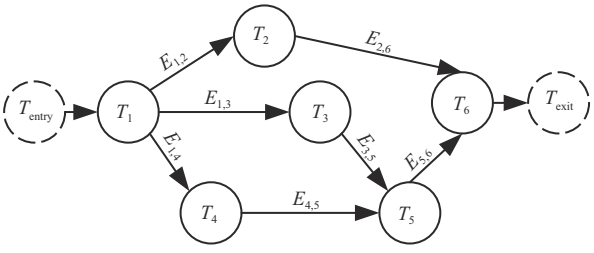

图3　组合密码服务作业流

为方便描述，本文引入伪进入节点 T_{entry} 和伪离开节点 T_{exit}，2 个节点所需的计算任务为 0，通信成本分别为输入、输出数据的带宽大小，且满足 $\text{PREV}(T_{\text{entry}}) = \varnothing$，$\text{SUCC}(T_{\text{exit}}) = \varnothing$。

对于仅包含一个任务 T 的密码计算作业，边表示为 $E = \{E_{\text{entry},T}, E_{T,\text{exit}}\}$，值分别是用户传入计算单元的数据大小及密码计算作业生成的数据大小。

定义 3　密码算法（CA, cryptography algorithm）：指使用特定的参数和密钥来对数据进行加密和解密的算法，以保护数据的机密性、安全性和不可篡改性等，包括对称加解密、非对称加解密、签名验签、哈希和随机数算法等。

定义 4　密码算法知识产权（IP, intellectual property）核：指密码芯片中的 IP 核，下文用算法核表示，其实现了特定的密码学算法并提供了一定算力的逻辑模块，例如加密、哈希等。算法核可以用二元组 $I = (A, C)$ 表示，其中，A 表示该算法核可提供的算法类型，C 表示该算法核可提供的最大算力。

定义 5　密码计算单元：指可以提供密码算法和计算能力的硬件。每个密码计算单元可以提供的密码算法类型和计算能力由其承载的算法核决定。密码计算单元可以用 $U = (B^U, \text{IP}, A, M, D)$ 表示，其中，B^U 表示计算单元的带宽；$\text{IP} = \{I_1, \cdots, I_{N_I}\}$ 表示构成该计算单元的核列表，N_I 表示装配的密码算法核数量；$A = \{A_1, \cdots, A_{N_A}\}$ 表示当前计算单元所提供算法的算力集合，比如，算法 a 的算力 $A_a = \sum_{I \in \text{IP and } I.A = a} I.C$，其中 I 表示算法核集合中的算法核，$I.C$ 表示该算法核的最大算力为 C；M 和 D 分别表示所装配的密码计算组件和密码计算设备。

定义 6　密码计算组件：提供多个密码计算单元接口，对密码计算单元进行统一管理和配置。组件出口带宽用 B^M 表示，装配在密码计算组件中的密码计算单元可表示为 $U = \{U_1, \cdots, U_{N_U}\}$，其中，$N_U$ 表示一个组件上装配的计算单元数量。

定义 7　密码计算设备：指统一组织、监控、管理和配置多个密码计算组件，并提供密码计算作业资源分配与调度功能的计算设备。设备出口带宽用 B^D 表示，部署在密码计算设备中的组件可表示为 $M = \{M_1, \cdots, M_{N_M}\}$，其中，$N_M$ 表示一个设备上装配的计算组件数量。

定义 8　瞬时激增负载适应度（FTSL, fitness of transient surge load）：指密码计算设备对瞬时激增负载的容忍能力，适应度越高，瞬时激增负载对任务完成时间的影响越小。云环境中作业负载通常是不稳定的，因此计算资源可能会因负载激增而导致拥塞，从而延误作业的完成时间。为了应对该问题，本文提出了瞬时激增负载适应度的概念，来评估调度算法对瞬时激增负载的抵抗能力，并用关键任务分配均衡度来描述瞬时激增负载适应度大小。

定义 9　关键任务（CT, critical task）：指从输入到输出经过的时间最长的逻辑路径，在该路径上的任务为关键任务。关键任务由任务的最早开始时间（EST, earlist start time）和最晚开始时间（LST, latest start time）确定，如果任务的最早开始时间和最晚开始时间相等，则该任务为关键任务。本文用 $\text{CT}(T_i^k)$ 表示任务 T_i^k 是否为作业 J_k 的关键任务，如果任务 T_i^k 是作业 J_k 的关键任务，则 $\text{CT}(T_i^k) = 1$，否则等于 0，即

$$\text{CT}(T_i^k) = \begin{cases} 1, & \text{EST}(T_i^k) = \text{LST}(T_i^k) \\ 0, & \text{EST}(T_i^k) \neq \text{LST}(T_i^k) \end{cases} \tag{3}$$

任务的最早开始时间表示该任务最早可以开始执行的时间，由所有前驱任务最早开始时间、前驱任务执行时间（ET, execute time）及前驱任务与当前任务的传输时间（TT, transmission time）决定，表示为

$$\text{EST}(T_i^k) = \max_{T_j^k \in \text{prev}(T_i^k)} \{\text{EST}(T_j^k) + \text{ET}(T_j^k) + \text{TT}(E_{j,i})\} \tag{4}$$

任务最晚开始时间表示在不影响作业整体完成时间的前提下任务最晚可以开始执行的时间，由所有后继任务最晚开始时间、当前任务与后继任务的传输时间以及后继任务等待时间决定，计算式为

$$\text{LST}(T_i^k) = \min_{T_j^k \in \text{succ}(T_i^k)} \{\text{LST}(T_j^k) - \text{ET}(T_j^k) - \text{TT}(E_{i,j})\} \tag{5}$$

执行时间表示在目标运算节点的数据处理速率下完成当前任务所需的时间。对于计算单元 U，任务 T 所需的执行时间为

$$\text{ET}(T, U) = \frac{T.W}{U.A_{T.A}} \tag{6}$$

数据传输时间表示计算作业中相邻计算任务间数据传输的时间。如果 2 个计算任务处在同一个计算单元上，则传输时间为 0；如果在不同计算单元上，则传输时间需要根据部署任务的计算单元间的

位置决定。假设同一CCJ中相邻2个计算任务T_i和T_j分别在计算单元m和n上运行，则传输时间计算方式如下。

1) 当2个任务分配在同一计算单元上，即$m=n$时，传输时间为

$$\mathrm{TT}(E_{i,j},m,n)=0 \tag{7}$$

2) 当2个任务分配在同一计算组件的不同计算单元上，即$m.M=n.M$且$m\neq n$时，传输时间等于计算单元m到计算组件$m.M$以及计算组件$m.M$到另一个计算单元n的时间，即

$$\mathrm{TT}(E_{i,j},m,n)=\frac{E_{i,j}}{\min(m.B^U,m.M.B^M)}+\frac{E_{i,j}}{\min(m.M.B^M,n.B^U)} \tag{8}$$

3) 当2个任务分配在同一计算设备的不同计算组件上，即$m.D=n.D$且$m.M\neq n.M$时，传输时间由计算单元m到计算组件$m.M$的传输时间、2个计算组件$m.M$和$n.M$之间的传输时间以及计算组件$n.M$到计算单元n的传输时间决定，即

$$\mathrm{TT}(E_{i,j},m,n)=\frac{E_{i,j}}{\min(m.B^U,m.M.B^M)}+\frac{E_{i,j}}{\min(m.M.B^M,n.M.B^M)}+\frac{E_{i,j}}{\min(n.M.B^M,n.B^U)} \tag{9}$$

4) 当2个任务分配在不同计算设备上，即$m.D\neq n.D$时，传输时间由计算单元m到计算组件$m.M$的传输时间、计算组件$m.M$到计算设备$m.D$之间的传输时间、2个计算单元$m.D$和$n.D$之间的传输时间、计算设备$n.D$到计算组件$n.M$的传输时间和计算组件$n.M$到计算单元n的传输时间决定，即

$$\mathrm{TT}(E_{i,j},m,n)=\frac{E_{i,j}}{\min(m.B^U,m.M.B^M)}+\frac{E_{i,j}}{\min(m.M.B^M,m.D.B^D)}+\frac{E_{i,j}}{\min(m.D.B^D,n.D.B^D)}+\frac{E_{i,j}}{\min(n.D.B^D,n.M.B^M)}+\frac{E_{i,j}}{\min(n.M.B^M,n.B^U)} \tag{10}$$

定义10 关键任务均衡度（CTB, critical task balance）：指在当前调度策略下关键任务分配的均衡程度。在分配计算资源时，考虑将关键任务在多个计算单元上交叉分配，优先执行计算单元上的关键任务，而非关键任务只要在其最晚开始时间前执行就不会影响作业的最终完成时间。

本文用关键任务方差表示关键任务均衡程度，即

$$\mathrm{CTB}(J,U,\Omega)=\sum_{u\in U}\left(\sum_{J_i\in JT_{i,j}}\sum_{J_i.G.T}\Omega_{i,j}^{u}\mathrm{CT}(T_{i,j})-\overline{\mathrm{CT}}\right)^2,$$

$$\overline{\mathrm{CT}}=\frac{\sum_{u\in U}\sum_{J_i\in JT_{i,j}}\sum_{J_i.G.T}\Omega_{i,j}^{u}\mathrm{CT}(T_{i,j})}{|U|} \tag{11}$$

其中，$J=\{J_1,\cdots,J_{N_J}\}$表示所有的密码计算作业，$U=\{1,\cdots,N_U\}$表示所有的密码计算单元，Ω为调度策略，其元素$\Omega_{i,j}^{u}\in\{0,1\}$表示任务$T_{i,j}$是否调度给计算单元$u$。

定义11 期望等待时间（EWT, expected waiting time）：指在调度过程中预计每个任务需要等待的时间。为方便描述，假设算法核算力为C，分配到该计算单元的任务集合为$T^Q=\{T_1^Q,\cdots,T_{N_Q}^Q\}$，为估算每个任务的等待时间，本文将每个计算单元的服务建模为一个排队论模型，每个算法核可以同时处理一个请求，该请求在算法核上进行服务的时间服从指数分布。基于上述假设，有N个相同算法类型算法核的计算单元模型可以用肯德尔记号表示为$M/M/N$。其中，M表示指数分布，第一个M代表泊松请求到达过程，第二个M代表指数分布的服务时间，N表示服务台的数量。服务速率$\mu=\frac{|T^Q|}{\sum_{i=1}^{N_Q}\frac{T_i^Q.W}{C}}$表示单位时间内能够提供服务的平均速率，其中，$T_i^Q.W$表示任务$T_i$的计算量。到达速率$\lambda$表示单位时间内平均到达的请求数量，可以根据服务请求中申请的值估算。假设作业T_j对应的服务请求的到达速率为λ_j，因为指数分布拥有无记忆性，所以计算单元的到达速率等于所有分配到该计算单元上的所有任务的速率之和，即$\lambda=\sum_{t=1}^{N_Q}\lambda_t$。

如果一个计算单元中只有一个相同算法类型的算法核，则将该计算单元相关算法对应的队列建模为一个$M/M/1$排队系统，单算法核的期望等待时间为

$$\mathrm{EWT}=W-\mathrm{E}[S]=W-\frac{1}{\mu}=\frac{\lambda}{\mu(\mu-\lambda)} \tag{12}$$

其中，E[S]表示服务时间的期望。

如果一个计算单元中有k个相同算法类型的算法核，则将该算法建模为一个$M/M/k$排队系统，由厄兰（Erlang）等待式可知，对于多服务台等待的排队系统，由已得到的平稳分布可得平均排队长度

L_q为

$$L_q = \frac{c(k,\rho)\rho_k}{1-\rho_k} \tag{13}$$

$$\rho = \frac{\lambda}{\mu} \tag{14}$$

$$\rho_k = \frac{\rho}{k} = \frac{\lambda}{k\mu} \tag{15}$$

其中，$c(k,\rho)$是指当系统中有k个服务台时，单位时间内有k个顾客同时到达系统并排队的概率。通常，$c(k,\rho)$是通过齐次马尔可夫链的稳态概率计算的，即

$$c(k,\rho) = \sum_{n=k}^{\infty} P_n = \frac{\rho^k}{k!(1-\rho_k)} P_0 \tag{16}$$

$$P_0 = \left[\sum_{n=0}^{k-1}\frac{\rho^n}{k!} + \frac{\rho^n}{k!(1-\rho_k)}\right]^{-1} \tag{17}$$

因此，将式(13)~式(17)代入利特尔式，对于拥有k个相同算法类型的算法核的排队系统，期望平均等待时间为

$$\mathrm{EWT}_k = \frac{L_q}{\lambda} = \frac{\rho^k\rho_k}{\lambda k!(1-\rho_k)^2\sum_{n=0}^{k-1}\frac{\rho^n}{k!} + \frac{\rho^k}{k!(1-\rho_k)}} \tag{18}$$

式(18)中共有4个变量：k，λ，ρ，ρ_k，其中，k为队列数量，λ为到达速率，$\rho = \frac{\lambda}{\mu}$和$\rho_k = \frac{\rho}{k} = \frac{\lambda}{k\mu}$可以由到达速率$\lambda$、服务速率$\mu$和队列数量$k$计算，而到达速率和服务速率可以根据上一个调度阶段计算。因此，可以根据式(12)和式(18)计算出期望等待时间。

2.2　问题定义

给定一组组合密码计算作业$J=\{J_1,\cdots,J_{N_J}\}$和一组密码计算单元$U=\{1,\cdots,N_U\}$，生成一个调度策略Ω，其元素$\Omega_{i,j}^u \in \{0,1\}$表示任务$T_{i,j}$是否调度给计算单元$u$。在满足任务完成时间要求的前提下，实现能耗最低、迁移成本最低和瞬时激增负载适应度最高的优化目标。

目标函数 P1　能耗。在本文中，计算单元是由现场可编程门阵列（FPGA）实现的密码计算硬件，作为一种互补金属氧化物半导体（CMOS）工艺实现的电子设备，FPGA在一定速度下的主导功耗由有效负载电容k_{ef}、电源电压和速度S_j决定[24]。根据文献[25]，计算单元的能耗函数为

$$P_j^{\mathrm{total}} = P_j^s + P_j^d$$
$$P_j^d = \xi\left(S_j\right)^2 f \tag{19}$$

由式(19)可知，计算单元的能耗由静态能耗P_j^s和动态能耗P_j^d两部分组成，静态能耗是由晶体管内部的漏电电流引起的，只与温度和该节点的运行时间有关，动态能耗是由负载电容的充放电以及晶体管开关时的短路电流的开关活动引起的；$\xi = \frac{k_{\mathrm{ef}}}{\left(k_h\right)^2}$由计算单元的有效负载电容$k_{\mathrm{ef}}$和比例常数$k_h$确定，$S_j$是设备运行速度，受电压大小影响，参数$\xi$和$S_j$均由设备本身参数确定。因此，每个计算单元的动态能耗大小只和计算任务所需要的时钟数量f有关。本文用$\mathrm{CNT}^k(C)$表示算法类型k计算C bit的数据所需要的时钟数量，并用$a_j = \xi\left(S_j^k\right)^2$表示计算单元相关的特定参数。给定运行在计算单元$j$上的任务$T$，则动态能耗函数可通过密码运算数据量计算，如式(20)所示。

$$\mathrm{EC}_j^d(T) = a_j\mathrm{CNT}^{T.A}(T.C) \tag{20}$$

最小化能耗的目标函数可表示为

$$\min \mathrm{EC}(J,U,\Omega) = \left(\sum_{i=1}^{N_J}\sum_{T_{i,j}\in J_i.G.T}\Omega_{i,j}^u\mathrm{EC}_u^d(T_{i,j}) + P_u^s t_j\right) \tag{21}$$

其中，P_u^s是j单位时间内计算单元的静态能耗，t_j表示计算单元处于运行状态的时长，第一个累加符号表示对所有计算单元的能耗进行遍历求和，第二个累加符号表示对所有的计算任务进行遍历求和，第三个累加符号表示对所有计算作业i中运行在计算单元u上的任务的能耗进行遍历求和，$J_i.G.T$表示计算作业J_i中所有的计算作业集合。当任务$T_{i,j}$分配在计算单元u上，则$\Omega_{i,j}^u$为1，计算得到的能耗值累加在总能耗上；否则$\Omega_{i,j}^u$为0，对总能耗的影响为0。在特殊情况下，如果所有计算单元同构，则密码计算任务运行在任何计算单元上的动态能耗都相等，上述优化目标等价于求解处于运行状态中的计算单元数量最少的调度策略。

目标函数 P2　迁移成本。在执行调度算法时，需要考虑迁移成本，不同数据类型的迁移成本不同。根据“明态密钥不出设备”的安全性需求，密钥需要保存在分配给计算任务的设备中，因此，在将任务从一个计算单元迁移到另一个计算单元时，需要将相关的所有密钥安全地迁移，带来的成本为密钥迁移成本（KMC, key migration cost）。配置新计算单元环境所需要的成本叫作上下文迁移成本（CMC, context migration cost）。

迁移成本由密钥迁移成本和上下文迁移成本决定。对于密码计算任务T，从计算单元m迁移到计算单元n的迁移成本为

$$\mathrm{MC}(T,m,n)=p_{\mathrm{kmc}}\mathrm{KMC}(T,m,n)+(1-p_{\mathrm{kmc}})\mathrm{CMC}(T,m,n) \tag{22}$$

其中，$p_{\mathrm{kmc}}\in[0,1]$是密钥迁移成本权重。

密钥迁移成本包括密钥传输时间和密钥空间处理和同步时间$t_{\mathrm{pre}}^{\mathrm{key}}$，即

$$\mathrm{KMC}(T,m,n)=\begin{cases}\mathrm{TT}(T,K,m,n)+t_{\mathrm{pre}}^{\mathrm{key}}, & m\neq n\\ 0, & m=n\end{cases} \tag{23}$$

在本文中，上下文迁移成本用固定值$t_{\mathrm{pre}}^{\mathrm{cfg}}$表示配置和执行上下文迁移带来的成本，即

$$\mathrm{CMC}(T,m,n)=\begin{cases}t_{\mathrm{pre}}^{\mathrm{cfg}}, & m\neq n\\ 0, & m=n\end{cases} \tag{24}$$

本文用Ω'表示上一个调度区间内的分配策略，则迁移成本目标函数可表示为

$$\min\mathrm{MC}(J,\Omega,\Omega')=\sum_{J_i\in J}\sum_{T_j^i\in J_i.G.T}\mathrm{MC}(T_{i,j},p,q),\quad \Omega_{i,j}^{p}=1,\Omega'^{p}_{i,j}=1 \tag{25}$$

目标函数P3　瞬时激增负载适应度。本文用关键任务均衡程度来评估瞬时激增负载适应度，用$\mathrm{CB}_{\max}$表示最大的CTB取值，本文引入函数$\mathrm{FTSL}(J,U,\Omega)$来表示瞬时激增负载适应度，即

$$\max\mathrm{FTSL}(J,U,\Omega)=\min_{\Omega}\{\mathrm{CB}_{\max}-\mathrm{CTB}(J,U,\Omega)\} \tag{26}$$

关键任务负载均衡度CTB计算式如式(11)所示。CTB取值越小，表示关键任务负载越均衡，则瞬时激增负载适应度越高，即CTB值越小瞬时激增负载适应度越高，因此函数$\mathrm{FTSL}(J,U,\Omega)$取值越大则调度策略$\Omega$的瞬时激增负载适应度越高。

约束C1　作业完成时间。作业完成时间由每个任务的等待时间（WT, wait time）、ET和TT决定。因此，作业J的作业完成时间（MS, makespan）计算式为

$$\mathrm{MS}(J)=\sum_{T_i\in J.G.T}(\mathrm{WT}(T_i)+\mathrm{ET}(T_i))+\sum_{E_i\in J.G.E}\mathrm{TT}(E_i) \tag{27}$$

等待时间表示等待队列中当前任务之前所有任务的总执行时间。传统调度算法中使用平均执行时间的方式也不适用于密码计算单元中算法类型多、算力差异大的场景，因此，本文引入排队论的思想来更加准确地计算任务的期望完成时间

$$\mathrm{MS}(J)=\sum_{t\in J.G.T}(\mathrm{EWT}(t)+\mathrm{ET}(t))+\sum_{e\in J.G.E}\mathrm{TT}(e) \tag{28}$$

综上所述，任务在约定时间内完成的约束可表示为

$$\forall J_i\in J,\mathrm{MS}(J_i)\leqslant J_i.D \tag{29}$$

约束C2　同一个任务只能分配到一个计算单元上，即

$$\forall J_i\in J,T_{i,j}\in J_i.G.T,\sum_{u\in U}\Omega_{i,j}^{u}=1 \tag{30}$$

约束C3　算力约束。分配到一个计算单元上的任务算力总和不能超过所提供的最大算力

$$\forall u\in U,\forall a,\sum_{J_i\in J}\sum_{T_{i,j}\in J_i.G.T,J_j.A=A}T_{i,j}.C\,\Omega_{i,j}^{u}\leqslant u.A_a \tag{31}$$

综上所述，组合密码服务任务调度问题可描述为

$$\begin{aligned}&\mathrm{P1}:\min\mathrm{EC}(J,U,\Omega)\\&\mathrm{P2}:\min\mathrm{MC}(J,U,\Omega,\Omega')\\&\mathrm{P3}:\max\mathrm{FTSL}(J,U,\Omega)\\&\mathrm{C1}:\forall J_i\in J,\mathrm{MS}(J_i)\leqslant J_i.D\\&\mathrm{C2}:\forall J_i\in J,T_j\in J_i.G.T,\sum_{u\in U}\Omega_{i,j}^{u}=1\\&\mathrm{C3}:\forall u\in U,\forall a,\sum_{J_i\in J}\sum_{T_{i,j}\in J_i.G.T,j.A=a}T_{i,j}.C\Omega_{i,j}^{u}\leqslant u.S_a\end{aligned} \tag{32}$$

其中，P1为第一个优化目标，即最小化能耗，计算方法和解释如式(21)所示；P2为第二个优化目标，即最小化迁移成本，计算方法如式(25)所示；P3为第三个优化目标，即最大化瞬时激增负载的适应度，计算方法和解释如式(26)所示；约束C1表示所有密码计算作业都应在约定时间内完成，计算方法如式(27)所示；约束C2表示一个密码任务只能分配给一个计算单元执行，计算方法和解释如式(30)所示；约束C3表示分配给计算单元的所有任务的算力和不能超过计算单元相关算法的最大算力，计算方法和解释如式(31)所示。

3　算法设计

解决上述多目标优化问题的关键挑战是求解整数变量$\Omega_{i,j}^{u}\in\{0,1\}$的值，本文采用两阶段的调度算法DRFF-NSGA-III-PS，在计算单元选择阶段，设计了基于改进的DRFF-NSGA-III算法生成计算单元选择算法，为每个密码计算任务选择合适的计算单元；在密码计算任务排队阶段，设计了任务紧迫程度的优先级排序算法，对密码计算任务排序，以提高执行效率。

3.1　基于 DRFF-NSGA-III 的计算单元选择算法

本文提出的 DRFF-NSGA-III 算法是非支配占优排序遗传学算法的改进算法[26]。NSGA-III 在 NSGA-II 的基础上引入了参考点的概念，在非支配解的拥挤度计算上进行了优化，并通过改进排序机制提供了更多的多样性及前沿覆盖，因此更适用于大规模多目标问题。然而 NSGA-III 在大规模问题中的收敛速度仍然较慢，因此，本文引入了 DRFF 算法用来创建初始种群，加快收敛速率。DRFF-NSGA-III 算法的关键步骤如下。

1) 编码方式。DRFF-NSGA-III 的编码方式与传统遗传算法一样，需要将问题的解编码为一个染色体。本文采用整数编码的方式，因为求解该问题时，整数编码更加便捷。本文将一个潜在的任务调度策略编码为一个染色体信息，对于第 i 个染色体 I_i，首先将所有作业按顺序排列，然后对作业内的所有任务按顺序排列，排序后的任务列表组成一个染色体 $I_i=[g_1^i,\cdots,g_{N_g}^i]$，其中，$g_j^i\in[1,N_U]$ 为染色体中基因的值，表示第 j 个任务分配到的计算单元索引，如图 4 所示；N_g 为染色体大小，等于总任务个数，即

$$N_g=\sum_{i=0}^{N_J}|J_i.G.T| \tag{33}$$

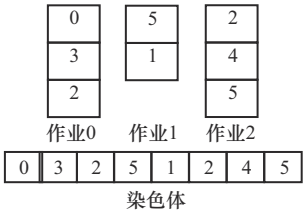

图 4　编码方式

多目标优化的解编码为染色体的规则为：如果 $\Omega_{i,j}^u=1$，则 $I_{\sum_{k<i}|J_k.G.T|+j}=u$。根据约束 C2 可知，一个任务只能分配到一个计算节点，因此每个基因只有一个值。

染色体解码的规则为：若 $I_k=u$，则 $\Omega_{i,j}^u=1$，$i=\max_i\{\sum_{x<i}|J_x.G.T|<k\}$，$j=k-\sum_{x<i}|J_x.G.T|$。

2) 初始化种群。在产生第一代种群时，本文提出的 DRFF 算法引入了更多的随机性，有助于更好地探索解空间，提高算法的多样性，同时，DRFF 算法可以产生满足一定条件的初始解，提高初始种群的质量，加快收敛的速度，如算法 1 所示。

算法 1　DRFF 算法

输入　染色体长度 N_g，按顺序排列的任务序列 $T=\{T_1,\cdots,T_{N_T}\}$ 和计算单元序列 $U=\{U_1,\cdots,U_{N_U}\}$

输出　第一代种群染色体 $I=[g_1,\cdots,g_{N_g}]$

① 初始化：计算基因索引列表 $\mathrm{IDX}_g=[1,\cdots,N_g]$，计算单元索引列表 $\mathrm{IDX}_U=[1,\cdots,N_U]$，计算单元不同算法的已用算力 $V=[V_{0,0},\cdots,V_{N_U,N_A}]$，$V_{i,j}=0$。

② shuffle(IDX_g)

③　shuffle(IDX_U)

④　for i in IDX_g do

⑤　　for u in IDX_U do

⑥　　if $V_{u,T_i.A}+T_i.C\leqslant U_u.A_{T_i.A}$ then

⑦　　$g_i=n$

⑧　　$V_{u,T_i.A}=V_{u,T_i.A}+T_i.A$

⑨　　break

⑩　　end if

⑪　end for

⑫　end for

算法 1 首先打乱任务索引和计算单元索引，如步骤①~步骤②所示；然后按照打乱后的顺序为每个任务遍历可用的计算单元，如步骤④~步骤⑫所示，如果当前遍历到的首个计算单元可用算力可以满足当前计算任务的算力需求，则将该计算单元索引值赋给染色体中对应计算任务索引的基因，并更新该计算单元已使用的算力大小，如步骤⑥~步骤⑨所示。

3) 交叉算子。随机选择 $2N_{\mathrm{off}}$ 个染色体，两两进行交叉操作，本文采用两点交叉算法进行交叉运算，生成 2 个新染色体，目的在于探索更换不同计算任务间的计算单元对提高解效果的影响，如图 5 所示。

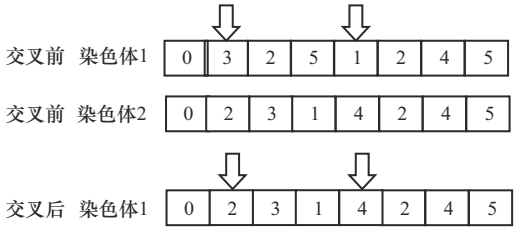

图 5　交叉算子

4) 变异算子。为了提高算法的探索能力，避免陷入局部最优解，本文采用倒置变异算法，将选择的2个位置之间的元素进行倒置，如图6所示。

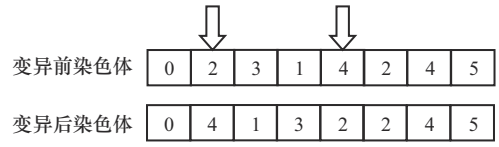

图6 变异算子

5) 评估个体适应度。将子代和父代个体整合为一个群体，并计算所有个体的适应度。对于每个染色体I计算适应度，即根据每个染色体计算目标函数P1~P3的值。不同目标函数往往具有不同的量纲和量纲单位，影响收敛速度，因此，本文对目标函数的取值进行归一化操作，采用最大最小标准化操作

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)} \tag{34}$$

对于目标函数P1和P2，计算最小值复杂度较高，因此最小值都设置为0，同理，目标函数P3最小值设置为0。目标函数P1最大能耗值EC_{max}按照所有设备、模块和计算单元满负载运行时能耗取值，目标函数P2最大迁移成本MC_{max}按照所有密钥迁移距离最远带来的迁移成本取值，目标函数P3瞬时激增负载适应度按照所有关键路径分配到同一个计算单元上的均衡度取最大值FC_{max}。

6) 非支配排序。对种群进行非支配排序，可以得到不同的帕累托层级。一个解I_i被另一个解I_j所支配，意味着I_j在至少一个目标函数上不劣于另一个解I_i，且在至少一个目标函数上更优，本文用$I_j \prec I_i$表示解I_j对解I_i的支配关系。非支配排序算法如算法2所示。

算法2 非支配排序算法

输入 解集合$I = \{I_1,\cdots,I_{N_I}\}$

输出 分层后的解集合$L = \{L_1,\cdots,L_{N_L}\}$

① 初始化：非支配层数$k = 1$，每个解I_i被支配的数量$S_i = 0$，每个解I_i所支配的解集合$P_i = \varnothing$，下一层非支配解集合$L' = \varnothing$，分层的解集合序列$L = \varnothing$。

② for each I_i in I do

③ for each I_j in $I - I_i$ do

④ if $I_i \prec I_j$ then

⑤ $P_i = P_i \cup I_j$

⑥ else if $I_j \prec I_i$ then

⑦ $S_i = S_i + 1$

⑧ end if

⑨ if $S_i == 0$ then

⑩ $F = F \cup I_i$

⑪ end if

⑫ end for

⑬ end for

⑭ $L_k = F$

⑮ while $F \neq \varnothing$ then

⑯ for each I_p in F do

⑰ for each I_q in P_p

⑱ $S_q = S_q - 1$

⑲ if $S_q == 0$ then

⑳ $L' = L' \cup I_q$

㉑ end if

㉒ end for

㉓ end for

㉔ $F = L'$

㉕ $k=k+1$

㉖ $L_k = F$

㉗ $L = L \cup L_k$

㉘ end while

算法2首先计算每个解之间的支配关系，并记录被支配的次数，如步骤①~步骤⑧所示。如果一个解没有被其他解支配，则该解为帕累托前沿上的解，如步骤⑨~步骤⑪所示。最后根据解的支配关系分为多层，如步骤⑮~步骤⑱所示。

7) 计算拥挤度。对于每一层解集合L_k，计算个体之间的拥挤度距离，拥挤度距离用于衡量个体在目标空间中的分布密度，以促进种群的多样性，防止种群收敛到某个局部区域。最后根据权重向量和参考点计算出个体的密度估计。本文采用Das-Dennis[27]算法计算参考点。

8) 选择算子。根据非支配排序和拥挤度距离，按照支配顺序从高到低选择非支配排序集合进入下一代，对于无法全部入选的非支配排序集合，根据拥挤度选择个体进入下一代。选择过程应确保在整个帕累托前沿上均匀分布的个体被保留。

重复执行上述步骤，直到满足停止条件，例如

达到最大迭代次数或达到足够收敛程度。返回帕累托前沿上的非支配解集合，这些解集合在目标空间中相互之间没有明显的优劣关系。最后根据一定的选择标准，选择其中一个个体作为问题的最终解。本文采用伪权重算法选择个体。

当获得了帕累托前沿，即一组互相无法支配的调度策略后，本文需要根据需求从众多策略中选择一个作为本文的调度策略。多目标优化问题的这种决策过程也称为多标准决策（MCDM, multi-criteria decision making）。本文采用伪权重算法选择解，首先计算每个候选解的伪权重值，然后找到与所需要的权重值 w 最接近的伪权重值对应的解，计算式为

$$\mathrm{pw}_i = \frac{\dfrac{f_i^{\max} - f_i(x)}{f_i^{\max} - f_i^{\min}}}{\displaystyle\sum_{m=1}^{M} \frac{f_m^{\max} - f_m(x)}{f_m^{\max} - f_m^{\min}}}$$

$$i = \arg\min\left(\sum_{m=1}^{M} |\mathrm{pw}_i^m - w^m| \right) \tag{35}$$

其中，M 是目标函数的个数，pw_i 是计算得的伪权重值，w 是目标权重值，pw_i^m 和 w^m 分别是第 i 个解的伪权重值和权重值在目标函数 m 上的取值。

3.2　基于任务紧迫程度的优先级排序算法

计算单元上的待处理任务队列 Q 通常包含多个计算作业的计算任务等待执行，为了保障所有计算作业都能尽快完成，需要优先计算关键任务，同时还需要考虑任务的紧迫程度，如果某任务及其后续任务的剩余可用时间比较紧迫，则需要提高该任务的优先级。本文采用加权方式，综合考虑关键任务及剩余处理时间，以确定任务优先级

$$\mathrm{PRI}(T_{i,j}) = p_{\mathrm{ct}}\mathrm{CT}(T_{i,j}) + p_{\mathrm{ur}}\frac{\mathrm{MS}(J_i) - \mathrm{EST}(T_{i,j})}{J_i.D - t}$$

$$p_{\mathrm{ct}} + p_{\mathrm{ur}} = 1 \tag{36}$$

其中，p_{ct} 为关键任务的权重系数，p_{ur} 是紧迫程度权重系数。式(36)前半部分表示是否为关键任务；后半部分表示完成剩下任务需要的时间和剩余执行时间的比例，比值越大说明可用的时间越少，优先级越高，当比值大于 1 时，表示完成剩下任务的预计时间比剩余可用时间要长，需要更早完成，优先级更高，$\mathrm{MS}(J_i)$ 表示任务 i 的总完成时间，计算方法如式(20)所示，$\mathrm{EST}(T_{i,j})$ 为任务 $T_{i,j}$ 的最早开始时间，计算方法如式(4)所示。等待队列的重排序算法如算法 3 所示。

算法 3　等待队列的重排序算法

输入 待执行的密码计算任务序列 Q，计算单元支持的算法个数 N_A，计算单元的算法核集合 IP

输出 不同算法核的任务执行序列 $Q'_1,\cdots,Q'_{|\mathrm{IP}|}$

1) for i in 1 : N_A do
2)　$C_i = \varnothing$
3) end for
4) for each T in Q do
5)　T.pri=PRI(T)
6) end for
7) sort_by_PRI(Q)
8) for each T in Q do
9)　$C_{T.A} = C_{T.A} \cup T$
10) end for
11) for a in 1 : N_A do
12)　for T in C_a do
13)　　I=min{I.ready_time|I in IP, $I.a$=a}
14)　　$Q'_1 = Q'_1 \cup T$
15)　　I.ready_time = I.ready_time + ET(T)
16)　end for
17) end for

算法 3 首先对不同密码算法类型的任务队列初始化为空，如步骤 1)~步骤 3)所示。其次根据式(36)计算每个任务的优先级并根据优先级排序，如步骤 4)~步骤 7)所示。再次根据不同算法类型，将任务添加到指定的队列中，如步骤 8)~步骤 10)所示。最后，根据算法核最早可使用时间 I.ready_time 为任务分配算法核，并根据式(6)计算任务执行时间并更新该算法核的最早可用时间，如步骤 11)~步骤 17)所示。

在正常情况下，数据包按照到达顺序执行，当计算单元的待处理队列长度超过阈值时才被触发。需要注意的是，排队论模型中假设作业包按照顺序到达和处理，没有优先级区别。接下来，分析在本文调度场景中引入排序对预测准确性的影响。对于非关键任务，只要在最晚开始时间前启动，就不会影响整体完成时间，因此，本文在这里只分析排序对关键任务的影响。在正常情况下，排队论模型基于每个作业流平均发包速率预测完成时间，每个任

务按序到达正常执行；当负载突然增加时，按照原本到达速率计算的结果已经失准，无法预测正常的等待时间，因此即便调整数据包处理顺序也没有影响，反而通过计算优先级，先执行关键任务和即将超时的任务会提高作业按时完成的可能性。因此，通过排队论模型预测正常负载下的等待时间，通过优先级算法处理负载激增情况下数据包处理顺序的方式是可行的。

4　仿真分析

4.1　场景和参数设置

密码计算设备集群调度仿真场景模拟了基于高性能密码机密码计算设备集群的任务调度，一台高性能 ATCA（advanced telecom computing architecture）密码机就是一个 CCD，CCD 配置参考恒光 6U6SAC 机箱配置，该机箱提供一个交换槽位和 5 个业务槽位，装配一块时代通信 AS100 交换板和 4 块 ATCA1400 业务板，每块业务板中装配 8 片型号为 Xilinx XC7K410T 的 FPGA 加密卡。因此在模拟环境中本文配置了一台 CCD，并且该 CCD 提供一个对外的数据通路和 4 个对内的 CCM 之间的数据通路，所有用户数据均通过该通路与 CCD 完成交互。每个 CCM 也和自身的 8 个 CCU 相连接，构成密码计算设备集群。计算服务系统和加密存储系统等抽象为密码计算设备集群的用户。仿真实验在 OMNet++平台上进行，仿真拓扑如图 7 所示。

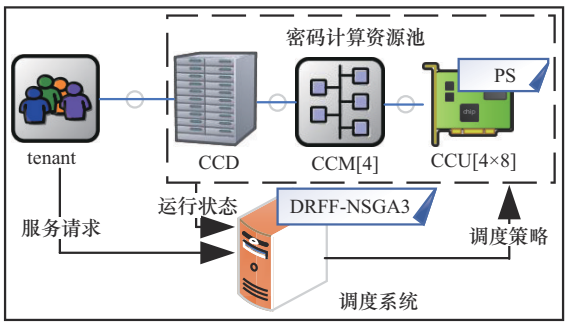

图 7　OMNet++仿真拓扑

在仿真实验中，用 tenant 来仿真所有的云密码服务租户。tenant 和密码计算设备之间存在数据通路，并通过密码计算设备和密码计算组件的转发完成与密码计算单元之间的通信。系统配置及仿真参数如表 1 所示。调度策略生成算法部署在图 1 中的调度系统中，运行仿真环境的实验平台配置为 Intel (R) Core(TM) i7-6700，16 GB RAM；调度系统的运行环境配置为 Intel(R) Xeon(R) Gold 6242，128 GB RAM，Python3.10.12；多目标优化框架采用 pymoo0.6。

表 1　　系统配置及仿真参数

仿真参数	取值
密码机 CCD 带宽 B^D/(Gbit·s^{-1})	100
业务板 CCM 带宽 B^M/(Gbit·s^{-1})	80
计算单元 CCU 带宽 B^U/(Gbit·s^{-1})	10
密码算法个数 N_A	3 (SM2、SM3、SM4)
CCD 装配 CCM 数量 N_M	4
CCM 装配 CCU 数量 N_U	8
CCU 配置的算法核数量 N_i	基于 Xilinx XC7K410T 和 325T 资源配置
单个 SM2 算法核算力	基于 Xilinx XC7K410T 和 325T 算力配置
单个 SM3 算法核算力	基于 Xilinx XC7K410T 和 325T 算力配置
单个 SM4 算法核算力	基于 Xilinx XC7K410T 和 325T 算力配置
算法核的算法类型	SM2、SM3、SM4
CCD 静态能耗 P^s_D	参考恒光 6USAC 机箱，时代通信 AS100 交换板能耗
CCM 静态能耗 P^s_M	参考 ATCA1400 业务板能耗
CCU 静态能耗 P^s_U	参考 Xilinx XC7K410T 和 325T 静态能耗
CCU 动态能耗 P^d_U	参考 Xilinx XC7K410T 和 325T 动态能耗

4.2　作业流生成方法

采用标准 P-method[28] 生成不同形状的随机任务图，以评估本文所提算法在不同类型任务图上的性能。此外，任务的作业大小和输出大小是随机的。本文实验测试中共随机生成 50 个密码计算作业，每个作业包含 3~10 个计算任务。

4.3　不同算法性能分析

4.3.1　DRFF-NSGA-III 算法收敛性

图 8 给出了 DRFF-NSGA-III 算法收敛效果。本文采用超体积（HV, hyper volume）法评估算法收敛性[29]，一个解集越优，HV 指标越大。从图 8 可以看出，本文算法在 35 000 次左右达到收敛。因此可以证明，本文提出的优化目标和调度算法是可以收敛的。

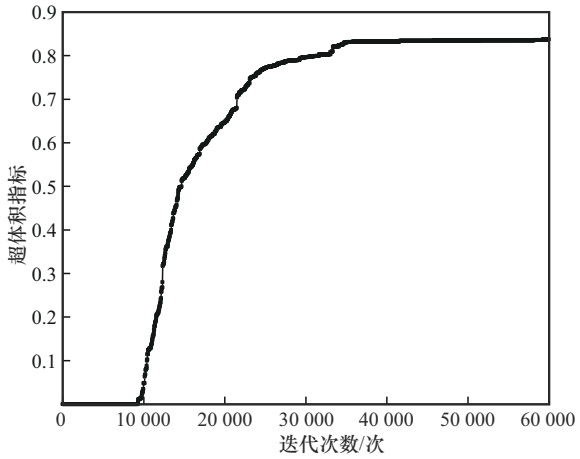

图 8　DRFF-NSGA-III 算法收敛效果

4.3.2　不同权重下调度策略对比

本文共有能耗、迁移成本和瞬时激增负载适应度 3 个优化目标，为了确定伪权重算法[30]在所提调度策略决策算法中的参数，图 9 对比了当迁移成本权重系数分别为 0.3、0.5、0.7 时，不同能耗权重系数对应的 3 个优化目标的变化情况，这里对优化目标进行了标准化处理。当能耗权重系数不断增加时，所选择的解也更加倾向于降低能耗，因此，能耗呈现出降低的趋势，而随着能耗权重系数的增加，迁移成本和瞬时激增负载适应度分别呈现升高和降低的趋势。当能耗权重系数不变时，本文对比迁移成本权重系数变化，以能耗权重系数为 0.15 为例，随着迁移成本权重系数增加，能耗和瞬时激增负载适应度降低。实验结果证明，伪权重算法可以有效选择非支配解，生成合理和所需要的调度方案。本文实验中兼顾能耗、迁移成本和瞬时激增负载适应度 3 个指标，将权重系数分别设置为 0.4、0.3、0.3。

4.3.3　有效性分析

为了验证本文提出的 DRFF-NSGA-III-PS 算法的有效性，本节将 DRFF-NSGA-III-PS 算法与以能耗最低为优化目标的遗传调度算法（EGA, energy-oriented genetic algorithm）[16]、First-Fit 算法[31]、随机调度（Rand-Robin）算法[32]和轮询（Polling）算法[33]进行对比，分别测试不同调度算法的能耗、迁移成本以及瞬时激增负载适应度 3 个指标。其中，EGA 适用于以降低能耗为目标的优化调度场景，考虑了能耗最低的要素；First-Fit 算法的适用场景是资源在线分配场景，以贪心的思想快速分配计算资源，分配完成后保持分配策略不变，不考虑其他优化目标；Rand-Robin 和 Polling 考虑了通过负载均衡保证服务质量的调度要素。本文算法针对密码服务调度场景，考虑能耗、迁移成本和瞬时激增负载适应度等要素，较其他 4 种算法考虑要素更全面，更适用于密码服务的高效调度。为了更加清晰地体现不同算法在 3 个优化目标下的对比效果，本节对优化目标进行标准化处理，如图 10 所示。

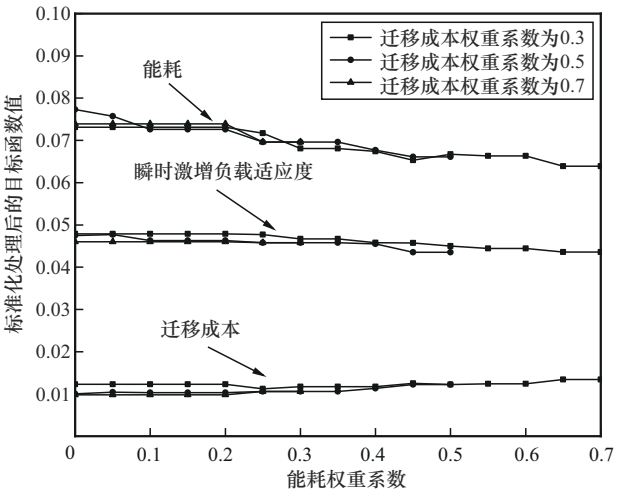

图 9　不同能耗权重系数对应的 3 个优化目标的变化情况

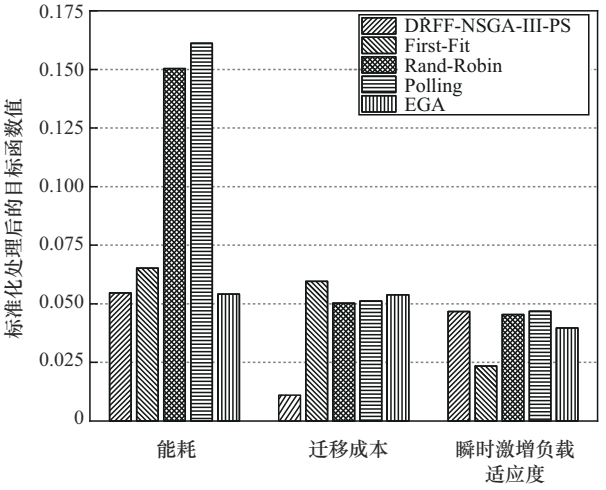

图 10　不同算法在 3 个优化目标下的标准化处理后的目标函数值

图 10 结果表明，在能耗方面，本文算法相比 First-Fit 算法降低了约 16%；在迁移成本方面，只有本文算法考虑到了上一个调度周期的作业分配情况，因此和其他调度算法相比有极大的优势；在瞬时激增负载适应度方面，本文算法与 Polling 算法相近，因为 Polling 算法更加均衡地将关键任务分配给不同的计算单元，因此抗瞬时激增负载能力更

强。实验结果表明，本文算法在兼顾能耗和迁移成本的同时，使得抗瞬时激增负载能力与Polling算法相近。

4.3.4　抗瞬时激增负载能力

为了验证本文算法的瞬时激增负载适应度指标的有效性，本节随机增加作业的发包速率，并不断提升负载激增作业的比例，查看当负载激增时不同调度算法的抗瞬时激增负载能力。与原始作业相比，负载激增作业的发包速率提高200倍。本节比较了不同调度算法在不同瞬时激增作业比例下的作业按时完成情况，结果如图11所示。

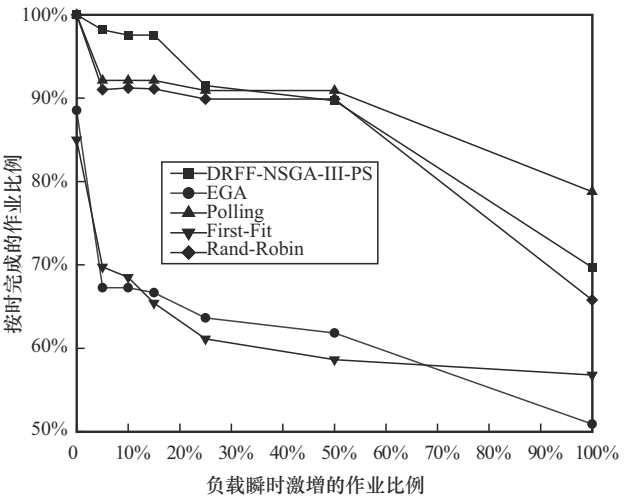

图11　不同调度算法在不同瞬时激增作业比例下的作业按时完成情况

图11结果表明，本文算法和Polling算法对瞬时激增负载的适应能力相对较高，且远高于不考虑瞬时激增情况的First-Fit和EGA，这与瞬时激增适应度评估结果基本一致，验证了该指标在体现调度算法抗瞬时激增负载方面的有效性。Polling算法不考虑能耗以及密码计算作业之间的依赖关系，将密码计算任务分散地交给不同的密码计算单元执行，保证了多个关键任务在分配时更加均匀地运行在不同的密码计算单元上，当负载激增时，密码计算单元优先执行关键任务，因此其抗瞬时激增负载能力更强。实验结果表明，本文算法和Polling算法的抗瞬时激增负载能力较强，随着负载瞬时激增作业比例的增加，Polling算法所体现出来的能力更加显著，然而Polling算法没有考虑能耗的因素，以牺牲能耗的方式保证了抗瞬时激增负载能力，而本文算法兼顾了能耗和抗瞬时激增负载能力。

5　结束语

针对云环境下多密码算法协同计算、多密码任务随机交叉、用户数量多、并发需求高和作业负载瞬时激增等特点，本文提出了“选择–排序”两阶段的多密码任务协同计算的服务调度算法，在保证服务质量的前提下最小化密码计算设备集群的能耗。在选择计算单元阶段，以能耗、迁移成本和瞬时激增负载适应度为优化目标，将协同任务的调度问题建模为多目标优化问题，并采用改进的NSGA-III算法求解。在执行任务排序阶段，根据任务的紧迫程度排序，保证关键任务和紧急任务优先执行。仿真实验表明，本文算法可以收敛到一个相对较好的调度策略，同时在能耗、迁移成本和瞬时激增负载适应度等方面也优于其他算法。本文提出了瞬时激增负载适应度和$M/M/N$排队系统预测等待时间2个方案，也可以应用在物联网、车联网、卫星互联网等复杂网络场景的调度任务中，具有一定可拓展性和的普适性。

参考文献：

[1] BLEIKERTZ S, BUGIEL S, IDELER H, et al. Client-controlled cryptography-as-a-service in the cloud[C]//Applied Cryptography and Network Security. Berlin: Springer, 2013: 19-36.

[2] RAHMANI H, SUNDARARAJAN E, ALI Z M, et al. Encryption as a service (EaaS) as a solution for cryptography in cloud[J]. Procedia Technology, 2013, 11: 1202-1210.

[3] Key management as a service (KMaaS) explained[R]. 2023.

[4] 张岳公, 高志权, 邵淼, 等. 云密码服务技术白皮书[R]. 2019.
ZHANG Y G, GAO Z Q, SHAO M, et al. White paper on cloud cryptography service technology[R]. 2019.

[5] Future Market Insights. Cloud encryption market outlook (2023 to 2033)[R]. 2023.

[6] 李建鹏, 史国振, 李莉, 等. 异构云环境中的实时密码服务调度策略[J]. 计算机工程, 2019, 45(10): 1-7.
LI J P, SHI G Z, LI L, et al. Scheduling strategy for real-time cipher service in heterogeneous cloud environment[J]. Computer Engineering, 2019, 45(10): 1-7.

[7] 孙德洋, 娄嘉鹏, 李建鹏, 等. 异构云环境下的密码服务调度方法[J]. 计算机应用与软件, 2019, 36(6): 302-307, 333.
SUN D Y, LOU J P, LI J P, et al. Cryptographic service scheduling method in heterogeneous cloud environment[J]. Computer Applications and Software, 2019, 36(6): 302-307, 333.

[8] 寇文龙, 张宇阳, 李凤华, 等. 密码服务资源按需高效调度方案[J]. 通信学报, 2022, 43(6): 108-118.
KOU W L, ZHANG Y Y, LI F H, et al. On-demand and efficient scheduling scheme for cryptographic service resource[J]. Journal on Communications, 2022, 43(6): 108-118.

[9] NARAYANAN D, SANTHANAM K, KAZHAMIAKA F, et al. Heterogeneity-aware cluster scheduling policies for deep learning workloads[C]//Proceedings of the 14th USENIX Conference on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2020: 481-498.

[10] XIAO W C, BHARDWAJ R, RAMJEE R, et al. Gandiva: introspective cluster scheduling for deep learning[C]//Proceedings of the 13th USENIX Conference on Operating Systems Design and Implementation. Berkeley: USENIX Association, 2018: 595-610.

[11] 关川江, 李建鹏, 史国振, 等. 基于关联数据本地化的多密码作业流调度算法[J]. 计算机工程与科学, 2020, 42(11): 1988-1995.
GUAN C J, LI J P, SHI G Z, et al. A cloud cipher job stream scheduling algorithm based on associated data localization[J]. Computer Engineering & Science, 2020, 42(11): 1988-1995.

[12] HUSSAIN M, WEI L F, REHMAN A, et al. Deadline-constrained energy-aware workflow scheduling in geographically distributed cloud data centers[J]. Future Generation Computer Systems, 2022, 132(C): 211-222.

[13] LI X P, YU W, RUIZ R, et al. Energy-aware cloud workflow applications scheduling with geo-distributed data[J]. IEEE Transactions on Services Computing, 2022, 15(2): 891-903.

[14] SUN T, XIAO C B, XU X J. A scheduling algorithm using sub-deadline for workflow applications under budget and deadline constrained[J]. Cluster Computing, 2019, 22(3): 5987-5996.

[15] CHEN L, LI X P, RUIZ R. Idle block based methods for cloud workflow scheduling with preemptive and non-preemptive tasks[J]. Future Generation Computer Systems, 2018, 89: 659-669.

[16] XIE Y, SHENG Y H, QIU M Q, et al. An adaptive decoding biased random key genetic algorithm for cloud workflow scheduling[J]. Engineering Applications of Artificial Intelligence, 2022, 112: 104879.

[17] ANWAR N, DENG H F. A hybrid metaheuristic for multi-objective scientific workflow scheduling in a cloud environment[J]. Applied Sciences, 2018, 8(4): 538.

[18] CHOUDHARY A, GUPTA I, SINGH V, et al. A GSA based hybrid algorithm for bi-objective workflow scheduling in cloud computing[J]. Future Generation Computer Systems, 2018, 83: 14-26.

[19] VERMA A, KAUSHAL S. A hybrid multi-objective particle swarm optimization for scientific workflow scheduling[J]. Parallel Computing, 2017, 62: 1-19.

[20] MOHAMMADZADEH A, MASDARI M. Scientific workflow scheduling in multi-cloud computing using a hybrid multi-objective optimization algorithm[J]. Journal of Ambient Intelligence and Humanized Computing, 2023, 14(4): 3509-3529.

[21] YOU C Q. Hierarchical multi-resource fair queueing for network function virtualization[C]//Proceedings of the IEEE INFOCOM 2019 - IEEE Conference on Computer Communications. Piscataway: IEEE Press, 2019: 406-414.

[22] CALZAROSSA M C, VEDOVA M L D, MASSARI L, et al. Multi-objective optimization of deadline and budget-aware workflow scheduling in uncertain clouds[J]. IEEE Access, 2021, 9: 89891-89905.

[23] BELGACEM A, BEGHDAD-BEY K. Multi-objective workflow scheduling in cloud computing: trade-off between makespan and cost[J]. Cluster Computing, 2022, 25(1): 579-595.

[24] GOH L K, VEERAVALLI B, VISWANATHAN S. Design of fast and efficient energy-aware gradient-based scheduling algorithms heterogeneous embedded multiprocessor systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2009, 20(1): 1-12.

[25] JING C, ZHU Y M, LI M L. Energy-efficient scheduling on multi-FPGA reconfigurable systems[J]. Microprocessors & Microsystems, 2013, 37(6/7): 590-600.

[26] 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, 2014, 18(4): 577-601.

[27] DAS I, DENNIS J E. Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems[J]. SIAM Journal on Optimization, 1998, 8(3): 631-657.

[28] AL-SHARAEH S, WELLS B E. A comparison of heuristics for list schedules using the Box-method and P-method for random digraph generation[C]//Proceedings of 28th Southeastern Symposium on System Theory. Piscataway: IEEE Press, 1996: 467-471.

[29] WHILE L, HINGSTON P, BARONE L, et al. A faster algorithm for calculating hypervolume[J]. IEEE Transactions on Evolutionary Computation, 2006, 10(1): 29-38.

[30] DEB K, GONDKAR A, ANIRUDH S. Learning to predict Pareto-optimal solutions from pseudo-weights[C]//International Conference on Evolutionary Multi-Criterion Optimization. Berlin: Springer, 2023: 191-204.

[31] ALAHMADI A, ALNOWISER A, ZHU M M, et al. Enhanced first-fit decreasing algorithm for energy-aware job scheduling in cloud[C]//Proceedings of the 2014 International Conference on Computational Science and Computational Intelligence. Piscataway: IEEE Press, 2014: 69-74.

[32] HAHNE E L. Round-robin scheduling for max-min fairness in data networks[J]. IEEE Journal on Selected Areas in Communications, 1991, 9(7): 1024-1039.

[33] HSU C F, LIU C Y. An adaptive traffic-aware polling and scheduling algorithm for bluetooth piconets[J]. IEEE Transactions on Vehicular Technology, 2010, 59(3): 1402-1414.

[作者简介]

曹晓刚（1996– ），男，河北邢台人，中国科学院信息工程研究所博士生，主要研究方向为信息安全。

李凤华（1966– ），男，湖北浠水人，博士，中国科学院信息工程研究所研究员、博士生导师，主要研究方向为网络与系统安全、信息保护、隐私计算。

耿魁（1989– ），男，湖北红安人，博士，中国科学院信息工程研究所高级工程师、硕士生导师，主要研究方向为网络安全、信息保护。

李子孚（1992– ），女，内蒙古赤峰人，博士，中国科学院信息工程研究所高级工程师，主要研究方向为网络与系统安全。

寇文龙（1990– ），男，河南许昌人，中国科学院信息工程研究所工程师，主要研究方向为卫星互联网安全。