面向广域分布式智能计算的运行时算力网络资源协同调度方法研究

宋尧[1]，宋平[1]，高巍[1]，刘述[1]，霍志胜[2]
1. 中国信息通信研究院，北京 100191；
2. 北京航空航天大学计算机学院，北京 100191

摘要
随着人工智能等新一代信息通信技术飞速发展，广域分布式智能计算环境已成为一种重要基础设施。针对广域分布式智能计算环境中资源的高效协同调度难题，提出了一种面向广域分布式智能计算的运行时算力网络资源协同调度方法。该方法设计了关键任务决策与回填、基于关键流量调度的执行保障、数据自适应布局等策略，通过综合分析算力网络中的算、网、存资源使用情况，协同应用3类策略以优化运行时资源的全局利用。实验结果表明，相较于已有方法，该方法可有效提升系统吞吐量，并优化全局数据迁移开销。

关键词
任务调度；网络资源调度；数据布局；协同调度

中图分类号：TP316 文献标志码：A doi：10.11959/j.issn.2096–0271.2025037

Runtime collaborative scheduling of resources in computing power network oriented to the wide-area distributed intelligent computing environment

SONG Yao[1], SONG Ping[1], GAO Wei[1], LIU Shu[1], HUO Zhisheng[2]
1. China Academy of Information and Communications Technology, Beijing 100191, China
2. School of Computer Science and Engineering, Beihang University, Beijing 100191, China

Abstract
With the rapid development of new generation information and communication technologies such as artificial intelligence, the wide-area distributed intelligent computing environment has become an important infrastructure. In response to the challenge of efficient collaborative scheduling of resources in a wide-area distributed intelligent computing environment, a runtime collaborative scheduling of resources in computing power network method is proposed. The key task selection and backfilling, network traffic scheduling of core flow, and adaptive data placement strategies are designed. The usage of computing, network, and storage resources in the computing power network are

comprehensively analyzed, and the strategies are collaboratively implemented to optimized the global utilization of resources. The experimental results indicate that the proposed method can effectively improve the system throughput while optimizing the global data migration cost compared with the existing methods.

Key words
task scheduling, network resource scheduling, data placement, collaborative scheduling

0 引言

随着大数据、人工智能等新一代信息通信技术的发展，广域分布式智能计算已引起学术界和产业界的广泛关注[1]。广域分布式智能计算环境可发挥广域分散的算、网、存资源的聚合效应，进而提升计算效率[2-5]。我国陆续启动了中国算力网[6]、国家超算互联网[7]等以智能计算和超级计算为核心的算力网络建设工程。

在广域分布式智能计算环境中，网络是承载算力互联和数据流转的核心底座[8]。智算中心的选址与规划经常将网络因素作为重要考虑之一。但在广域网环境中，带宽受限和网络时延大将导致调度信息延迟[9]和数据迁移迟缓[10]等问题，进而造成任务执行效率低、任务长时间等待数据等情况，这给提升系统资源利用率和吞吐量带来了挑战。

调度方法是提升系统性能的有效措施之一。传统调度方法可分为任务调度[11-13]、网络资源调度[14-15]和数据布局[10, 16]等。任务调度方法通过发现系统资源空闲，编排系统中的任务以优化计算资源利用；网络资源调度通过路径规划、带宽分配等方式提升网络资源使用效率；数据布局方法通过对数据的热度和存储资源能力进行综合判断，使用数据副本等策略来降低数据迁移开销。但上述调度方法往往只针对算、网、存资源中的一个方面进行优化，优化方向单一，无法满足全局资源的优化利用需求，难以有效提升系统吞吐量等性能。随着信息技术的不断演进，通过协同调度方法综合利用广域环境中的多种资源成为提升系统性能的有效措施[8, 17-18]。为了保障调度决策的准确性，现有的协同调度方法需要通过广域网通信获取系统中各类资源的使用情况，这会带来较高的响应时延，难以满足智能计算任务的低时延需求。

针对上述挑战，本文提出了一种面向广域分布式智能计算的运行时算力网络资源协同调度方法。本方法通过综合分析算力网络中的算、网、存资源使用情况，协同调整运行时的任务分配、网络资源分配和数据布局。在任务分配优化方面，本方法综合系统资源信息，考虑任务开销与任务优先级，进行广域环境中任务队列间的任务回填，以提升计算效率；在网络资源分配优化方面，本方法实施关键任务最优路由和带宽分配机制，保障系统中的重要任务及时执行；在数据布局优化方面，本方法通过权衡存储资源状态和访问热度自适应地调整广域环境中的副本布局，以优化数据访问。

1 国内外现状

调度方法是通过优化任务分配、网络

资源分配和数据布局来提升系统性能的重要方式。目前，大部分调度方法通过任务调度实现计算任务的合理分配从而提升计算资源利用率，通过网络资源调度实现最优路由和带宽合理分配，或通过优化数据布局来降低数据迁移开销。因此，现有调度方法可分为任务调度[11-13]、网络资源调度[14-15]、数据布局[10,16]和协同调度[8,17-18]等。

1.1 任务调度方法

通过任务调度方法优化任务分配、利用空闲资源，是提升计算效率的重要优化手段。文献[19]采用了一种基于仿真和机器学习的动态回填调度方法，通过收集系统中用户历史信息并构建一个非线性方程来分析任务的特征，并预测任务的行为，从而配合回填方法更精准地优化任务分配。文献[20]提出了一种资源高效的工作负载任务调度方法，专注于降低能耗，利用少量资源带来良好的性能权衡。文献[21]提出了基于级联深度学习技术的调度方法，通过资源和任务特征的提取和分类，以较低响应时延完成最佳分配。文献[22]面向智能电网任务提出了动态资源调度方法，旨在优化资源利用率和经济利润。上述任务调度方法可以在一定程度上优化计算资源利用率，但由于系统中的数据布局、网络带宽等资源没有得到充分协同，现有的任务调度方法对系统性能的提升仍然有限。

1.2 网络资源调度方法

复杂多变的广域网络环境是影响系统性能的重要因素，通过带宽分配、路由策略等实现网络资源调度是应对网络挑战的有效措施。文献[14]提出一种基于关键流的流量调度优化算法，采用线性规划求解出每一条关键流的最优显式路径，并对普通流和关键流进行负载均衡，从而降低传输时延。文献[23]面向交通领域智能边缘计算场景提出了一种基于确定性策略梯度的带宽分区和功率分配优化策略，进而提升系统性能。上述网络资源调度方法以保障关键任务执行、降低数据传输时延为目标，但在决策关键任务或执行数据传输时，缺乏对广域网络动态变化和数据副本的考虑，因此难以有效提升任务执行效率。

1.3 数据布局方法

数据布局是存储系统中提升数据可靠性和访问性能的常用技术手段。Kosar等人[24]设计了STORK存储资源管理系统，通过数据副本、数据迁移管理等技术手段实现存储资源的高效访问，以实现网格环境中存储资源的高效管理。文献[10]和文献[16]面向边缘云环境提出了数据布局方法，通过在调度过程中感知数据的访问热度以及不同算力中心的计算和存储资源负载情况来优化数据布局。虽然数据布局方法可有效提升广域数据访问性能，但数据布局决策过程中对系统计算资源状态考虑不足易造成计算资源负载不均衡，进而损失计算性能。

1.4 协同调度方法

协同调度方法通过合理调度广域分散的算、网、存资源，有效提升计算效率。文献[8]提出了一种基于超图划分的位置感知调度方法，调度系统优先选择距数据源位置最近的数据中心作为任务调度目标，结合动态带宽分配方法降低数据迁移开销，并优化总体任务完成时间。文献[25]提出了一种容器化和任务调度结合的协同调度方法，基于数据分布和任务间的关联性对

任务和数据进行容器化聚合，然后采用关键路径优先机制保障重要任务的执行，以缩短总体任务完成时间。文献[26]和文献[27]提出了存算协同调度方法，通过协同任务调度和数据布局策略来提升系统性能。文献[28]提出了任务与网络协同调度方法，将推理任务拆分到不同的节点计算，从而提升调度灵活性。文献[29]提出了低成本的计算卸载和网络切片联合优化算法，以降低任务执行时延。但上述协同调度方法在调度决策过程中，需要通过广域网通信获取系统信息，这个过程中的响应时延难以满足智能计算任务的需求。

综上所述，现有的调度方法优化方向单一，无法满足对系统性能的优化需求。简单组合算、网、存资源的调度，难以实现各类资源信息的互通共享，各类调度方法对算、网、存资源做出的决策可能相互矛盾，无法高效协同。同时，现有的协同调度方法面向广域网复杂的环境难以满足响应时延需求。因此，本文设计了一种动态、协同、高效的调度方法，以实现广域分布式智能计算环境系统性能的提升。

2 方法研究

本文提出了一种面向广域分布式智能计算的运行时算力网络资源协同调度（runtime collaborative scheduling of resources in computing power network，RCSR）方法，该方法综合系统中的任务分配、数据布局、系统资源等信息，对任务、网络、数据进行协同调度决策。RCSR方法可避免现有调度方法中优化方向单一的问题，有效优化系统执行过程中资源的全局利用，进而提升系统的吞吐量。

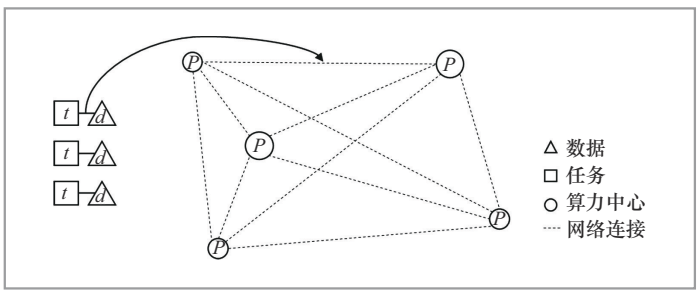


图1 广域分布式智能计算环境模型

2.1 模型构建

为了更加清晰地描述广域分布式智能计算环境中的任务、数据、包含存算资源的算力中心以及算力中心间互联网络等要素，RCSR方法构建模型如**图1**所示。

RCSR方法用P表示算力中心，算力中心内包含计算和存储资源，$C_{P,\max}$，$C_{P,\text{used}}$分别表示算力中心P的总算力资源和已占用算力资源，$S_{P,\max}$，$S_{P,\text{used}}$分别表示算力中心P的总存储资源和已占用存储资源。RCSR方法以N_{P_1,P_2}表示算力中心P_1和P_2间的网络连接，本方法假设系统是全连接的，$\text{BW}_{N_{P_1,P_2},\max}$表示网络连接$N_{P_1,P_2}$的最大带宽。RCSR方法以$t$表示系统中的任务，$d$表示任务所需的输入数据，$C_t$表示任务所需的计算资源量，$S_d$表示数据所需的存储资源量。本方法假设每个任务都需要对应的输入数据，不同任务间可能共享同一份输入数据。

2.2 策略研究

如**图2**所示，RCSR方法通过综合系统中的算、网、存资源信息和任务及数据分布情况，采用关键任务决策与回填策略，以充分利用系统中的空闲资源；面向选定的关键任务，RCSR方法通过基于关键流量调度的执行保障策略，优先传输关键任务

的输入数据，以保障执行；数据传输完成后，RCSR方法通过权衡当前算力中心的存储资源状态和该数据访问热度，实施数据自适应布局策略，降低后续网络传输负载。

在调度过程中，RCSR方法进行了两方面的协同。在信息协同方面，所提出的调度策略在资源信息、数据布局信息、任务分配信息等方面进行共享，并基于这些信息进行综合决策。在决策协同方面，RCSR方法中所有调度策略的决策结果都将反馈至系统以动态更新信息，并影响其他策略决策过程。例如：任务分配结果将影响网络资源分配时的关键流量和数据布局时的访问热度；流量调度结果将改变数据迁移时间，并影响任务调度和数据副本生成；数据布局的调整将改变数据分布特征和数据迁移开销，进而影响任务和网络资源的调度。通过3类调度策略的协同运行，RCSR方法可优化系统运行过程中资源的全局利用，进而提升系统吞吐量等性能。

（1）关键任务决策与回填策略

如**图3**所示，广域网环境中调度信息延迟和数据迁移迟缓等问题可能造成调度决策不准确，导致实际执行时任务延迟，进而产生资源碎片。此时，任务回填策略可有效利用这些资源碎片提升系统性能。RCSR方法通过感知数据分布特征和资源空闲情况，提出关键任务决策与回填策略。当感知到算力中心的资源空闲时，该策略在广域环境中的多个任务队列间进行任务回填以优化任务分配，从而提升系统执行效率。

本策略将关键任务定义为可用较小的代价进行回填并合理利用空闲资源、提升整体执行效率的任务t_x。对于存在空闲资源的算力中心P，本策略通过感知数据分布特征和全局资源负载，从其他算力中心内正在排队等待执行的任务中选取关键任务。算力中心P包含的空闲资源应可满足

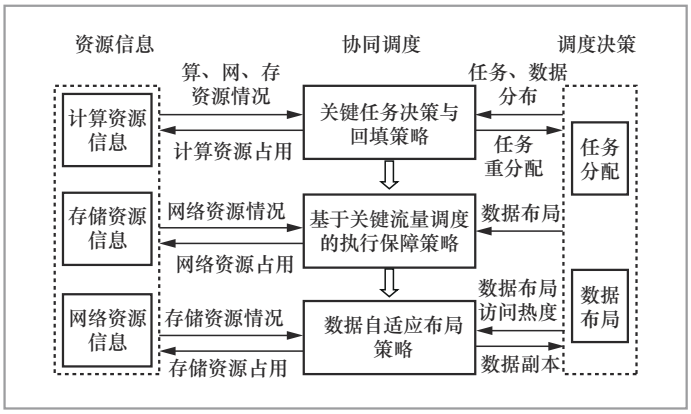


图2 面向广域分布式智能计算的运行时算力网络资源协同调度方法

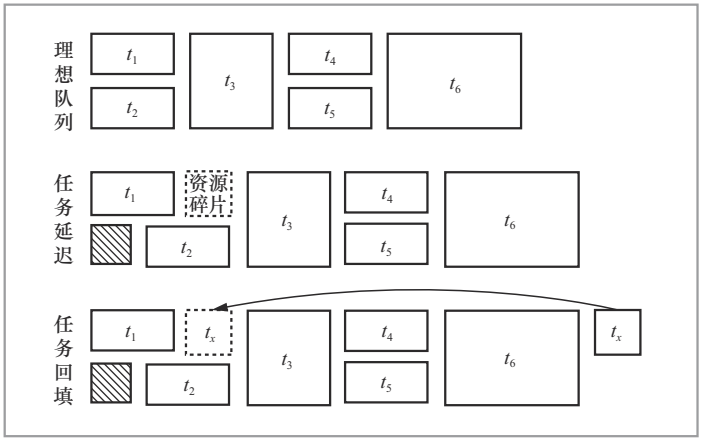


图3 任务回填策略

关键任务t_x的资源需求。在此基础上，本策略定义关键任务搜索集合$\mathrm{Task}_{P,\mathrm{data}}$和$\mathrm{Task}_{P,\mathrm{comp}}$。对于$\mathrm{Task}_{P,\mathrm{data}}$中的任务，其所需输入数据均在算力中心$P$内存在副本，因此该任务被回填后可立即开始执行，而无须经历数据迁移过程。对于$\mathrm{Task}_{P,\mathrm{comp}}$中的所有任务，其所需输入数据在算力中心$P$内不存在副本，因此这些任务在被回填时需要经历数据迁移阶段。本策略将优先从集合$\mathrm{Task}_{P,\mathrm{data}}$中选取关键任务，避免在任务回填期间产生二次数据迁移开销，在降低网络负载的同时保障广域任务窃取与回填策略的时效性。仅当$\mathrm{Task}_{P,\mathrm{data}}$为空集或

其中的所有任务都不符合约束时，才会从集合 $\text{Task}_{P,\text{comp}}$ 中选取关键任务。此外，本策略将优先从负载较高的算力中心内选取关键任务，以均衡算力中心间的负载。对于每个任务 t，设定其关键任务参数 $K(t)$ 如式（1）所示。

$$K(t)=\begin{cases}\dfrac{\text{priority}_t}{\left\|\overrightarrow{L_{t,P_s}}\right\|}, t\in \text{Task}_{P,\text{data}} \\ \dfrac{\text{priority}_t}{\left\|\overrightarrow{L_{t,P_s}}\right\|\times T_{d,P,\text{tran}}}, t\in \text{Task}_{P,\text{comp}}\end{cases} \tag{1}$$

在式（1）中，priority_t 表示任务 t 的优先级。本策略基于任务的计算量定义任务优先级，较小的任务会被优先选取，以降低系统中的平均等待时间。向量 $\overrightarrow{L_{t,P_s}}$ 表示任务 t 原本所在的算力中心 P_s 的资源综合空闲程度，如式（2）定义。算力中心 P_s 的资源空闲程度越大，该中心内任务被选中的可能性就越低。当需要从集合 $\text{Task}_{P,\text{comp}}$ 中选取关键任务时，任务的筛选条件中加入了数据迁移时间 $T_{d,P,\text{tran}}$，即任务 t 的输入数据 d 被迁移至目标算力中心 P 所需的时间，数据迁移时间可由 RCSR 方法中的基于关键流量调度的执行保障策略控制。数据迁移开销越大的任务，被选中的可能性越低，以避免产生较大的额外数据迁移开销。

$$\overrightarrow{L_{t,P_s}}=(L_{t,P_s,\text{comp}},L_{t,P_s,\text{stor}}) \tag{2}$$

其中，$L_{t,P_s,\text{comp}}$ 和 $L_{t,P_s,\text{stor}}$ 分别代表 min-max 归一化处理后算力中心 P_s 的相对计算和存储资源空闲程度。算力中心 P_s 的原始资源空闲程度 $L'_{t,P_s,\text{comp}}$ 和 $L'_{t,P_s,\text{stor}}$ 如式（3）所示。

$$\begin{cases}L'_{t,P_s,\text{comp}}=\dfrac{C_{P,\max}-C_{P,\text{used}}-C_t}{C_{P,\max}} \\ L'_{t,P_s,\text{stor}}=\dfrac{S_{P,\max}-S_{P,\text{used}}-S_d}{S_{P,\max}}\end{cases} \tag{3}$$

为提升搜索的效率以保障在较短时间内找出合适的关键任务，本策略采用爬山算法[30]寻找局部最优解。对于在搜索集合 $\text{Task}_{P,\text{data}}$ 和 $\text{Task}_{P,\text{comp}}$ 中的每个任务 t，设定式（4）~式（6）所示的筛选条件。

$$t_x=\arg\max K(t) \tag{4}$$

$$T_{d_x,P,\text{tran}}+T_{t_x,P,\text{exec}}<T_{t_P,P,\text{queue}} \tag{5}$$

$$T_{d_x,P,\text{tran}}<T_{t_s,P_s,\text{queue}} \tag{6}$$

式（5）保证了任务被回填后不会导致目标算力中心内排队任务的等待时间增长，即任务 t_x 在目标算力中心 P 内的计算时间 $T_{t_x,P,\text{exec}}$ 与其所需的输入数据 d_x 被迁移至目标算力中心 P 的迁移时间 $T_{d_x,\text{P},\text{tran}}$ 之和小于目标算力中心 P 内下一排队任务 t_P 的预期队列等待时间 $T_{t_P,P,\text{queue}}$。式（6）保证了关键任务的开始执行时间不会延后，即数据 d_x 迁移时间 $T_{d_x,P,\text{tran}}$ 应小于任务 t_x 在原算力中心 P_s 的队列等待时间 $T_{t_s,P_s,\text{queue}}$。关于任务的计算时间和队列等待时间，本策略采用文献[5]、文献[31]和文献[32]等文中的方法进行计算。本策略选择出满足条件的关键任务 t_x 后，将该任务重调度至目标算力中心 P 执行，以有效利用空闲资源，提升系统性能。

（2）基于关键流量调度的执行保障策略

在选定关键任务后，RCSR 方法采用基于关键流量调度的执行保障策略，对选定关键任务 t_x 所需的输入数据 d_x 进行传输。本策略将关键流量定义为输入数据从数据源算力中心 P_s 传输至目标算力中心 P 的数据传输任务。本策略通过调整关键流量所在网络连接上其他传输任务的带宽分配来保障关键流量获取尽可能大的传输带宽。

由于数据 d_x 在系统中可能存在多个副本，本策略从所有包含数据 d_x 副本的算力

中心集合$\{P_{x,\text{all}}\}$中选取作为数据源的算力中心P_s，在保障系统中其他任务按计划执行的前提下，使关键流量的数据传输时间$T_{d_{x,P_s},P,\text{tran}}$最短。在选择数据源算力中心$P_s$时，应满足式（7）~式（10）所示的条件。

$$P_s=\arg\min_{P'\in\{P_{x,\text{all}}\}} T_{d_{x,P'},P,\text{tran}} \tag{7}$$

$$T_{d_{x,P'},P,\text{tran}}=\frac{S_{d_{x,P'}}}{\text{BW}_{N_{P',P},d_{x,P'}}} \tag{8}$$

$$\text{BW}_{N_{P',P},\max}-\sum_{\forall d'_{P'}}\text{BW}_{N_{P',P},d'_{P'},\min}\leqslant \text{BW}_{N_{P',P},d_{x,P'}}\leqslant \text{BW}_{N_{P',P},\max} \tag{9}$$

$$T_{d'_{P'},P,\text{tran}}\leqslant T_{t',P,\text{queue}} \tag{10}$$

式（8）表示数据副本$d_{x,P'}$从数据源算力中心P'到目标算力中心P所需的数据传输时间。式（9）约束了网络连接$N_{P',P}$中分配给数据副本$d_{x,P'}$传输任务的带宽范围，其中$\text{BW}_{N_{P',P},d'_{P'},\min}$表示分配给网络连接$N_{P',P}$上任意数据$d'_{P'}$传输任务的最小带宽，该参数的值由式（10）决定。式（10）约束了任意任务t'对应输入数据$d'_{P'}$的传输时间应小于任务t'在算力中心P的排队等待时间。式（9）和式（10）表示给数据副本$d_{x,P'}$的传输任务分配带宽后，可能降低网络连接$N_{P',P}$上其他数据传输任务的传输速度，但仍需保证这些数据在对应任务排队结束前可到达目标算力中心P。

对于数据d_x的所有数据副本，本策略选取最佳的数据副本进行传输，采用带宽分配方法保障关键流量的最大传输带宽，并对该网络连接上的其他传输任务采用负载均衡策略，以实现关键任务的及时执行，同时不影响系统中其他任务的执行。

（3）数据自适应布局策略

在数据传输任务完成后，RCSR方法通过感知系统中的数据访问热度和存储资源状态，实施数据自适应布局策略。本策略通过数据访问热度和存储资源状态间的权衡，决定是否在目标算力中心P生成副本，优化数据布局以降低后续数据迁移开销。

对于数据d，本策略根据任务对数据的访问信息计算数据副本的访问热度。数据d在算力中心P中的访问热度$h_{d,P}$定义如式（11）所示。

$$h_{d,P}=f_{d,P}\times\left(1-\frac{\Delta t_{d,P}-\Delta t_{\min,d,P}}{\Delta t_{\max,d,P}-\Delta t_{\min,d,P}}\right) \tag{11}$$

在式（11）中，$f_{d,P}$是算力中心P内数据d的访问次数与该数据的全局访问次数的比值。$\Delta t_{d,P}$代表算力中心P内上一次与本次访问数据d的时间间隔。在计算数据访问热度时，本策略对访问时间间隔进行min-max归一化以使数据访问热度可统一衡量。访问时间间隔越小、访问比例越高的数据，访问热度就越大。

本策略设定了数据访问热度和存储资源情况的权衡评价值Score_d，如式（12）所示。

$$\text{Score}_d=\sqrt{h_{d,P}\times L_{t,P,\text{stor}}} \tag{12}$$

式（12）中，$L_{t,P,\text{stor}}$是归一化后算力中心P相对全局的存储资源空闲程度，本策略采用数据访问热度和存储资源空闲程度的几何平均值作为权衡评价值。对于目标算力中心存储资源相对空闲但访问热度高的数据，权衡评价值较高，本策略更倾向于对该数据生成副本。与之相对，当权衡评价值较低时，说明数据的访问热度低、算力中心存储资源负载较大，此时本策略将不对数据生成副本，以避免加重存储资源负载。同时，对于访问时间间隔$\Delta t_{d,P}$过大的数据d，系统将动态删除其在算力中心P内的副本。

2.3 调度算法设计

RCSR 方法分别采用关键任务决策与回填、基于关键流量调度的执行保障、数据自适应布局等策略，以实现广域分布式智能计算环境中算、网、存资源的协同调度，优化运行时资源的全局利用，提升系统吞吐量并降低数据迁移开销。RCSR 方法伪代码如算法 1 所示。

算法 1　面向广域分布式智能计算的运行时算力网络资源协同调度方法

输入：资源使用情况、任务队列信息、数据布局信息、存在空闲资源的算力中心 P；

输出：任务、数据、网络资源调度决策；

```
if Task_{P,data} ≠ ∅ then                    //关键任务决策与回填策略
    for all t ∈ Task_{P,data} do
        根据式（1）在不考虑数据迁移的情况下计算 K(t)；
        根据式（4）~式（6）选定关键任务 t_x；
    end for
else
    for all t ∈ Task_{P,comp} do
        根据式（1）在考虑数据迁移的情况下计算 K(t)；
        根据式（4）~式（6）选定关键任务 t_x；
    end for
end if
生成关键任务回填决策；
生成关键任务 t_x 对应输入数据 d_x 的所有副本位置集合 {P_{x,all}}；//基于关键流量调度的执行保障策略
if P ∈ {P_{x,all}} then
    continue;
else
    for all P' ∈ {P_{x,all}} do
        根据式（9）和（10）计算数据传输带宽 BW_{N_{P'P},d_{x,P'}} 的取值范围；
        根据式（8）计算 T_{d_{x,P'},P,tran}；
        根据式（7）选定数据源算力中心 P_s；
    end for
end if
生成关键流量调度决策；
for all d in P do    //数据自适应布局策略
    根据式（11）计算数据访问热度 h_{d,P}；
    根据式（12）计算权衡评价值 Score_d；
    依据 Score_d 判定是否生成副本；
    依据访问时间间隔 Δt_{d,P} 判定是否删除副本；
end for
生成数据布局决策；
```

3 实验结果

3.1 实验环境

本实验将地理位置分布的5个算力节点构建为广域分布式智能计算环境以进行实验验证。本实验通过广域分布式智能计算环境中受限的网络带宽、动态变化的资源负载验证RCSR方法的优化效果。在实验环境中，设定各算力中心的计算和存储资源能力多样，见**表1**。此外，本实验对实验环境中各算力中心间的网络带宽进行了测试，如**图4**所示。

本实验采用多个标准的计算任务及其相应输入数据在实验环境中进行验证，设定了所有输入数据的初始布局，并在实验期间根据各种调度执行优化方法通过节点间的链路按需对输入数据进行复制和迁移。**表1**中，相较于智算中心的真实运行环境，本实验中各节点使用的算力和存储资源规模都较小。然而，本文提出的RCSR方法通过发现和利用系统中空闲的算、网、存资源，协同调度任务、网络和数据以提升性能，小规模仿真实验环境和真实环境在产生和利用空闲资源的方面并不存在差异表现。因此，本实验对分析验证RCSR方法在实际环境中的部署应用效果具有重要意义。

3.2 对比基准算法

本实验采用侧重数据重分布的runData方法[10]、侧重算网协同调度的JTSC方法[25]和侧重存算协同调度的DRS方法[26]作为对比基准算法。runData方法是一种弱协同调度方法，通过感知数据访问热度和资源状态变化对数据布局进行动态优化来降低数据迁移开销，然后根据数据布局对任务分配进行调整，从而提升系统吞吐量。JTSC方法基于数据分布和任务间的关联性对任务进行容器化聚合，对任务集进行二阶段调度以降低总体任务完成时间，同时采用关键路径优先机制以获得性能提升。DRS方法根据系统资源信息和数据分布特征进行任务和数据的联合调度，根据调度结果的反馈实施数据副本，以获得系统在数据访问和任务执行方面的性能提升。本文提出的RCSR方法与各基准算法的对比情况见**表2**。

表1 RCSR方法广域分布式计算环境中实验环境配置

算力中心	总计算能力/(TFLOPS)	总存储容量/GB
算力中心1	40	35
算力中心2	21	30
算力中心3	40	15
算力中心4	29	40
算力中心5	35	35

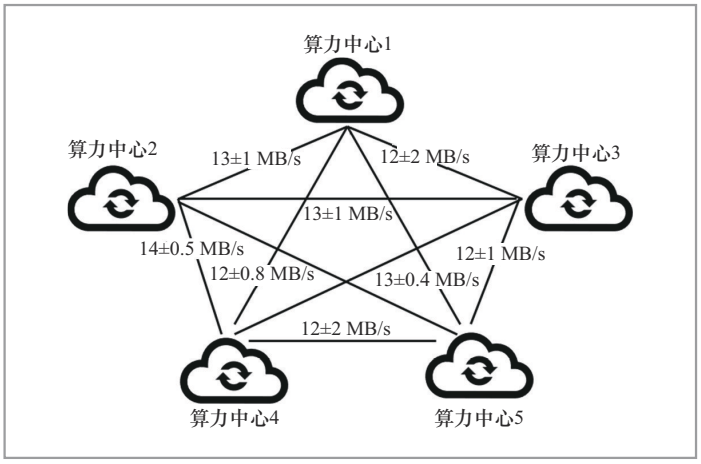


图4 广域分布式智能计算实验环境

3.3 实验结果分析

本实验从系统吞吐量和全局数据迁移

表2 RCSR方法广域分布式计算环境实验中各对比算法特征

方法名称	任务调度	网络资源调度	数据布局	协同优化
runData	√		√	√
JTSC	√	√		√
DRS	√		√	√
RCSR	√	√	√	√

开销等方面分析了实验结果。其中，系统吞吐量是反映系统执行效率的主要性能评价指标，全局数据迁移开销性能可反映RCSR方法中数据布局和网络资源调度策略带来的性能影响。

（1）系统吞吐量

系统吞吐量是在系统运行的一定时间段内完成的任务总数量。本实验将100个计算任务作为输入，并将系统分别运行100 s、200 s、300 s、400 s、500 s、600 s，以测试系统在运行时间内的吞吐量指标。各对比方法在系统吞吐量方面的实验结果如**图5**所示。

如**图5**所示，系统采用RCSR方法的吞吐量实验结果在各种运行时间下相较于对比基准算法都是更优的，在运行时间为600 s时，分别优于runData、JTSC、DRS方法5.37%、11.36%和3.15%。

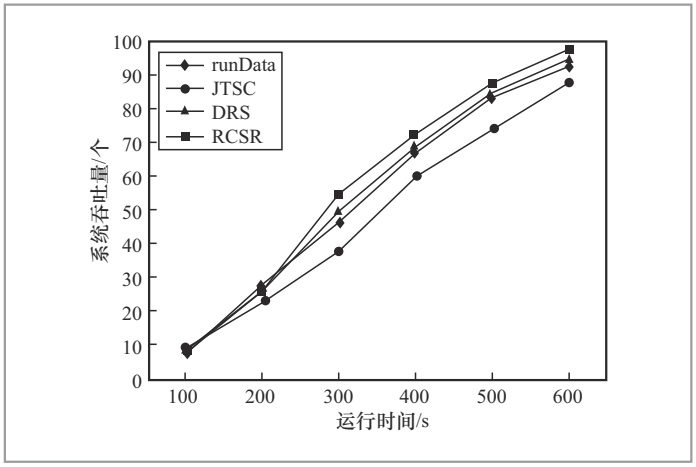


图5 系统吞吐量实验结果

JTSC方法将任务和数据“打包”进行调度，考虑了网络资源的动态分配，但没有协同优化数据布局，导致对系统吞吐量的优化效果有限。runData方法更加侧重数据重分布，通过分析数据热度在执行过程中对数据布局进行优化，并基于数据布局对任务分配进行调整。runData方法可以有效地优化系统中的任务分配和数据布局，但未采用数据副本机制，因此其系统吞吐量性能还有进一步的提升空间。DRS方法有效协同了数据副本机制和任务调度策略，在系统吞吐量性能方面有较好的表现，但缺乏对网络资源的控制，难以有效保障系统中瓶颈任务的执行。本文提出的RCSR方法通过综合考虑资源负载情况、数据访问热度等多重因素，协同调整系统中的任务分配、网络资源分配和数据布局，使系统中的任务执行更加高效，因此RCSR方法相较于对比基准算法具有更优的系统吞吐量性能。

（2）全局数据迁移开销

本实验中全局数据迁移开销指在系统执行过程中算力中心间通过广域网进行数据迁移的总时间开销。全局数据迁移开销体现了系统中数据布局的合理程度，可在一定程度上体现RCSR方法中数据自适应布局策略对数据布局的优化效果，并反映基于关键流量调度的执行保障策略的性能影响。本实验将100个计算任务作为输入，测试各对比方法在全局数据迁移开销方面的实验结果，如**图6**所示。

如**图6**所示，系统采用RCSR方法在各种情况下的全局数据迁移开销平均优于采用JTSC方法6.73%，平均劣于采用runData方法133.97%，而与DRS方法效果相近。

在系统采用JTSC方法调度时，系统

中的数据都将跟随任务一同迁移，因此在任务执行期间将造成较多的数据迁移开销。随着任务数量的增长，未采用数据布局机制的JTSC方法在全局数据迁移开销方面与其他方法的性能差距越发明显。runData方法着重优化了数据布局，并基于数据布局对任务进行重调度，因此runData方法在执行过程中具有相对较小的全局数据迁移开销。DRS方法和RCSR方法为了提升系统吞吐量性能均采用了数据副本机制，数据副本生成、一致性维护等过程都会产生数据迁移开销。此外，RCSR方法中的网络资源调度策略将增加部分数据的迁移开销。相较于未采用数据副本机制的runData方法，DRS方法和RCSR方法都具有更优的系统吞吐量性能，而在全局数据迁移开销性能上做出了让步。由图6可见，在系统负载较小时，即任务数量较少时，数据副本机制带来的额外开销将造成较大的性能损失，此时DRS方法和RCSR方法的全局数据迁移开销达到runData方法的近3倍。然而，随着系统负载的增加，数据副本机制的优势将会更加明显，RCSR方法在全局数据迁移开销性能上与runData方法的差距逐渐缩小。在任务数量为100时，RCSR方法的全局数据迁移开销高于runData方法87.78%。

综上所述，本文提出的RCSR方法相较于其他对比基准算法可有效提升系统的吞吐量并优化全局数据迁移开销，进而提升广域分布式智能计算环境中的系统性能。

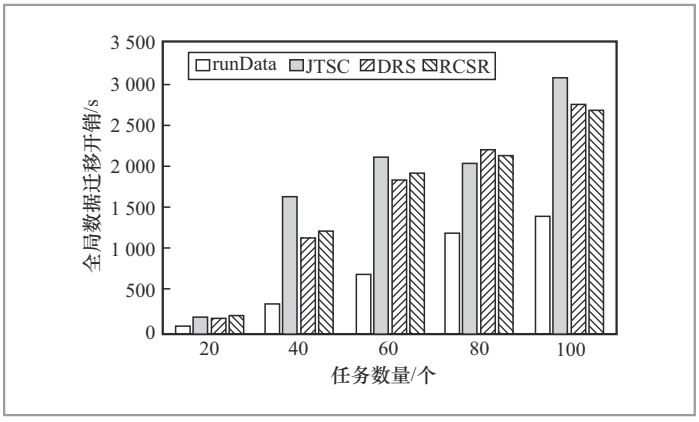


图6 全局数据迁移开销实验结果

4 结束语

本文针对广域分布式智能计算环境中资源的高效协同调度难题，研究了运行时算力网络资源协同调度方法。本文提出的RCSR方法通过综合分析算力网络中的算、网、存资源使用情况，协同调整运行时的任务分配、网络资源分配和数据布局，从而实现执行过程中资源全局利用的优化。在广域环境中的实验验证和理论分析表明，本文方法相比于现有方法可有效提升系统吞吐量并优化全局数据迁移开销，进而实现了广域分布式智能计算环境系统性能的提升。

参考文献：

[1] ZHU S Q, YU T, XU T, et al. Intelligent computing: the latest advances, challenges, and future[J]. Intelligent Computing, 2023, 2: 6.

[2] TOWNS J, COCKERILL T, DAHAN M, et al. XSEDE: accelerating scientific discovery[J]. Computing in Science & Engineering, 2014, 16(5): 62-74.

[3] GAGLIARDI F. The EGEE European grid infrastructure project[M]//High Performance Computing for Computational Science - VECPAR 2004. Heidelberg: Springer, 2005: 194-203.

[4] DEPEI Q. CNGrid: a test-bed for grid

technologies in China[C]//Proceedings of 10th IEEE International Workshop on Future Trends of Distributed Computing Systems, 2004. Piscataway: IEEE Press, 2004: 135-139.

[5] KANG S, VEERAVALLI B, AUNG K M M. Dynamic scheduling strategy with efficient node availability prediction for handling divisible loads in multi-cloud systems[J]. Journal of Parallel and Distributed Computing, 2018, 113: 1-16.

[6] 高文. 中国算力网的机遇与挑战[J]. 中国计算机学会通讯, 2023(2): 1-6.
GAO W. The opportunities and challenges of China's computing power network[J]. Communications of the CCF, 2023(2): 1-6.

[7] 钱德沛, 栾钟治, 刘轶. 从网格到"东数西算": 构建国家算力基础设施[J]. 北京航空航天大学学报, 2022, 48(9): 1561-1574.
QIAN D P, LUAN Z Z, LIU Y. From grid to"east-west computing transfer": constructing national computing infrastructure[J]. Journal of Beijing University of Aeronautics and Astronautics, 2022, 48(9): 1561-1574.

[8] ZHAO L P, YANG Y N, MUNIR A, et al. Optimizing geo-distributed data analytics with coordinated task scheduling and routing[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(2): 279-293.

[9] XU K, LYU L, LI T, et al. Minimizing tardiness for data-intensive applications in heterogeneous systems: a matching theory perspective[J]. IEEE Transactions on Parallel and Distributed Systems, 2020, 31(1): 144-158.

[10] JIN Y B, QIAN Z Z, GUO S, et al. Run runData: re-distributing data via piggybacking for geo-distributed data analytics over edges[J]. IEEE Transactions on Parallel and Distributed Systems, 2022, 33(1): 40-55.

[11] LI Z J, CHANG V, HU H Y, et al. Real-time and dynamic fault-tolerant scheduling for scientific workflows in clouds [J]. Information Sciences, 2021, 568: 13-39.

[12] GAUSSIER E, LELONG J, REIS V, et al. Online tuning of EASY-backfilling using queue reordering policies[J]. IEEE Transactions on Parallel and Distributed Systems, 2018, 29(10): 2304-2316.

[13] CHUNG M T, WEIDENDORFER J, SAMFASS P, et al. Scheduling across multiple applications using task-based programming models[C]//Proceedings of the 2020 IEEE/ACM Fourth Annual Workshop on Emerging Parallel and Distributed Runtime Systems and Middleware. Piscataway: IEEE Press, 2020: 1-8.

[14] 赵鹏程, 于俊清, 李冬. 一种基于深度学习的SRv6网络流量调度优化算法[J]. 信息网络安全, 2024, 24(2): 272-281.
ZHAO P C, YU J Q, LI D. An optimal algorithm for traffic scheduling in SRv6 network based on deep learning[J]. Netinfo Security, 2024, 24(2): 272-281.

[15] SANJAY G, PRASHANTH Y S, REDDY M S, et al. Bayesian optimization neural network based opportunistic routing in WSN[M]//Proceedings of International Conference on Computational Intelligence. Singapore: Springer Nature Singapore, 2024: 427-438.

[16] SHI W Y, TANG Q. Retraction Note: Cost-optimized data placement strategy for social network with security awareness in edge-cloud computing environment[J]. Journal of Combinatorial Optimization, 2024, 47(3): 51.

[17] LI C L, BAI J P, TANG J H. Joint optimization of data placement and scheduling for improving user experience in edge computing[J]. Journal of Parallel and Distributed Computing, 2019, 125: 93-105.

[18] BREITBACH M, SCHÄFER D, EDINGER J, et al. Context-aware data and task placement in edge computing environ-

ments[C]//Proceedings of the 2019 IEEE International Conference on Pervasive Computing and Communications. Piscataway: IEEE Press, 2019: 1-10.

[19] CARASTAN-SANTOS D, DE CAMARGO R Y. Obtaining dynamic scheduling policies with simulation and machine learning[C]//Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. New York: ACM, 2017: 1-13.

[20] CHOWDAIAH N K, DAMMUR A. Resource-efficient workload task scheduling for cloud-assisted Internet of Things environment[J]. International Journal of Electrical and Computer Engineering, 2023, 13(5): 5898.

[21] VIJAYASEKARAN G, DURAIPANDIAN M. An improved resource scheduling strategy through concatenated deep learning model for edge computing IoT networks[J]. International Journal of Communication Systems, 2024, 37(7): e5724.

[22] ZOU J, XIN P Z, WANG C, et al. AI services-oriented dynamic computing resource scheduling algorithm based on distributed data parallelism in edge computing network of smart grid[J]. Future Internet, 2024, 16(9): 312.

[23] KE H C, WANG H, SUN H B. Deep reinforcement learning-based power control and bandwidth allocation policy for weighted cost minimization in wireless networks[J]. Applied Intelligence, 2023, 53(22): 26885-26906.

[24] KOSAR T, BALMAN M. A new paradigm: data-aware scheduling in grid computing[J]. Future Generation Computer Systems, 2009, 25(4): 406-413.

[25] ZHANG J W, ZHOU X C, GE T Y, et al. Joint task scheduling and containerizing for efficient edge computing[J]. IEEE Transactions on Parallel and Distributed Systems, 2021, 32(8): 2086-2100.

[26] RAMBABU D, GOVARDHAN A. Data replication and scheduling in the cloud with optimization assisted work flow management[J]. Multimedia Tools and Applications, 2024, 83(27): 68883-68905.

[27] LI C L, LIU J, WANG M, et al. Fault-tolerant scheduling and data placement for scientific workflow processing in geo-distributed clouds[J]. Journal of Systems and Software, 2022, 187: 111227.

[28] YANG Y T, WEI H Y. Edge - IoT computing and networking resource allocation for decomposable deep learning inference[J]. IEEE Internet of Things Journal, 2023, 10(6): 5178-5193.

[29] 王晔, 王逸飞, 陈康, 等. 6G网络任务卸载与细粒度切片资源调度联合优化算法[J]. 电信科学, 2024, 40(5): 86-99.
WANG Y, WANG Y F, CHEN K, et al. Joint optimization algorithm for 6G network task offloading and fine-grained slice resource scheduling[J]. Telecommunications Science, 2024, 40(5): 86-99.

[30] MASADEH R, SHARIEH A, JAMAL S, et al. Best path in mountain environment based on parallel hill climbing algorithm [J]. International Journal of Advanced Computer Science and Applications, 2020, 11(9): 107-116.

[31] SONG Y, WANG L, XIAO L M, et al. Hypergraph-partitioning-based online joint scheduling of tasks and data[J]. The Journal of Supercomputing, 2022, 78(14): 16088-16117.

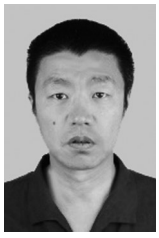
[32] MCKENNA R, HERBEIN S, MOODY A, et al. Machine learning predictions of runtime and IO traffic on high-end clusters[C]//Proceedings of the 2016 IEEE International Conference on Cluster Computing. Piscataway: IEEE Press, 2016: 255-258.

作者简介

宋尧（1994– ），男，博士，中国信息通信研究院技术与标准研究所工程师，主要研究方向为高性能计算、边缘计算、智能计算、算网融合、分布式存储、分布式调度系统、存算联动调度等。

宋平（1987– ），男，博士，中国信息通信研究院技术与标准研究所高级工程师，主要研究方向为高性能计算、边缘计算、人工智能等。

高巍（1978– ），男，中国信息通信研究院技术与标准研究所互联网中心主任、高级工程师，主要研究方向为数据通信、云计算、工业互联网等。



刘述（1972– ），男，中国信息通信研究院技术与标准研究所高级工程师，主要研究方向为信息技术、网络技术、智算网络等。

霍志胜（1983– ），男，博士，北京航空航天大学计算机学院助理研究员，主要研究方向为大数据存储、分布式存储系统、分布式/并行文件系统等。作为项目主持人和项目骨干，主持和参与了多项博士后基金面上项目、国家重点研发计划项目、国家自然科学基金面上项目等。

收稿日期: 2025-02-13

通信作者: 宋平,songping1@caict.ac.cn

基金项目: 国家重点研发计划项目(No.2024YFB2908701)

Foundation Item: The National Key Research and Development Program of China (No.2024YFB2908701)