采用机器学习的大规模并行程序 I/O 性能建模与预测方法

刘恒[1]，王子衡[1]，王强[1,2]，刘新栋[1]，龙炳志[3]，董小社[1]
（1. 西安交通大学计算机科学与技术学院，710049，西安；
2. 西安交通大学网络信息中心，710049，西安；3. 西安交通大学软件学院，710049，西安）

摘要：针对大规模并行程序因输入/输出(I/O)数据难以获取、建模与优化成本高昂而导致的性能分析效率低下和调优困难问题，提出了一种新的机器学习驱动的大规模并行程序 I/O 性能建模与预测方法。该方法在建模阶段，一方面基于小规模节点环境下的采样数据，利用线性回归方法构建可用于大规模节点外推的并行程序 I/O 特征预测模型；另一方面，将小规模节点环境下采集的并行程序 I/O 特征与对应的 I/O 栈参数空间共同输入人工神经网络(ANN)进行训练，以学习系统配置参数与 I/O 性能之间的非线性映射关系，从而构建并行程序的 I/O 性能预测模型。在预测阶段，使用 I/O 特征预测模型预测大规模并行程序的 I/O 特征，并将其与对应的大规模并行程序 I/O 栈参数空间输入 I/O 性能预测模型，实现对大规模并行程序 I/O 性能的准确外推预测。实验结果表明：在国产超级计算机上，采用所提方法在 4 种典型测试程序 IOR、S3D-IO、BT-IO 和 Flash-IO 上的预测精度均较高；当采用 1～16 节点规模下训练得到的模型对 128 节点(2 048 进程)场景进行外推预测时，其平均绝对百分比误差分别为 19.07%、18.96%、12.34%和 14.16%；基于所构建的 I/O 性能预测模型对 I/O 栈参数调优后，4 种典型程序 I/O 性能加速比分别达到 16.38、23.16、45.36 和 65.38 倍。

关键词：大规模并行程序；I/O 性能建模；I/O 自动调优；人工神经网络
中图分类号：TP316 **文献标志码**：A
DOI：10.7652/xjtuxb202607019 **文章编号**：0253-987X(2026)07-0207-12

I/O Performance Modeling and Prediction Method for Large-Scale Parallel Applications Using Machine Learning

LIU Heng[1], WANG Ziheng[1], WANG Qiang[1,2], LIU Xindong[1],
LONG Bingzhi[3], DONG Xiaoshe[1]
(1. School of Computer Science and Technology, Xi'an Jiaotong University, Xi'an 710049, China;
2. Network Information Center, Xi'an Jiaotong University, Xi'an 710049, China;
3. School of Software, Xi'an Jiaotong University, Xi'an 710049, China)

Abstract: To address the issues of inefficient performance analysis and difficulty in tuning caused by the scarcity of input/output (I/O) data and the prohibitive costs of modeling and optimization in large-scale parallel applications, a novel method for machine learning-driven I/O performance modeling and prediction in large-scale parallel applications is proposed. During the modeling

收稿日期：2025-12-10。 作者简介：刘恒(2000—)，男，博士生；王子衡(通信作者)，男，助理教授。 基金项目：国家重点研发计划资助项目(2023YFB3001804)。
网络出版时间：2026-03-25 网络出版地址：https://link.cnki.net/urlid/61.1069.T.20260325.1253.002

phase, on the one hand, a linear regression-based model for I/O feature prediction in parallel applications is constructed using data sampled from small-scale node environments to enable the extrapolation of features to large-scale configurations; on the other hand, I/O features in parallel applications acquired from small-scale node environments, along with the corresponding I/O stack parameter space, are fed into an artificial neural network (ANN) for training. This allows for the learning of the nonlinear mapping between system configuration parameters and I/O performance, leading to the construction of a model for I/O performance prediction in parallel applications. During the prediction phase, the I/O features of large-scale parallel applications are first predicted via the I/O feature prediction model. These predicted features, together with the corresponding I/O stack parameter space of large-scale parallel applications, are then fed into the I/O performance prediction model to accurately predict the I/O performance of such applications. Experimental results demonstrate that, on a domestic supercomputer, the proposed method exhibits high prediction accuracy across four benchmark applications: IOR, S3D-IO, BT-IO, and Flash-IO. When models trained on 1—16 nodes are extrapolated to a 128-node (2 048-process) scenario, the mean absolute percentage errors (MAPEs) are 19.07%, 18.96%, 12.34%, and 14.16%, respectively. Furthermore, by tuning the I/O stack parameters based on the proposed model, I/O performance speedups of 16.38, 23.16, 45.36, and 65.38-fold are achieved for the four benchmark applications.

Keywords: large-scale parallel applications; I/O performance modeling; I/O auto-tuning; artificial neural network

随着高性能计算(HPC)领域科学计算应用解决问题的粒度更细,规模更大,求解数据规模也急剧增长[1],常常达到 GB 乃至 PB 级别[2]。尽管计算能力在过去几十年中取得了显著提升,但输入/输出(I/O)性能的发展却远远滞后,往往成为大规模科学计算工作流中的主要瓶颈[3]。在这一背景下,深入理解并有效预测 HPC 系统中的 I/O 性能,对优化应用程序性能、提升系统资源利用率具有重要意义。通过性能预测,可以提前评估作业在不同配置下的表现,为合理的任务调度、数据布局以及存储资源分配提供依据,从而提升整体吞吐能力和系统效率,同时为计算中心在系统采购与资源配置中提供决策支持。

部署在大型 HPC 系统上的 I/O 堆栈非常复杂,并行 I/O 栈[4]由高级 I/O 库、中间层 I/O 库、底层 I/O 库和并行文件系统组成[5],具有多个调优参数,旨在提高应用程序的数据吞吐量和整体执行效率。例如消息传递接口中的并行输入/输出扩展规范(MPI-IO)[6]中聚合器数量和缓冲区大小,以及底层 Lustre 文件系统[7-8]的条带数和条带大小。然而,如何在庞大且复杂的参数空间中选择合适的配置,成为 I/O 性能优化中的核心难题。一方面,参数组合数量庞大且存在强耦合关系,传统的穷举或经验调优方法难以奏效[9]。另一方面,不同应用的 I/O 行为差异显著,导致同一参数配置在不同规模或场景下可能表现出完全不同的性能。此外,随着节点规模扩展至上百甚至上千级别,实验测试与参数搜索的成本急剧上升,进一步限制了大规模并行程序性能建模和调优的可行性与效率。

现有研究主要分为两大类方向:一类是基于启发式搜索的 I/O 参数优化方法[10-11];另一类是基于数据驱动的 I/O 建模与性能调优方法[12-13]。前者通常以最小化 I/O 时间或最大化吞吐带宽为优化目标,通过多轮迭代在庞大的参数空间中进行全局或局部搜索,以获得近似最优的参数组合;而后者则基于已有的性能观测数据,利用机器学习或统计建模手段刻画 I/O 性能与系统参数之间的映射关系,从而在无需大量实验运行的情况下实现性能预测与配置优化[14]。

在启发式搜索方面,Behzad 等[15]利用遗传算法在 I/O 参数空间中探索最优组合,并将所得配置直接注入应用程序以改善性能;Bagbaba 等[16]同样基于遗传算法,从 I/O 堆栈的不同层次联合调优参数,以全局视角提升整体 I/O 效率。然而,这类方法往往需要大量实验运行来完成搜索与评估,计算和时间开销极高,且当节点规模扩展至数百甚至上千时,其探索代价会迅速增长,从而限制了在大规模系统中的实际应用。

在数据驱动方法方面，研究者们更多关注对系统性能波动和复杂性的建模。Nemirovsky 等[17]通过采集 HPC 系统的 I/O 数据并监控不同存储子系统的运行状态，将性能预测问题转化为分类任务，利用支持向量机预测未来 I/O 操作可能遭遇的性能状态。Kim 等[18]则从日志文件中提取 I/O 数据，利用多种特征选择算法与评分函数筛选关键特征，随后采用回归模型进行性能预测。Liu 等[19]进一步整合多种预测算法，提出基于集成学习的 I/O 调优框架，以提升整体预测与优化效果。然而，这类数据驱动方法通常依赖于 Meswani 等[20] I/O 分析工具进行数据采集，需要在大规模系统上持续记录应用运行轨迹，导致部署与运行开销较高。此外，现有模型多局限于训练数据范围内的性能拟合，难以实现跨规模、跨配置的泛化预测，尚未有效解决如何由小规模节点数据推断大规模节点 I/O 性能的问题。

针对上述不足，本文提出了一种采用机器学习的大规模并行程序 I/O 性能建模与预测方法(LMP-IO)。该方法基于人工神经网络(ANN)构建非线性映射模型，通过学习系统配置参数与 I/O 性能指标之间的内在关系，实现对大规模节点性能的外推预测。该研究突破了传统方法建模范围受限的局限，为大规模 HPC 系统的并行程序 I/O 性能分析与参数调优提供了一种高效、低成本的新途径。

1　背景和动机

在 HPC 应用程序中，实现大规模 I/O 性能建模与优化是一项极具挑战的任务。现有的多数性能建模与调优方法通常需要针对不同程序分别构建独立的数据集，而数据集的生成与采集过程往往极为耗时且代价高昂。例如 Liu 等[19]采用 XGBoost 对 IOR 基准程序进行建模，其中用于模型训练的数据为 14 000 条，支持的最大进程数为 128。Wang 等[21]采用多种机器学习模型对 IOR 基准程序进行建模，其中用于模型训练的数据为 9 000 条，支持的最大进程数为 192。Bagbaba 等[16]使用随机森林对 IOR 等程序构建模型，其中用于模型训练的数据为 2 153 条，支持的最大进程数为 1 200。尽管上述研究在一定程度上提升了 I/O 性能预测的精度，但其数据采集与建模成本依然极高，尤其是在大规模节点系统中，随着节点数与进程数的增加，参数空间维度呈指数级扩张，样本生成与实验运行的开销急剧上升，导致应用程序 I/O 性能建模与调优在实际环境中几乎难以实现。

因此，本文关注于一种更具可扩展性的建模思路：在仅给定小规模节点下有限样本数据的条件下，能够准确预测更大规模节点配置下的并行程序 I/O 性能，并进一步寻找最优参数组合以最小化 I/O 延迟。形式化地，可以将该问题表示为以下优化问题

$$\min_{p\in P} f(p, N_{\text{test}}) \tag{1}$$

式中：$f(p, N_{\text{test}})$ 表示节点数为 N 时，参数配置 p 下的应用程序 I/O 时间；P 为参数组合，包含节点数、每个节点的进程数以及其他可调 I/O 参数，也是一个参数化的向量。该优化问题的目标是在保证预测精度的前提下，利用小规模采样数据对大规模应用程序的 I/O 性能进行建模与调优，从而实现性能与成本的平衡。

在大规模科学计算应用中，I/O 写性能通常是决定整体运行效率的核心瓶颈之一。相比于读操作，写操作更容易受到系统带宽限制、元数据开销、并发访问冲突以及缓存一致性策略等多因素的共同制约，因此在高并发场景下更容易出现性能退化。此外，科学计算任务需要在运行过程中周期性地写入检查点(check point)以确保容错性和可恢复性。这使得写操作在执行频率和数据规模上都显著高于读操作，从而成为大规模并行系统中最关键、最昂贵的 I/O 环节。因此，本文重点关注科学计算应用的 I/O 写性能建模与调优，通过利用小规模节点的有限样本实现对大规模场景的外推预测，旨在解决写入性能难以建模、测量成本高以及配置空间巨大的问题。尽管本文实验聚焦于写操作，但提出的建模框架具有普适性，同样适用于读操作的性能行为分析和参数调优。

2　I/O 性能建模与调优方法

在大规模 HPC 系统中，I/O 栈参数空间具有高度的非线性和层次性，不同节点规模下的参数约束条件存在显著差异。例如，聚合器数量参数受到进程总数的限制，在小规模节点下取值范围相对较小，而在大规模节点下则会随系统规模扩展而增大。换言之，大规模节点的 I/O 栈参数空间远大于小规模节点的参数空间，参数间的耦合关系也更加复杂。因此，小规模节点下获得的最优 I/O 参数配置在大规模节点下并不一定保持最优性能，甚至可能导致 I/O 性能下降。

为解决这一问题，本文提出了一种采用机器学习的大规模并行程序 I/O 性能建模与预测方法——LMP-IO，该方法的框架如图 1 所示。在训

练阶段，首先在小规模节点下利用 Darshan 等性能采集工具，获取应用运行时的 I/O 操作信息；随后，对采样数据进行清洗与筛选，并依据负载随节点数变化的模式，将样本划分为强扩展、弱扩展和混合建模 3 类建模场景，以更准确地刻画不同并行特性下的 I/O 行为差异；接着，通过线性回归模型对小规模节点下的采样数据进行初步建模，构建可用于大规模节点外推的并行程序 I/O 特征预测模型，最后，将提取的并行程序 I/O 特征与小规模节点下的 I/O 栈参数空间一同输入至 ANN 模型中进行训练，以学习系统配置与 I/O 性能之间的非线性映射关系，从而得到 I/O 性能预测模型，并用于最终的大规模节点 I/O 性能外推。

在预测与调优阶段，根据目标程序的配置参数，首先利用已训练的 I/O 特征预测模型预测其在大规模节点下的 I/O 特征；随后将预测得到的 I/O 特征与大规模节点的 I/O 栈参数组合输入 I/O 性能预测模型，获得不同配置下的 I/O 性能预测结果；最后通过对预测的 I/O 时间进行排序，系统自动筛选出性能最优的参数组合。

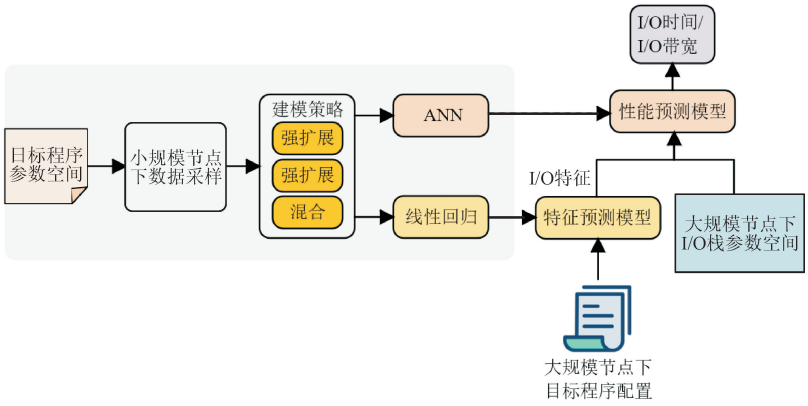

图 1 LMP-IO 方法结构图

Fig. 1 Structure of the LMP-IO method

2.1 数据采样与筛选

在数据采样阶段，参数选择的合理性对预测模型的最终性能起着至关重要的作用。本文所采用的应用程序参数和系统参数及取值范围如表 1 所示。

表 1 本文所采用的参数定义及取值范围

Table 1 Parameter definitions and value ranges used in this study

参数	取值范围
每个进程连续 I/O 字节数	10^6～10^9
MPI-IO 总数据量	10^6 10^9
节点数	1～128
每节点进程数	16
I/O 迭代次数	1～20
聚合器数	节点数大于等于 2^n
条带数	1，2，4，8，16，32，64，96，120
条带大小/MB	1，4，8，16

表 1 中第一类参数用于表征应用程序自身的 I/O负载特性与数据访问模式，反映了程序在运行过程中 I/O 请求规模与访问粒度的差异。本文选取节点数（nodes）、每节点进程数（nprocs）、I/O 迭代次数（segment）、MPI-IO 总数据量（mpiio total bytes）以及每个进程连续 I/O 字节数（block size）作为代表性指标，以描述应用在不同并行规模下的 I/O强度与访问行为。这些参数能够有效揭示计算规模变化对应用 I/O 负载分布的影响，为后续外推模型的构建与预测提供重要支撑。

对于小规模节点，第一类参数可通过 Darshan 等性能采集工具自动提取，能够较为全面地刻画应用的实际 I/O 行为。然而，在大规模节点场景下，I/O 数据采集与存储成本显著增加，不仅带来额外系统开销，还可能干扰程序正常运行。为此，本文在初步采样基础上引入数据筛选机制，依据并行负载变化规律将样本划分为强扩展和弱扩展两类场景，并分别建模。在强扩展场景下，总 I/O 负载保持不变，MPI-IO 总数据量可视为常量，因此重点建模每个进程连续 I/O 字节数；在弱扩展场景下，每个进程负载保持固定，每个进程连续 I/O 字节数基本不变，因此重点预测系统整体的 MPI-IO 总数据量。由于上述特征均与节点规模呈近似线性关系，本文采用线性回归进行拟合，从而在降低大规模数据采集开销的同时，实现对大规模节点 I/O 特征的有效外推。

第二类参数来自 I/O 堆栈中的可调参数，如聚合器数(cb nodes)、条带数(striping factor)和条带大小(striping unit)，这些参数分布在不同层次中，对 I/O 性能有着直接影响，是进行性能建模与调优的关键依据。

本文观察到单一参数独立变化时，I/O 时间和参数呈现近似线性的关系，多个参数共同变化时会呈现出非线性特征，因此线性模型无法对其进行预测，ANN 模型中的线性权重以及隐藏层结构能够同时捕捉这种线性与非线性的混合效应。因此，本文选择 ANN 模型构建不同并行程序的 I/O 性能预测模型。

2.2　ANN 模型设置

本文使用的 ANN 模型是基于误差反向传播算法训练的前馈神经网络模型，它主要由若干隐含层以及输出层组成。输入层用于接收系统配置参数与并行程序的 I/O 行为特征，经标准化处理后传递至隐含层。隐含层通过非线性激活函数对输入数据进行特征提取与映射，从而逐步捕获特征之间复杂的非线性关联关系。ANN 模型的结构如图 2 所示。

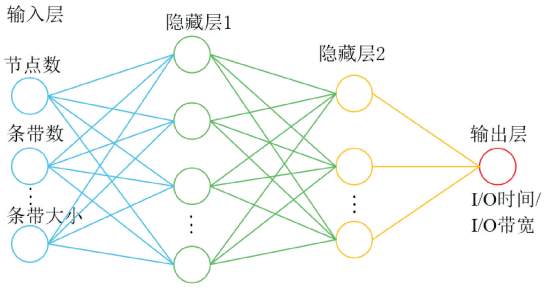

图 2　ANN 模型结构示意图

Fig. 2　Schematic diagram of the ANN model structure

在 ANN 中，神经元由权重 w、偏置 b 和激活函数组成，用于实现输入与输出之间的线性与非线性映射，其中权重反映了各输入特征对输出结果的影响程度，偏置用于调节神经元的激活阈值，激活函数则负责进行非线性运算，公式如下

$$\boldsymbol{x}_{i+1} = \boldsymbol{f}(w_i \boldsymbol{x}_i^{\mathrm{T}} + \boldsymbol{b}_i) \tag{2}$$

式中：$\boldsymbol{x}_i$ 为第 i 层网络的输出矩阵，$i = 1,2,3,4$，其中 $\boldsymbol{x}_4$ 为输出果(I/O 时间或 I/O 带宽)；$\boldsymbol{f}(\boldsymbol{x})$ 为逐元素作用的激活函数；$\boldsymbol{b}_i$ 为偏置向量，其在批量计算时扩展为偏置矩阵。由于模型本身的构造并非本文研究重点，因此在模型结构上采取了 3 层隐藏层并且 ReLU 函数作为激活函数，以便快速验证 ANN 模型在该问题中的适用性。

由于 MPI-IO 总数据量、每个进程连续 I/O 字节数等写入相关参数通常分布在 $10^6 \sim 10^9$ 量级之间，数值跨度较大，而聚合器数、条带数等系统级参数则呈倍数级变化。若直接将这些参数输入模型，容易导致特征分布不均衡和梯度更新不稳定，从而影响模型的收敛速度与预测精度。为此，本文在预处理阶段对上述参数统一进行对数变换，并结合 Min-Max 归一化压缩其分布范围，以提高模型训练的稳定性与收敛性。同时，I/O 时间同样具有跨数量级变化特征。为保持输入与输出在数值尺度上的协调性，本文在训练阶段对 I/O 时间进行对数变换，并在推理阶段通过指数变换恢复至原始尺度，以保证预测结果的物理可解释性与可比性。

2.3　数据集构建及模型训练

选择一个 I/O 基准程序、3 个典型应用I/O内核来评估本文所提方法的有效性，不同的设置代表不同的 I/O 模式。IOR 是高性能计算领域应用最广泛的 I/O 基准测试工具[22]，能够灵活模拟并行应用在 HPC 环境中的文件读写行为。典型应用I/O内核是从实际应用程序中提取的 I/O 部分，反映应用在典型应用场景下的 I/O 行为。S3D-IO 是燃烧模拟中典型的计算内核[23]；BT-IO 是一个专门用于评估并行 I/O 性能的典型 I/O 内核[24]；Flash-IO 是一种设计用于解决可压缩反应流问题开源代码的 I/O 内核[25]。上述内核均具有较强的代表性。

表 2 列出了 4 个 I/O 测试程序的节点规模、文件大小范围及数据集规模，其中文件大小表征程序在不同输入条件下的数据量。受实验平台最大节点规模限制，本文采用 1、2、4、8 和 16 节点样本进行训练，并对 128 节点配置开展外推预测。除特殊说明外，各应用均默认每节点运行 16 个进程。

表 2　测试程序及数据集配置

Table 2　Configurations of benchmark programs and datasets

测试程序	建模方式	文件大小/GB	训练集数据量	测试集数据量
IOR	强扩展	4	429	268
	弱扩展	0.03～4	535	268
S3D-IO	强扩展	10	556	189
	弱扩展	0.066～10	529	189
BT-IO	强扩展	6.25	321	243
	弱扩展	0.1～6.25	322	243
Flash-IO	弱扩展	1.12～143.82	536	273

需要说明的是：BT-IO 仅支持平方数进程配置，因此将每节点进程数固定为 4，以获取 1、4、16 和 64 节点下的数据集；Flash-IO 则受程序结构限制，仅支持弱扩展实验场景。

本文以 IOR 基准测试工具为例说明强扩展、弱扩展与混合建模在数据构造上的差异。对于强扩展建模，各节点规模下的总负载均固定为 4 GB；对于弱扩展建模，负载随节点数线性增长，以保持单进程负载不变，文件大小范围为 0.03～4 GB，例如节点 1 时负载为0.03 GB、节点 2 时为 0.06 GB、节点 16 时负载为0.50 GB，负载在节点 128 时达到 4 GB；混合建模则在 1～16 节点范围内同时包含强扩展与弱扩展样本，以增强模型对不同负载模式的泛化能力。3 种建模方式的最终目标均为预测 4 GB 负载下 128 节点(2 048 进程)的 I/O 性能。

接下来分别针对这 3 类建模场景构建并训练对应的对应的 I/O 特征预测模型和 I/O 性能预测模型。首先，对数据集进行预处理与随机打乱，以避免样本分布的顺序性偏差。随后，将小规模节点下的数据划分为训练集，用于模型参数学习；将大规模节点下的数据划分为测试集，用于验证模型的外推性能。在训练阶段，线性回归模型作为 I/O 特征预测模型，用于拟合特征随节点规模变化所表现出的近似线性规律。ANN 模型作为性能预测模型，用于学习系统配置、并行程序 I/O 特征与最终 I/O 性能之间潜在的非线性映射关系。ANN 模型的训练迭代次数设为 100 次，学习率为 0.001，以保证模型的收敛稳定性与预测精度。

2.4 实验环境和评价指标

所有实验均在天津国家超级计算中心的天河三号平台上进行测试与验证。该平台每个计算节点包含 16 个计算核心，实验所使用的最大规模为 2 048 个核心。存储系统采用 Lustre2.15.0 并行文件系统，总存储容量为 9.7 PB，单用户最大可用空间为 1.5 TB，并配置 120 个 OST 以支持高并发数据存储与访问。软件环境方面，实验在 Ubuntu20.04.2 LTS 操作系统下开展，采用 gcc 9.3.0 作为编译器，并使用 MPICH 4.0.2 支持并行程序运行。

由于 MPI-IO 总数据量和每个进程连续 I/O 字节数在相同节点规模下基本不变，I/O 特征预测主要关注预测结果的相对误差，因此本文采用平均绝对百分比误差(E_{MAPE})进行评价。对于受系统负载及参数配置等多因素影响、波动性和复杂性更强的 I/O 时间预测任务，进一步引入均方根误差(E_{RMSE})、平均绝对误差(E_{MAE})和决定系数(R^2)，以更全面地衡量模型的预测精度、稳定性及趋势拟合能力。

3 实验结果

3.1 大规模节点下并行程序 I/O 特征预测

在完成模型训练与参数确定后，本节首先对大规模节点下的并行程序 I/O 特征进行预测。该阶段的目标是基于小规模节点(1～16)的采样数据，推断更大规模节点(如 128)下的关键 I/O 特征，包括 MPI-IO 总数据量和每个进程连续 I/O 字节数等指标，从而验证模型在高并行场景下的外推能力与泛化性能。

针对 IOR 和 BT-IO 这类规则 I/O 应用，由于输入参数中已显式包含每个进程连续 I/O 字节数和I/O 迭代次数，而 MPI-IO 总数据量又可由二者与总进程数直接计算得到，因此此类应用的 3 个I/O特征均可由参数关系确定，无需额外进行特征建模。

对于 S3D-IO、Flash-IO 等无法通过程序参数直接推导关键 I/O 特征的应用，本文进一步采用线性回归模型进行预测。预测结果表明：S3D-IO 在弱扩展和强扩展场景下的 E_{MAPE} 分别为 $6.01\times10^{-4}\%$ 和 $1.06\times10^{-12}\%$；BT-IO 在弱扩展场景下的 E_{MAPE} 为 $1.06\times10^{-12}\%$；Flash-IO 在弱扩展场景下的 E_{MAPE} 为 0.042%，表明本文方法能够较准确地刻画上述不同 I/O 程序在跨规模扩展过程中的 I/O 特征变化规律。

3.2 大规模节点下并行程序 I/O 性能预测

在完成并行程序 I/O 特征的线性预测后，本文进一步对大规模节点下的并行程序 I/O 时间进行外推预测，以验证所提方法在复杂非线性场景下的建模能力和泛化性能。本节选取多种已有的 I/O 性能建模方法进行对比分析，包括：基于随机森林回归的 AIO 方法[21]、基于 K 近邻回归的 DIPL 方法[18]，以及采用 XGBoost 算法进行性能建模的 OPRAE 方法[19]。

在默认参数配置下，IOR 的负载随节点数增加而同步增长，因此表现出弱扩展特征。为构造强扩展预测任务，本文选取节点 128 下每个进程连续 I/O字节数为 2 MB、I/O 迭代次数为 1 的配置作为目标场景。在强扩展条件下，总 I/O 负载需保持恒定，因此每个进程连续 I/O 字节数应随节点数增加按反比缩减。由此，节点 128 下 2 MB 的配置等效于节点 1 下 256 MB，从而保证不同节点规模下的总 I/O数据量一致。若在节点 128 下设置更大的每个进程连续 I/O 字节数，则总负载将超出小规模节点

可对应的样本范围，因而难以构造满足强扩展约束的训练数据。

如表 3 所示，对于 IOR 基准程序，本文方法在 3 种建模方式下均取得了最优的相对误差表现，强扩展、弱扩展和混合建模的 E_{MAPE} 分别为 19.07%、37.26% 和 19.17%。在强扩展场景中，尽管其余 3 个指标相较 AIO 和 OPRAE 方法略有差距，但 E_{MAE}、E_{RMSE} 和 R^2 仍分别达到 1.30、2.22 和 0.843，说明本文方法仍具有较好的预测精度与稳定性。

表 3　IOR 基准程序上 4 种方法的性能对比

Table 3　Performance comparison of 4 methods on the IOR benchmark program

建模方式	方法	E_{MAE}	E_{MAPE}/%	E_{RMSE}	R^2
强扩展	本文	1.30	**19.07**	2.22	0.843
	AIO	**0.91**	19.16	**1.42**	**0.935**
	DIPL	1.51	23.57	2.58	0.788
	OPRAE	0.94	20.30	1.47	0.931
弱扩展	本文	**2.13**	**37.26**	**3.30**	**0.654**
	AIO	5.63	71.89	7.57	−0.820
	DIPL	5.69	72.76	7.68	−0.876
	OPRAE	5.64	74.01	7.56	−0.813
混合	本文	**1.47**	**19.17**	**2.49**	**0.803**
	AIO	4.27	53.44	5.83	−0.080
	DIPL	4.12	50.01	5.80	−0.070
	OPRAE	5.63	72.78	−0.07	−0.815

注：黑体数据为较优值。

图 3 展示了本文方法对 IOR 基准程序在强扩展建模下 128 节点(2 048 进程)的 I/O 时间散点分布图。从图中可以观察到，预测值与实际值整体分布较为集中，大部分样本点紧邻理想预测线，表明模型能够准确刻画 I/O 时间随节点规模变化的非线性行为。

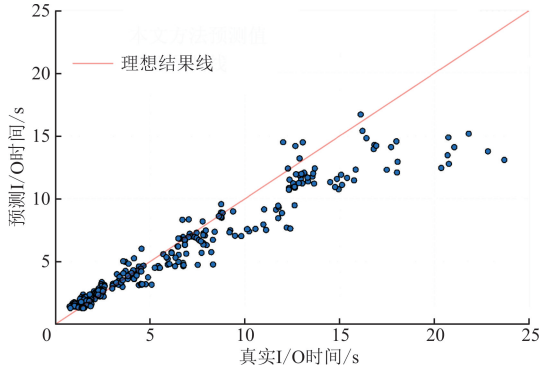

图 3　本文方法在 128 节点(2 048 进程)下对 IOR 基准程序的 I/O 时间散点图

Fig. 3　I/O time scatter plot of the IOR benchmark program using the proposed method at 128 nodes(2 048 process)

S3D-IO 与 BT-IO 均通过设置三维网格来控制程序负载。当参数固定时，负载随节点数增加而分摊，因此两者在默认情况下属于强扩展环境。本文将两者的网格规模在 128 节点下均设置为[256，256，256]，并以此配置作为目标预测任务。

如表 4 和表 5 所示，对于 S3D-IO 和 BT-IO 2 个典型应用 I/O 内核，本文方法在 3 种建模方式下的 4 项评价指标均优于对比方法。其中，在 S3D-IO 程序弱扩展建模场景下取得了更低的 E_{MAPE}，而在 BT-IO 程序混合建模场景下表现最优。

表 4　S3D-IO 程序上 4 种方法的性能对比

Table 4　Performance comparison of 4 methods on the S3D-IO program

建模方式	方法	E_{MAE}	E_{MAPE}/%	E_{RMSE}	R^2
强扩展	本文	**8.27**	**19.06**	**13.41**	**0.937**
	AIO	22.32	40.78	34.12	0.589
	DIPL	22.56	37.31	37.82	0.496
	OPRAE	22.35	41.73	34.11	0.590
弱扩展	本文	**8.52**	**18.96**	**13.62**	**0.934**
	AIO	43.25	49.14	65.38	−0.506
	DIPL	43.16	78.75	65.76	−52.330
	OPRAE	43.09	78.80	65.33	−0.503
混合	本文	**9.35**	**22.96**	**15.16**	**0.919**
	AIO	37.09	64.65	56.80	−0.136
	DIPL	43.22	78.65	65.86	−0.528
	OPRAE	22.35	41.73	65.35	−0.504

注：黑体数据为较优值。

表 5　BT-IO 程序上 4 种方法的性能对比

Table 5　Performance comparison of 4 methods on the BT-IO program

建模方式	方法	E_{MAE}	E_{MAPE}/%	E_{RMSE}	R^2
强扩展	本文	**6.05**	**21.46**	**11.98**	**0.743**
	AIO	10.95	34.21	19.84	0.297
	DIPL	11.52	34.12	21.50	0.174
	OPRAE	10.88	34.27	19.70	0.306
弱扩展	本文	**3.98**	**15.93**	**7.15**	**0.908**
	AIO	17.79	69.27	26.40	−0.245
	DIPL	18.03	68.85	27.27	−0.328
	OPRAE	17.72	68.85	26.28	−0.233
混合	本文	**2.46**	**12.34**	**4.81**	**0.959**
	AIO	14.89	53.09	23.39	0.022
	DIPL	14.49	49.99	23.36	0.025
	OPRAE	10.87	33.97	19.71	0.306

注：黑体数据为较优值。

图 4 和图 5 给出了本文方法在 2 个应用程序、128 节点下最低 E_{MAPE} 对应的 I/O 时间散点分布。由图可见：本文方法在 S3D-IO 程序上，对于 I/O 时间小的区域接近理想预测线，在 I/O 时间较大的区域出现一定偏差，但整体趋势仍与理想线保持一致；在 BT-IO 程序上预测值与实际值整体分布较为集中，紧邻理想预测线。

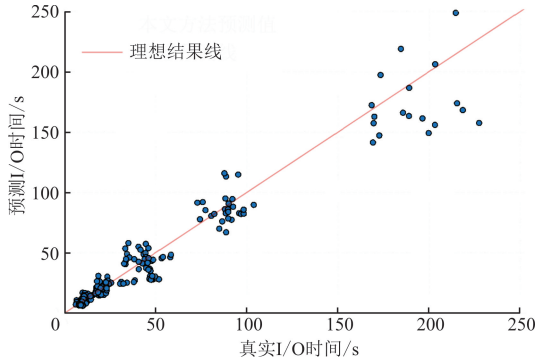

图 4 本文方法在 128 节点(2 048 进程)下对 S3D-IO 程序的 I/O 时间散点图

Fig. 4 I/O time scatter plot of the S3D-IO program using the proposed method at 128 nodes(2 048 processes)

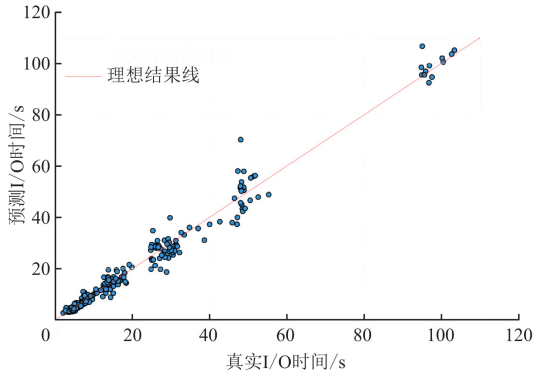

图 5 本文方法在 128 节点(2 048 进程)下对 BT-IO 程序的 I/O 时间散点图

Fig. 5 I/O time scatter plot of the BT-IO program using the proposed method at 128 nodes(2 048 process)

Flash-IO 的运行方式与 S3D-IO 和 BT-IO 存在本质差异，在默认情况下属于弱扩展环境。尽管 Flash-IO 的块大小可以调节，但受限于内存开销，其块尺寸在实际应用中通常无法设置得过大。因此，无法通过缩小局部负载来构造类似 S3D-IO 和 BT-IO 的强扩展环境。

如表 6 所示，对于 Flash-IO 典型应用 I/O 内核，本文方法的 E_{MAPE} 为 14.16%，整体性能完全优于对比方法。图 6 展示了本文方法在 128 节点(2 048)下对 Flash-IO 程序的 I/O 时间散点图。由图中散点的聚集程度可以看出，当 I/O 时间较短时，预测点与真实值高度一致。

表 6 Flash-IO 程序上 4 种方法的性能对比

Table 6 Performance comparison of 4 methods on the Flash-IO program

建模方式	方法	E_{MAE}	E_{MAPE}/%	E_{RMSE}	R^2
强扩展	本文	**52.49**	**14.16**	**98.56**	**0.896**
	AIO	297.55	83.11	406.65	−0.769
	DIPL	300.25	83.10	413.02	−0.825
	OPRAE	297.43	83.03	406.59	−0.824

注：黑体数据为较优值。

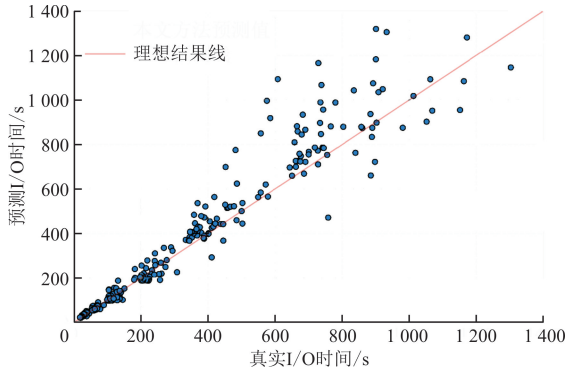

图 6 本文方法在 128 节点(2 048 进程)下对 Flash-IO 程序的 I/O 时间散点图

Fig. 6 I/O time scatter plot of the Flash-IO program using the proposed method at 128 nodes(2 048 process)

从 4 个典型测试程序的结果来看，在强扩展建模场景下，由于问题规模固定且训练与测试样本分布差异较小，对比方法均能取得一定的预测效果。而在弱扩展场景中，节点负载随规模增加而显著提升，I/O 时间呈现明显的分布外增长特征，使得依赖局部插值或分段近似的对比方法难以有效外推。相比之下，本文方法依托 ANN 模型的连续映射能力及其对整体变化趋势的拟合优势，能够更准确地刻画I/O性能随规模扩展的非线性变化规律，因此在弱扩展场景下表现出更高的预测精度与稳定性。对于混合扩展建模场景，由于节点规模与单节点负载同时变化，I/O 性能呈现更复杂的非线性特征，对比方法整体泛化能力不足，而本文方法能够较好保持不同扩展方向下 I/O 性能映射关系的一致性，因此仍具有较强的泛化能力与稳定性。

3.3 扩展实验

本文以 IOR 基准程序为例，在强扩展建模场景下设计了 2 组扩展实验，以全面评估模型在不同数据量与负载变化下的性能表现。

第 1 组实验用于评估不同训练样本规模对模型外推性能的影响，设置了 3 组递进式训练集：训练集 1 包含 1、2、4、8 和 16 节点数据，训练集 2 在此基础上增加至 32 节点数据，训练集 3 进一步增加至 64 节点数据。

第 2 组实验用于评估不同训练负载组合对模型外推性能的影响，在训练节点范围固定为 1～16 的条件下，设置了 3 组递进式训练负载，负载组 1 为 4 GB，负载组 2 为 4、2 GB，负载组 3 为 1、2、4 GB。

从表 7 可知，随着训练节点范围的扩大，各项指标均得到显著改善，其中 E_{MAPE} 由 19.07% 降至 14.23%，说明更丰富的训练样本有助于模型更充分地学习 I/O 参数与性能间的非线性关系，从而提升外推精度。

从表 8 可见，在保持训练节点范围不变的情况下，随着负载多样性的增加，模型的预测精度同样持续提升，E_{MAPE} 从 19.07% 降至 13.59%，表明负载多样性对模型泛化能力的增强具有积极作用。

表 7　IOR 基准程序中不同训练样本规模下对节点 128(2 048 进程)的预测结果

Table 7　Prediction results on 128-node (2 048 processes) IOR benchmark program using different training sample scales

评价指标	训练集 1	训练集 2	训练集 3
E_{MAE}	1.30	0.92	0.86
E_{MAPE}/%	19.07	15.98	14.23
E_{RMSE}	2.22	1.47	1.26
R^2	0.843	0.931	0.959

表 8　IOR 基准程序中不同负载下对节点 128(2 048 进程)的预测结果

Table 8　Prediction results on 128-node (2 048 processes) IOR benchmark program under different workloads

评价指标	负载组 1	负载组 2	负载组 3
E_{MAE}	1.30	1.09	0.76
E_{MAPE}/%	19.07	16.22	13.59
E_{RMSE}	0.843	1.720	1.170
R^2	2.220	0.906	0.956

3.4　基于模型性能预测的调优对比

在性能调优阶段，首先需要预设一个由多组可调参数构成的节点 128 I/O 参数空间。基于前述节点 1～16 训练得到的外推模型，对该节点 128 参数空间中每一组参数组合的 I/O 时间进行预测，从而获得对应的性能评估结果。随后，通过比较所有预测值，选取 I/O 时间最短的参数组合作为最终的最优 I/O 参数配置。在本实验中，对各建模方式下的最优参数组合进行了 3 次独立运行，并取 3 次测量的 I/O 时间平均值作为最终结果。

本文设计了 2 组实验以验证所提方法的有效性。第 1 组实验针对 4 个程序在 3 种建模方式下开展调优测试，并以系统默认配置作为参考基线。默认配置中聚合器数、条带数和条带大小分别设为 1、1、1 MB。第 2 组实验选取 3 种已有的基于建模的 I/O 调优方法作为对比，并以 I/O 性能加速比为指标，对不同方法的调优效果进行比较分析。

如表 9 所示，IOR 在弱扩展建模下预测带宽为 4 357.45 MB/s，而实测带宽为 3 338.22 MB/s，存在明显高估；此类高误差样本对整体相对误差具有显著拉升作用，也与表 5 中弱扩展 E_{MAPE} 达到 37.26% 的结果相一致。如表 10 所示，在 128 节点调优场景下，S3D-IO 与 IOR 同样表现出明显的高估趋势。其主要原因在于，训练样本仅覆盖 1～16 节点范围，其中聚合器数的上限受节点规模约束，最大仅为 16，而在 128 节点调优场景下，该参数已进入远超训练范围的取值区间；相比之下，条带数受文件系统上限约束，跨场景变化幅度相对有限。因此，模型在 128 节点场景下面临更强的聚合器数参数外推压力，仍可能沿用小规模节点下“增大聚合节点数通常有利于提升带宽”的主导规律，从而导致 I/O 带宽的预测值高于实测值。

表 9　IOR 基准程序在不同建模方式下对节点 128(2 048 进程)的调优结果

Table 9　Tuning results on 128 nodes (2 048 processes) for the IOR benchmark program under different modeling strategies

建模方式	聚合器数	条带数	条带大小/MB	预测时间/s	实测时间/s	预测带宽/(MB·s^{-1})	实测带宽/(MB·s^{-1})
强扩展	128	120	8	1.28	1.47	3 200.00	2 649.42
弱扩展	128	64	16	0.94	1.23	4 357.45	3 338.22
混合建模	128	120	16	1.07	0.80	3 828.04	5 145.73

表 10 S3D-IO 程序及不同建模方式下对节点 128(2 048 进程)的调优结果

Table 10 Tuning results on 128 nodes (2 048 processes) for the S3D-IO program under different modeling strategies

建模方式	聚合器数	条带数	条带大小/MB	预测时间/s	实测时间/s	预测带宽/(MB·s^{-1})	实测带宽/(MB·s^{-1})
强扩展	128	120	8	6.09	9.78	1 681.44	1 047.46
弱扩展	128	96	1	5.30	7.97	1 932.08	1 285.46
混合建模	128	120	8	6.09	9.78	1 681.44	1 047.46

如表 11 和表 12 所示,BT-IO 和 Flash-IO 的 I/O 时间与带宽预测误差均较小,说明模型能够较好捕捉这 2 类应用的 I/O 性能变化规律,并在不同建模场景下保持较稳定的预测精度。结合表 9 至表 12 的整体结果可以看出,当聚合器数和条带数被设置为与节点规模相匹配的数值时,可有效提升 I/O 请求在节点间与 OST 之间的并行度,使得系统带宽获得更充分利用,从而带来更明显的性能加速效果。

表 11 BT-IO 程序及不同建模方式下对节点 128(2 048 进程)的调优结果

Table 11 Tuning results on 128 nodes (2 048 processes) for the BT-IO program under different modeling strategies

建模方式	聚合器数	条带数	条带大小/MB	预测时间/s	实测时间/s	预测带宽/(MB·s^{-1})	实测带宽/(MB·s^{-1})
强扩展	64	64	4	3.85	3.37	1 662.34	1 897.42
弱扩展	64	32	8	2.67	2.10	2 397.00	3 041.83
混合建模	64	32	8	2.67	2.10	2 397.00	3 041.83

表 12 Flash-IO 程序及不同建模方式下对节点 128(2 048 进程)的调优结果

Table 12 Tuning results on 128 nodes (2 048 processes) for the Flash-IO program under different modeling strategies

建模方式	聚合器数	条带数	条带大小/MB	预测时间/s	实测时间/s	预测带宽/(MB·s^{-1})	实测带宽/(MB·s^{-1})
弱扩展	128	120	4	21.42	19.40	6 874.47	7 591.44

图 7 给出了在 128 节点(2 048 进程)下,本文方法分别采用强扩展建模、弱扩展建模和混合建模 3 种方式时,在 4 个程序上推荐参数配置所获得的 I/O 时间加速比。结果表明,3 种建模策略推荐的参数配置均能够有效降低 I/O 时间,说明本文方法在不同建模方式下均具有较好的参数优化能力。进一步比较不同方式可以发现:弱扩展建模在 S3D-IO、BT-IO 和 Flash-IO 3 个程序上表现更优,分别取得 23.16、45.36 和 65.38 的 I/O 时间加速比,说明弱扩展样本能够更有效地刻画这些程序在 128 节点下的 I/O 行为;混合建模在 IOR 基准程序上取得最高加速比,并在 BT-IO 程序上与弱扩展建模持平,表明融合不同扩展模式的信息在部分程序上也具有优势;相比之下,强扩展建模虽然同样能够带来明显加速,但整体调优收益相对较低。

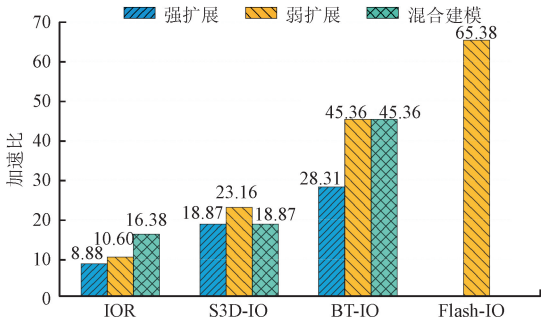

图 7 128 节点(2 048 进程)时本文方法在 3 种建模方式下的 I/O 时间加速比对比

Fig. 7 Comparison of I/O time speedup of the proposed method under three modeling strategies at 128 nodes (2 048 processes)

图 8 比较了本文方法在各程序上采用最优建模方式时,与其他 3 种对比方法所获得的 I/O 时间加速比。结果表明,本文方法在 4 个程序上均取得了更高的加速比,整体优于现有方法,验证了本文所提方法在大规模并行 I/O 参数调优中的有效性。

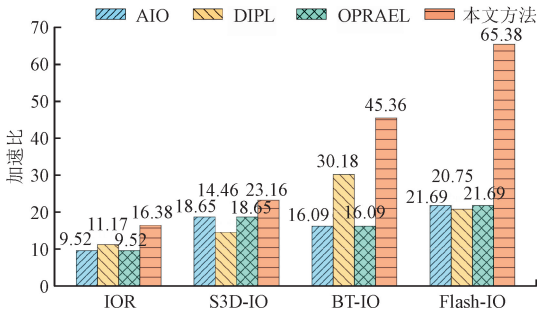

图 8　不同建模方法下对节点 128(2 048 进程)的调优结果

Fig. 8　Tuning results on 128 nodes (2 048 processes) using different modeling methods

4　结　论

本研究针对大规模并行程序在 I/O 性能建模与预测中存在的数据采集成本高和建模效率低的问题，提出了一种采用机器学习的大规模并行程序 I/O性能建模与预测方法。该方法通过在小规模节点下采集有限的性能数据，构建系统配置参数与 I/O性能指标之间的非线性映射模型，从而实现对大规模节点下 I/O 性能的外推预测。实验结果表明，本文方法在 IOR、S3D-IO、BT-IO 和 Flash-IO 4 种典型测试程序上的预测精度均较高，当采用 1～16节点规模下训练得到的模型对 128 节点(2 048 进程)场景进行外推预测时，在 4 种测试程序上的平均绝对百分比误差分别为 19.07%、18.96%、12.34%、14.16%。经过调优后，在 4 种程序的性能加速比分别达到 16.38、23.16、45.36 和 65.38 倍。

参考文献：

[1] Luu H, Winslett M, Gropp W, et al. A multiplatform study of I/O behavior on petascale supercomputers [C]//Proceedings of the 24th International Symposium on High-Performance Parallel and Distributed Computing. New York, USA: ACM, 2015: 33-44.

[2] 张成. 基于回归分析和集成学习的 HPC 应用 I/O 性能优化方法研究 [D]. 西安：西北大学，2023.

[3] 孙经纬. 数据驱动的高性能计算程序执行时间预测与优化研究 [D]. 合肥：中国科学技术大学，2020.

[4] Zhu Zhaobin, Neuwirth S, Lippert T. A comprehensive I/O knowledge cycle for modular and automated HPC workload analysis [C]//2022 IEEE International Conference on Cluster Computing (CLUSTER). Piscataway, NJ, USA: IEEE, 2022: 581-588.

[5] Liu Wei, Wu Kai, Liu Jialin, et al. Performance evaluation and modeling of HPC I/O on non-volatile memory [C]//2017 International Conference on Networking, Architecture, and Storage (NAS). Piscataway, NJ, USA: IEEE, 2017: 1-10.

[6] Neuwirth S, Paul A K. Parallel I/O evaluation techniques and emerging HPC workloads: a perspective [C]//2021 IEEE International Conference on Cluster Computing (CLUSTER). Piscataway, NJ, USA: IEEE, 2021: 671-679.

[7] 汤志航，兰颢，刘政国，等. HiTrain:面向大模型训练的异构内存卸载与 I/O 优化 [J]. 计算机研究与发展，2026，63(3)：627-639.
Tang Zhihang, Lan Hao, Liu Zhengguo, et al. HiTrain: heterogeneous memory offloading and I/O optimization for large language model training [J]. Journal of Computer Research and Development, 2026, 63(3): 627-639.

[8] 程稳，李焱，曾令仿，等. 面向 Lustre 集群存储的应用日志分析及系统自动优化框架 [J]. 计算机工程与科学，2022，44(4)：594-604.
Cheng Wen, Li Yan, Zeng Lingfang, et al. An application log analysis and system automation optimization framework for Lustre cluster storage [J]. Computer Engineering & Science, 2022, 44(4): 594-604.

[9] 张文韬，汪璐，程耀东. 基于强化学习的 Lustre 文件系统的性能调优 [J]. 计算机研究与发展，2019，56(7)：1578-1586.
Zhang Wentao, Wang Lu, Cheng Yaodong. Performance optimization of Lustre file system based on reinforcement learning [J]. Journal of Computer Research and Development, 2019, 56(7): 1578-1586.

[10] 田鸿运，武林平，董勇，等. 面向大规模集群的并行 I/O 用户层配置优化策略 [J]. 国防科技大学学报，2020，42(2)：23-30.
Tian Hongyun, Wu Linping, Dong Yong, et al. User-level parallel I/O configuration optimize strategy toward large-scale cluster [J]. Journal of National University of Defense Technology, 2020, 42(2): 23-30.

[11] Egersdoerfer C, Rashid M H, Dai Dong, et al. Understanding and predicting cross-application I/O interference in HPC storage systems [C]//SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis. Piscataway, NJ, USA: IEEE, 2024: 1330-1339.

[12] Behzad B, Luu H V T, Huchette J, et al. Taming parallel I/O complexity with auto-tuning [C]//Proceedings of the International Conference on High Per-

formance Computing, Networking, Storage and Analysis. New York, USA: ACM, 2013: 68.

[13] Tipu A J S, Conbhuí P Ó, Howley E. Artificial neural networks based predictions towards the auto-tuning and optimization of parallel IO bandwidth in HPC system [J]. Cluster Computing, 2024, 27(1): 71-90.

[14] Nicolas L M, Mimouni S, Couvée P, et al. I/O patterns modeling of HPC applications with call stacks for predictive prefetch [J]. Future Generation Computer Systems, 2026, 175: 108034.

[15] Behzad B, Byna S, Prabhat, et al. Optimizing I/O performance of HPC applications with autotuning [J]. ACM Transactions on Parallel Computing, 2019, 5(4): 15.

[16] Bagbaba A, Wang Xuan. Improving the mpi-io performance of applications with genetic algorithm based auto-tuning [C]//2021 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW). Piscataway, NJ, USA: IEEE, 2021: 798-805.

[17] Nemirovsky D, Arkose T, Markovic N, et al. A machine learning approach for performance prediction and scheduling on heterogeneous CPUs [C]//201729th International Symposium on Computer Architecture and High Performance Computing (SBAC-PAD). Piscataway, NJ, USA: IEEE, 2017: 121-128.

[18] Kim S, Sim A, WU Kesheng, et al. Design and implementation of I/O performance prediction scheme on HPC systems through large-scale log analysis [J]. Journal of Big Data, 2023, 10(1): 65.

[19] Liu Zhangyu, Zhang Cheng, Wu Huijun, et al. Optimizing HPC I/O performance with regression analysis and ensemble learning [C]//2023 IEEE International Conference on Cluster Computing (CLUSTER). Piscataway, NJ, USA: IEEE, 2023:234-246.

[20] Meswani M R, Laurenzano M A, Carrington L, et al. Modeling and predicting disk I/O time of HPC applications [C]//2010 DoD High Performance Computing Modernization Program Users Group Conference. Piscataway, NJ, USA: IEEE, 2010: 478-486.

[21] Wang Wanxin, Wu Huijun, Yang Lihua, et al. AIO: automating I/O optimization pipeline for data-intensive applications in HPC [C]//2024 IEEE International Symposium on Parallel and Distributed Processing with Applications (ISPA). Piscataway, NJ, USA: IEEE, 2024: 1541-1548.

[22] Shan Hongzhang, Shalf J. Using IOR to analyze the I/O performance for HPC platforms [EB/OL]. [2015-10-22]. https://escholarship.org/uc/item/9111c60j.

[23] Mendez S, Rexachs D, Luque E. Analyzing the parallel I/O severity of MPI applications [C]//2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID). Piscataway, NJ, USA: IEEE, 2017: 953-962.

[24] Agarwal M, Singhvi D, Malakar P, et al. Active learning-based automatic tuning and prediction of parallel I/O performance [C]//2019 IEEE/ACM Fourth International Parallel Data Systems Workshop (PDSW). Piscataway, NJ, USA: IEEE, 2019:20-29.

[25] Rajesh N, Bateman K, Bez J L, et al. TunIO: an AI-powered framework for optimizing HPC I/O [C]//2024 IEEE International Parallel and Distributed Processing Symposium (IPDPS). Piscataway, NJ, USA: IEEE, 2024: 494-505.

[26] Chen Si, De Gonzalo S G, Wildani A. Few-shot HPC application runtime prediction [C]//2023 IEEE International Conference on Cluster Computing Workshops (CLUSTER Workshops). Piscataway, NJ, USA: IEEE, 2023: 46-47.

(编辑 刘杨 杜秀杰)

(上接第 96 页)

[32] Liu Zundi, Zhang Yi, Li Wei, et al. Self-promoted fuel pyrolysis under oxygen enrichment enables clean and efficient ammonia combustion [J]. The Innovation Energy, 2024, 1(1): 100006.

[33] Zhang Meng, An Zhenhua, Wei Xutao, et al. Emission analysis of the CH_4/NH_3/air co-firing fuels in a model combustor [J]. Fuel, 2021, 291: 120135.

[34] Stagni A, Cavallotti C, Arunthanayothin S, et al. An experimental, theoretical and kinetic-modeling study of the gas-phase oxidation of ammonia [J]. Reaction Chemistry & Engineering, 2020, 5(4): 696-711.

(编辑 武红江)