Computer Science and Application 计算机科学与应用, 2026, 16(6), 252-267
Published Online June 2026 in Hans. https://www.hanspub.org/journal/csa
https://doi.org/10.12677/csa.2026.166225

Hans汉斯

基于机器学习的MPI仿真性能测试与调优平台

赵英燕，曹群生

南京航空航天大学电子信息工程学院，江苏 南京

收稿日期：2026年5月18日；录用日期：2026年6月22日；发布日期：2026年6月30日

摘 要

工程数值模拟软件的MPI并行性能测试过程中，存在MPI参数调优难、人工测试低效、性能基线更新不及时等痛点，如何提升并行性能测试与调优效率成为并行计算领域研究热点之一。本文基于机器学习技术和XGBoost算法构建MPI参数与性能映射模型，运用惯性权重自适应调整、引入高斯变异策略改进IPSO自动调优算法，实现MPI并行参数自动调优，研发自动化回归测试技术，提出统计分析与异常剔除相结合的性能基线生成方法，最终设计并实现面向MPI并行软件研发的参数调优与自动化性能测试SimPerf平台。实验验证结果表明，该平台可显著降低仿真研发人员的MPI并行开发门槛、提升研发与测试效率，与传统人工方式对比，MPI并行仿真应用的性能测试与调优效率提升70%以上，测试覆盖率提升16%以上。

关键词

MPI并行应用，仿真应用研发，机器学习，性能测试，参数调优，性能基线

Machine Learning-Based MPI Simulation Performance Testing and Tuning Platform

Yingyan Zhao, Qunsheng Cao

College of Electronic and Information Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing Jiangsu

Received: May 18, 2026; accepted: June 22, 2026; published: June 30, 2026

Abstract

During the MPI parallel performance testing of engineering numerical simulation software, several critical challenges exist, including difficult tuning of MPI parameters, low efficiency of manual testing, and untimely updates of performance baselines. Enhancing the efficiency of parallel performance testing and parameter tuning has therefore become a research hotspot in the field of parallel

文章引用: 赵英燕, 曹群生. 基于机器学习的 MPI 仿真性能测试与调优平台[J]. 计算机科学与应用, 2026, 16(6): 252-267. DOI: 10.12677/csa.2026.166225

computing. In this paper, a mapping model between MPI parameters and performance is constructed based on machine learning techniques and the XGBoost algorithm. An improved IPSO automatic tuning algorithm is proposed by adaptively adjusting the inertia weight and introducing a Gaussian mutation strategy to realize automatic tuning of MPI parallel parameters. Automated regression testing technology is developed, and a performance baseline generation method combining statistical analysis and outlier elimination is put forward. Ultimately, the SimPerf platform for parameter tuning and automated performance testing is designed and implemented for the research and development of MPI parallel software. Experimental validation results demonstrate that the proposed platform can significantly lower the MPI parallel development threshold for simulation researchers and improve research-development and testing efficiency. Compared with traditional manual methods, the efficiency of performance testing and tuning for MPI parallel simulation applications is improved by more than 70%, and test coverage is increased by over 16%.

Keywords

MPI Parallel Applications, Simulation Application Development, Machine Learning, Performance Testing, Parameter Tuning, Performance Baseline

1. 引言

随着高性能计算技术在工业仿真、数值模拟等领域深度应用，MPI (Message Passing Interface)并行计算标准通信接口[1]已成为支撑大规模工程仿真与科学计算任务的关键技术。其凭借高效的进程间通信能力，有效突破了单节点计算性能瓶颈，为复杂工业场景下的高精度仿真提供了算力保障。在电磁仿真、计算流体动力学等场景中，MPI 并行应用通过多进程协同计算，可将仿真任务的执行效率提升数十倍甚至上百倍[2]。

当前，工程数值仿真应用研发人员在并行开发与测试调优过程中，面临人工调试效率低下、性能测试与参数调优环节割裂、缺乏一体化工具支撑等痛点，尤其是并行性能回归测试繁琐、性能退化问题难以定位，无法实现研发、测试、调优的闭环联动，导致仿真应用研发进度缓慢、并行性能优化不充分。

目前虽已有一些相关研究成果和工具，但均侧重性能分析与调试，缺乏针对 MPI 仿真应用的自动化参数调优、性能基线生成与回归测试等能力，而这些恰是 MPI 仿真应用性能测试中最耗时费力的环节。

TotalView [3]是一款同时支持源码级和硬件级调试的高级调试工具，可用于多线程和多进程应用程序的调试，提供分析、组织和测试程序的方法。虽可用于调试、分析 MPI 程序，但其偏向于对已发现的性能问题进行细致定位与性能剖析，无法实现性能预测与优化，也不具备自动化回归测试、性能基线构建能力。

Intel VTune™ Profile [4]：面向 Intel 平台的性能采样分析工具，可分析 CPU、缓存、通信瓶颈，但不提供参数配置、性能预测与参数调优等功能。Score-P [5]、Vampir [6]等工具同样侧重于并行程序性能采集与分析环节，不提供预测、调优及测试全流程闭环联动。

MTT (MPI Testing Tool) [7]、OSU Benchmarks [8]、PGMPI (Automatically Verifying Self-Consistent MPI Performance Guidelines) [9]等工具虽可调试分析 MPI 性能，但其主要面向 OpenMPI、MPICH 等底层 MPI 实现而非 MPI 并行仿真应用。

本研究围绕 MPI 工程数值仿真应用研发人员核心需求，重点开展四个方面的技术研究。

首先对 MPI 核心参数的分类、功能及应用场景进行系统梳理，构建完整的 MPI 核心参数谱系，为后续调优算法与平台实现提供基础支撑。

从 MPI 参数配置、应用运行数据中提取能够精准反映参数与性能关联的关键特征，构建标准化特征向量数据，基于机器学习方法构建高精度 MPI 参数与性能映射模型，基于该模型运用改进型粒子群优化算法(Improved Particle Swarm Optimization, IPSO) [10]设计实现 MPI 参数自动调优算法。

设计自动化回归测试流程与机制，实现测试用例的自动生成与测试过程的自动化；提出统计分析与异常剔除相结合的性能基线生成方法，实现性能基线的自动化生成、动态更新与多级异常预警。

最后，采用 B/S 架构、Vue + SpringBoot 开发框架与分层设计思想，设计实现了面向 MPI 并行仿真应用研发的参数调优与自动化性能测试一体化平台 SimPerf (MPI Sim Performance Testing & Tuning Platform)。实现参数性能预测、自动化回归测试、性能优化、性能基线生成与对比的性能测试全流程，提升工程数值仿真应用研发的效率与质量。

2. 核心技术与方法研究

2.1. MPI 核心参数谱系梳理

结合工程数值仿真 MPI 并行应用性能测试实际需求，将 MPI 参数分为进程管理、通信机制、核心绑定、缓冲区与内存、并行环境与运行控制、性能调优六大类。系统梳理每个参数的功能、取值范围、默认值、性能影响规律及应用场景，明确大规模网格计算、多物理场耦合仿真等场景的参数适配要求，构建完整的结构化 MPI 核心参数谱系，为后续调优算法与平台实现提供基础支撑。

2.2. MPI 参数性能映射模型与自动调优算法

MPI 并行应用的运行性能与参数配置密切相关，不同参数组合会直接影响应用的运行效率、资源利用率及稳定性。工程数值仿真应用研发与测试过程中，常需反复调整 MPI 参数以寻找最优配置，这一过程高度依赖经验且耗时费力。本研究基于机器学习技术，通过构建 MPI 参数与性能的映射模型，结合历史运行数据与硬件特征，实现参数组合的自动调优与自动收敛。

2.2.1. MPI 参数性能映射模型构建

基于多维度提取的特征数据，结合机器学习[11]算法，构建 MPI 参数与性能映射模型，实现从参数组合到应用性能的可预测性分析与精准预测。模型构建流程总体上分为数据预处理、模型选型、模型训练、模型优化四个主要步骤。整体流程见图 1 所示：

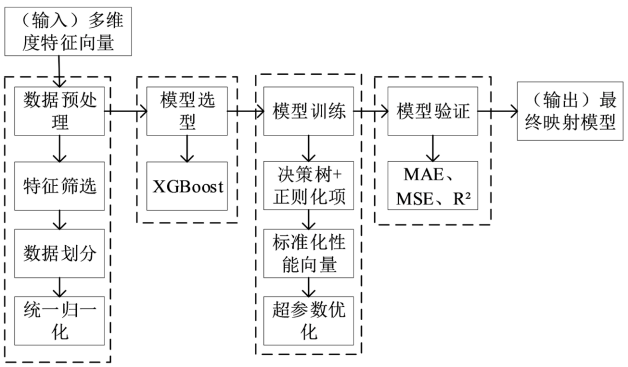

Figure 1. Construction process of MPI parameter and performance mapping model
图 1. MPI 参数与性能映射模型构建流程

(1) 多维度特征提取与特征向量构建

多维度特征提取[12]是构建 MPI 参数与性能映射模型的基础，从 MPI 参数配置、应用运行数据中提取能够精准反映参数与性能关联的关键特征，剔除冗余信息，提升模型的预测精度与泛化能力。结合工程数值仿真 MPI 并行应用的运行特性，特征提取主要围绕参数特征、性能特征、环境特征三个维度展开，最终整合为标准化特征向量，为映射模型构建提供数据支撑。

① 参数特征提取

基于 2.1 节中已梳理的 MPI 核心参数谱系，选取覆盖六大参数分类的关键参数指标作为核心参数特征，针对连续型、离散型、二值型参数采用差异化特征处理方式，同时提取参数间关联特征，丰富特征维度。

连续型参数可在限定范围内任意取值，如进程总数可取[1, Max 进程数]之间的任意整数。采用归一化[13]作为连续型参数的特征处理方式，计算公式为：

$$x_{\text{norm}} = \frac{x + x_{\min}}{x_{\max} - x_{\min}} \tag{1}$$

其中，x_{norm} 为归一化结果值，x 为参数原始值，$x_{\min}$ 为参数取值下限，$x_{\max}$ 为参数取值上限。

离散型参数指只能取限定的若干种取值，如核心绑定粒度可取核心、插槽、节点等值。对于有 n 个取值的离散型参数，使用独热编码[14]方式将其编码为 n 维二进制向量，例如核心绑定粒度为“核心”时，编码为[1, 0, 0]；为“插槽”时，编码为[0, 1, 0]。

二值型参数仅有 2 个互斥的有效值，如“开启核心绑定”仅可取值“关闭(0)”与“开启(1)”。对二值型参数进行特征提取时直接取值即可。

② 性能特征提取

采集 MPI 并行应用运行过程中的核心性能指标作为模型输出特征，主要反映应用的运行效率、资源利用率与通信性能，包括应用运行耗时(s)、CPU 平均利用率(%)、内存占用量(GB)、通信延迟(ms)、算力吞吐量(GFLOPS)、节点间通信量(GB)等。

为确保性能特征的准确性与稳定性，需对采集的性能数据进行清洗处理，剔除异常值、补全缺失值，采用滑动平均法[15]平滑数据波动。

③ 环境特征提取

考虑工程数值仿真场景中运行环境的差异性，提取硬件环境、软件环境、仿真场景特征作为辅助特征融入模型，提升模型在不同场景下的泛化能力。本次共提取 CPU 核心数、内存总容量、网络带宽、节点数、操作系统、编译器、MPI 版本、仿真规模、网格密度、计算任务类型等 10 个核心 MPI 运行环境特征，处理方式与参数特征类似，同样以独热编码和归一化为主。

④ 标准化特征向量构建

整合以上 3 种特征类型，构建模型的输入特征向量 X；选取应用运行耗时、CPU 平均利用率、通信延迟作为核心性能指标，构建模型的输出特征向量 Y。

输入特征向量 $X \in R^n$ (n 为向量总维度)，即

$$X = \left[X_{p1}, X_{p2}, \cdots, X_{pi}, X_{pr1}, X_{pr2}, \cdots, X_{prj}, X_{e1}, X_{e2}, \cdots, X_{ek} \right]^{\mathrm{T}} \tag{2}$$

其中，X_p 为参数特征，X_{pr} 为参数关联特征，X_e 为环境特征。

输出特征向量 $Y \in R^3$，即

$$Y = \left[T, U, D \right]^{\mathrm{T}} \tag{3}$$

其中，T 为应用运行耗时，U 为 CPU 平均利用率，D 为通信延迟，均为归一化后的值。

(2) 数据预处理

对提取的多维度输入特征与输出特征数据进行进一步预处理，提升模型训练效率与预测精度：

① 特征筛选：采用互信息法计算各输入特征与输出特征间的互信息值，筛选互信息值 ＞0.1 的特征，剔除冗余特征，最终保留核心输入特征，降低模型复杂度；

② 数据划分：将预处理后的数据集按一定比例随机划分为训练集、验证集、测试集，分别用于模型参数训练、调整模型超参数、及验证模型的预测性能。本研究采用 7:2:1 的划分比例，兼顾训练规模与验证的全面性；

③ 统一归一化：对所有特征数据采用“最小－最大归一化”处理，将取值映射至[0, 1]区间，确保模型训练的稳定性。

(3) 模型选型与构建

结合 MPI 参数与性能的非线性关联特性，对比线性回归、随机森林、支持向量机(SVM)、梯度提升树(GBDT) [16]等多种机器学习算法的性能，最终选取 XGBoost (Extreme Gradient Boosting)梯度提升树[17]作为基础模型。

XGBoost 模型的输入为预处理后的多维度标准化特征向量，中间经过决策树 + 正则化项处理，最终输出结果为标准化性能向量([T, U, D])。其目标函数由损失函数和正则化项组成：

$$\mathcal{L}(\theta)=\sum_{i=1}^{n} l\left(y_i,\hat{y}_i\right)+\sum_{k=1}^{K}\Omega\left(f_k\right) \tag{4}$$

其中，$l\left(y_i,\hat{y}_i\right)$ 为损失函数，衡量预测值 $\hat{y}_i$ 与真实值 y_i 的误差；$\Omega\left(f_k\right)$ 为正则化项，控制模型复杂度，避免过拟合；K 为决策树的数量。

本次研究选取“均方误差(MSE)”作为损失函数，公式为：

$$l\left(y_i,\hat{y}_i\right)=\left(y_i-\hat{y}_i\right)^2 \tag{5}$$

正则化项公式为：

$$\Omega\left(f_k\right)=\gamma T+\frac{1}{2}\lambda\left\|\omega\right\|^2 \tag{6}$$

其中，T 为决策树的叶子节点数，ω 为叶子节点的权重，γ 为叶子节点数的正则化系数，λ 为权重的正则化系数。

(4) 模型训练与超参数优化

利用训练集数据对 XGBoost 模型进行训练，采用梯度下降法最小化预测值与实际值之间的 MSE 损失函数，通过迭代更新决策树的结构与权重，逐步提升模型的拟合精度；训练过程中采用早停法(Early Stopping) [18]，若验证集的损失值连续 10 轮迭代未下降，则停止训练，避免过拟合。

利用验证集数据，采用“网格搜索法(Grid Search)”[19]对模型的核心超参数进行调优，确定最优超参数组合。

(5) 模型验证

利用测试集数据对训练好的XGBoost映射模型进行验证，计算平均绝对误差(MAE)、均方误差(MSE)、决定系数(R^2)作为评价指标，验证模型的预测效果。评价指标计算公式如下：

平均绝对误差(MAE)：

$$\mathrm{MAE}=\frac{1}{n}\sum_{i=1}^{n}\left|y_i-\hat{y}_i\right| \tag{7}$$

均方误差(MSE)：

$$\mathrm{MSE}=\frac{1}{n}\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2 \tag{8}$$

决定系数(R^2)：

$$R^2=1-\frac{\sum_{i=1}^{n}\left(y_i-\hat{y}_i\right)^2}{\sum_{i=1}^{n}\left(y_i-\hat{y}\right)^2} \tag{9}$$

其中，y_i为性能指标实际值，$\hat{y}_i$为性能指标预测值，$\hat{y}$为所有性能指标实际值的平均值。R^2越接近 1，表明模型预测效果越好。

验证结果表明，构建的 XGBoost 映射模型具有较高的预测精度，能够在不同参数组合、不同运行环境下精准预测应用性能，为后续自动调优算法提供可靠的性能预测支撑。

2.2.2. 自动调优算法设计

基于 MPI 参数与性能映射模型的预测能力，设计自动调优算法，实现 MPI 参数组合的自动搜索与优化，无需研发与测试人员手动调整参数，即可获得最优参数配置，提升应用运行性能。自动调优算法的核心目标是在参数取值范围内，以最小的搜索成本快速搜索到能够使应用性能最优的参数组合。

结合工程数值仿真 MPI 并行应用的参数特性(离散型与连续型混合、参数间存在关联约束)与性能需求(最小化运行耗时、最大化 CPU 利用率、最小化通信延迟)，选取改进型粒子群优化算法(Improved Particle Swarm Optimization, IPSO)作为自动调优核心算法。

(1) 传统 PSO 算法缺陷与改进策略

传统粒子群优化算法(Particle Swarm Optimization, PSO) [20]通过模拟鸟群觅食行为，将每个参数组合视为“粒子”，在解空间中通过迭代更新粒子的位置和速度搜索最优解。但在 MPI 参数调优场景中存在后期收敛速度慢、易陷入局部最优解的问题。针对该缺陷，重点对算法进行两方面改进：

① 惯性权重自适应调整策略：摒弃传统 PSO 的固定惯性权重，采用线性递减 + 非线性修正的自适应惯性权重，随着迭代次数增加，动态减小惯性权重，平衡算法的全局搜索能力(前期)与局部搜索能力(后期)。动态惯性权重计算公式为：

$$\omega(t)=(\omega_{\max}-\omega_{\min})\times\frac{T-t}{T}+\omega_{\min}+\alpha\times\mathrm{rand}(0,1) \tag{10}$$

其中，$\omega(t)$为第 t 次迭代的惯性权重，$\omega_{\max}=0.9$ (最大惯性权重)，$\omega_{\min}=0.4$ (最小惯性权重)，T 为总迭代次数，t 为当前迭代次数，$\alpha=0.1$(非线性修正系数)，rand(0,1)为 0~1 之间的随机数。

② 引入变异操作：在每次迭代后，对粒子的位置进行高斯变异[21]，对陷入局部最优的粒子进行位置扰动，避免算法陷入局部最优解，提升搜索的全面性，变异公式为：

$$x_i'(t)=x_i(t)+\sigma\times N(0,1) \tag{11}$$

其中，$x_i'(t)$为变异后的粒子位置，$x_i(t)$为变异前的粒子位置，$\sigma=0.05$ (变异标准差)，$N(0,1)$为标准正态分布随机数。变异操作仅对适应度值较差的 30%粒子执行，避免破坏已搜索到的较优解。

(2) 自动调优算法运行流程

IPSO 自动调优算法的运行流程结合 MPI 参数特性与映射模型的预测能力，加入参数组合剪枝策略与调优终止条件自适应调整机制，优化解空间与迭代次数，提升调优效率，具体流程见图 2 所示：

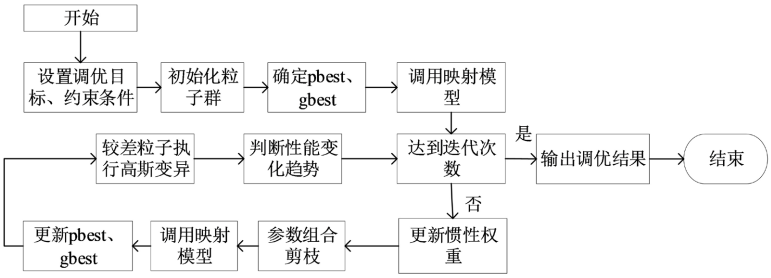

Figure 2. Operation flow of IPSO automatic tuning algorithm
图 2. IPSO 自动调优算法运行流程

剪枝策略基于 MPI 参数谱系中的参数关联规则，在粒子位置更新后立即剔除不合理的参数组合，如进程数量大于 CPU 核心数、通信缓冲区大小超出内存限制、低版本 MPI 选择高版本参数等，减少无效搜索；同时根据调优过程中参数组合的性能变化趋势动态调整迭代次数，若连续若干轮迭代的全局最优适应度值变化率均小于阈值，则提前终止迭代，在保证调优效果的前提下缩短调优时间。

2.3. 自动化回归测试与性能基线生成方法

MPI 并行应用在参数调整、版本迭代或运行环境变化后，可能出现性能退化、功能异常等问题，传统人工回归测试效率及覆盖率低，难以满足工程数值仿真应用高效研发需求。本章围绕自动化回归测试与性能基线生成展开研究，设计高效的测试流程及性能基线生成方法，保障 MPI 并行应用研发及测试的高效性、可对比性与可追溯性。

2.3.1. 自动化回归测试流程

将自动化回归测试划分为测试准备、测试用例生成、测试自动执行、测试结果分析四个阶段，各阶段自动进行，无需人工干预。完整流程及各阶段核心任务见图 3 所示：

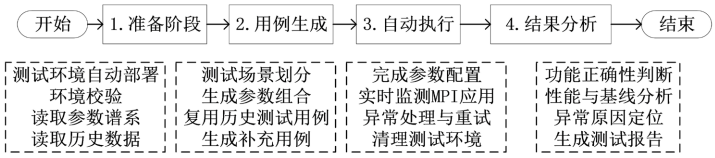

Figure 3. Automated regression testing process
图 3. 自动化回归测试流程

2.3.2. 性能基线生成

如图 4 所示，性能基线生成主要包含数据预处理、基线计算、基线验证三个步骤，每个步骤自动化执行，生成的性能基线为分类基线，即按参数配置、运行环境、测试场景分类生成基线，确保基线的针对性与准确性。

(1) 数据预处理

对原始性能数据实施规范化处理，旨在消除噪声、抑制抖动、规范样本划分，依次完成数据清洗、数据平滑、数据分配，最终输出高质量、适用于基线建模的标准性能数据集。

首先开展数据清洗，剔除无效、异常及重复数据，保障原始数据的有效性与一致性。无效数据包括字段缺失、采样失败、进程异常退出所产生的无效记录；基于“任务 ID + 节点 ID + 时间戳”联合键值去重，保留最新有效记录；针对运行耗时、CPU 利用率、通信延迟等核心指标，采用 3σ 准则剔除偏离均

值±3 倍标准差的极端异常值。

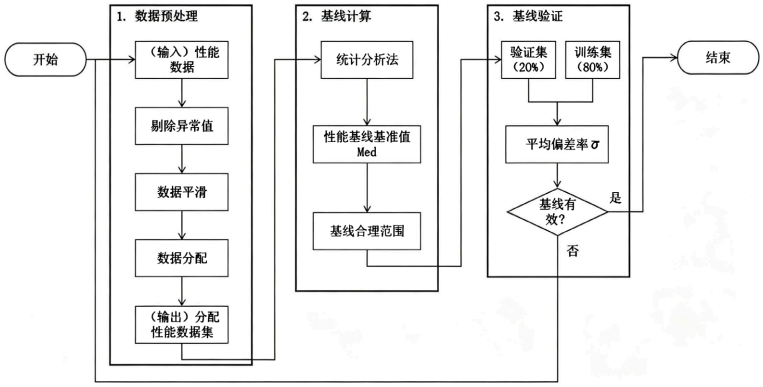

Figure 4. Performance baseline generation process
图 4. 性能基线生成流程

随后，采用滑动窗口均值滤波算法对清洗后的数据进行平滑处理，抑制由系统负载波动、网络扰动、MPI 进程调度抖动引入的高频随机噪声，保留真实性能变化趋势。

最后，采用分层随机抽样，参考机器学习与性能基线建模领域的常规工程实践，采用 8:2 比例划分训练集与验证集，为后续基线计算提供可靠基础数据。

(2) 基线计算

基于预处理后的训练集数据，采用中位数稳健统计分析法确定性能基线基准值与合理波动区间，为性能对比与退化识别提供量化依据。相较于均值，中位数(Med)对残留异常值、非对称分布数据具有更强鲁棒性，更适配 MPI 并行性能数据的随机波动特征。实验验证表明，中位数相较均值可使基线偏差率降低 2.9%，抗抖动能力更优。

基线合理波动范围设定为基准值 Med 上下浮动 5%，即区间[Med × 0.95, Med × 1.05]。通过对 3%、5%、7%三组边界进行敏感性验证，±5%区间可覆盖 95%以上正常性能抖动，兼顾异常识别灵敏度与正常波动容错性，为最优阈值区间。

(3) 基线验证

基线验证沿用预处理阶段相同的分层抽样策略，从分类性能数据集中随机抽取 20%样本作为验证集，剩余 80%作为训练集。通过以下公式计算验证集性能数据与基线基准值的平均偏差率[22]：

$$\bar{\sigma} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{x_i - \mathrm{Med}}{\mathrm{Med}}\right| \times 100\%$$

其中，x_i 为验证集单条性能数据，Med 为基线基准值，n 为验证集样本数量。

若平均偏差率 ≤5%，且验证集中 95%以上数据落在基线合理范围内，则判定基线有效；否则重新扩大训练集，重复数据预处理、基线计算步骤，直至基线验证通过。

2.3.3. 性能基线动态更新机制

为解决版本迭代、环境变化、参数优化导致的基线失效问题，设计定期更新 + 触发式更新的双重动态更新机制，确保性能基线始终与应用实际性能匹配，更新过程全程自动化。

定期更新基于工程数值仿真应用的常规迭代周期设置固定更新频率，默认每月自动更新 1 次。每次更新时，系统自动采集周期内自动化回归测试的所有有效性能数据，整合至对应场景的分类性能数据集，

重新计算基线基准值与合理范围，并与原基线对比，根据基准值变化率是否超过设定阈值决定是否将原基线替换为新基线。

此外，当软件功能、环境、参数出现重大变化时，系统自动触发基线紧急更新，触发条件及阈值基于 MPI 并行性能工程实践、敏感性分析与行业经验设定，兼顾变更显著性与基线稳定性，具体如下：

(1) MPI 参数配置变更：核心参数新增≥2 个、参数取值范围变化率 ≥30%、参数关联规则修改≥1 条；

(2) 运行环境变更：硬件(CPU 核心数、内存容量)升级比例 ≥ 20%、操作系统/MPI/编译器版本迭代(主版本号变更)；

(3) 软件版本迭代：核心算法重构、新增核心功能模块、整体架构调整；

(4) 性能数据异常波动：连续 3 次回归测试数据偏差率 ≥15%，或单指标偏差率 ≥25%。

满足任一条件，系统自动触发基线紧急更新，无需等待定期更新周期，更新流程与定期更新一致。

2.3.4. 性能基线可视化与异常预警机制

在平台的性能基线监测模块实现基线的多维度可视化展示，采用图、表结合的方式，支持研发人员按测试场景、运行环境、参数配置、时间维度筛选查看单指标基线趋势、多指标对比、实时性能与基线对比以及进行历史基线回溯等操作。

结合性能基线实现多级异常预警，当应用实时性能数据或回归测试性能数据超出基线合理范围时，系统自动触发预警，确保研发人员及时发现性能退化问题。

按性能偏差率，将预警划分为一级、二级、三级 3 个级别，一级预警表示轻微异常，通常无需人工紧急处理；二级预警为一般异常，研发人员通常应在规定时间内排查原因；三级预警为严重异常，研发人员应立即排查，并暂停相关版本迭代。

为确定最优预警阈值，设计三组分级方案开展敏感性分析，分别为低阈值(5%/15%/25%)、中阈值(10%/20%/30%)与高阈值(15%/25%/35%)，通过误报率与漏检率评估分级效果，测试结果如下：

(1) 低阈值(5%/15%/25%)：预警过于灵敏，误报率达 28.6%，易将正常性能抖动误判为异常，频繁干扰研发，预警可信度不足；

(2) 中阈值(10%/20%/30%)：误报率仅 6.3%、漏检率 4.1%，可有效区分正常波动与真实性能退化，在预警灵敏度与可靠性间实现最佳平衡，为最优分级方案；

(3) 高阈值(15%/25%/35%)：预警过于宽松，漏检率达 19.2%，难以及时识别潜在性能退化风险，预警有效性不足。

综上，最终选取 10%/20%/30%作为默认预警阈值划分方式；同时支持阈值自定义配置，以适配不同应用场景的容错需求。

3. 平台设计与实现

整合本文研究的 MPI 参数与性能映射模型、自动调优算法、自动化回归测试与性能基线生成方法，实现 MPI 并行应用从性能预测到自动调优、回归测试与性能基线的全流程闭环管理，设计实现 MPI 并行应用测试调优一体化平台。平台采用 B/S 架构，兼顾操作便捷性与跨平台支持，研发人员可通过浏览器访问，无需安装本地客户端。

3.1. 整体架构设计

SimPerf 平台采用 Vue + SpringBoot 开发框架与分层设计思想，整体分为 UI 界面层、业务逻辑层、数据层三个核心层，各层相互独立，通过标准化接口进行协同通信，同时引入中间件层提供通用服务支撑，确保平台的兼容性、可扩展性与稳定性；最底层为独立的基础设施层，用于向上层服务提供计算、

存储、网络等基础环境。平台整体架构见图 5 所示：

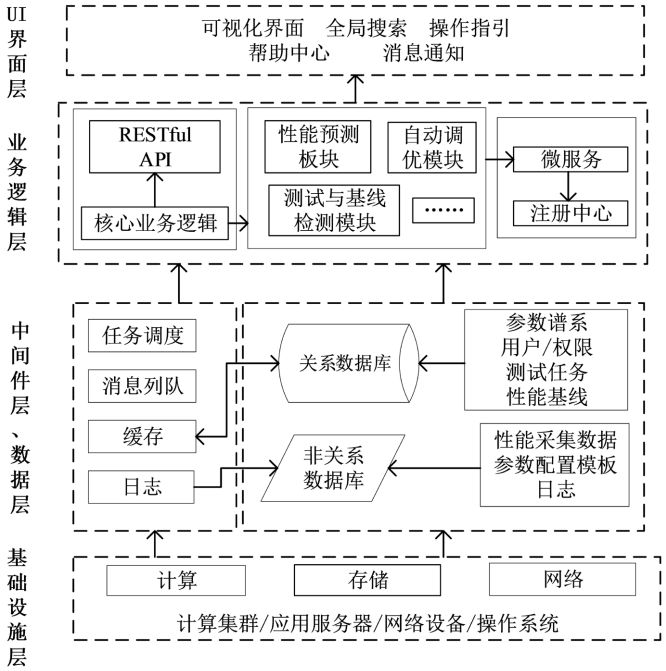

Figure 5. Overall architecture of SimPerf platform
图 5. SimPerf 平台整体架构

(1) UI 界面层

UI 界面层是研发人员与平台的交互入口，包含可视化操作界面，提供全局搜索、操作指引、消息通知、帮助中心等通用功能。

(2) 业务逻辑层

业务逻辑层是平台的大脑，采用微服务架构[23]，各服务通过注册中心实现服务发现与调用，核心职责包括处理 UI 界面层传递的操作指令、接收中间件层的服务支撑、实现平台所有核心业务逻辑、提供与 UI 界面层通信的标准化 RESTful API 接口等。

(3) 中间件层

中间件层为平台提供通用技术服务支撑，解决跨模块、跨层的通用问题，提升平台的运行效率、稳定性与可维护性，平台使用的中间件包括任务调度、消息队列、缓存、日志等。

(4) 数据层

数据层负责平台数据的持久化存储、管理与访问，采用关系型数据库 + 非关系型数据库混合存储架构，关系型数据库用来存储结构化、强关联的数据；非关系型数据库则用来存储非结构化、半结构化的数据。

(5) 基础设施层

基础设施层是平台运行的物理基础，为平台提供计算、存储、网络等硬件资源与运行环境，包括计算集群、应用服务器、网络设备、操作系统等。

3.2. 核心功能模块实现

基于平台整体架构设计，采用模块化开发思想设计四大核心功能模块，各模块相互解耦，通过接口

实现功能协同与数据互通，构成完整的 MPI 并行应用测试调优一体化解决方案。

(1) 性能预测与自动调优模块

整合多维度特征提取、XGBoost 性能映射模型、IPSO 自动调优算法实现，核心功能包括性能预测、参数自动调优、调优结果分析等。

(2) 自动化回归测试模块

基于前文设计的自动化回归测试流程与机制实现，核心功能包括测试用例自动生成、测试任务创建与调度、测试过程实时监控、测试结果自动分析、测试报告生成等。

(3) 性能基线监测模块

实现性能基线生成、动态更新、可视化展示、性能异常预警等核心功能，是平台实现性能闭环管理关键模块。

(4) 平台管理模块

实现平台的日常管理与维护功能，保障平台的安全性、稳定性与可维护性，采用基于角色的访问控制(RBAC)模型[24]实现权限管理，核心功能包括用户管理、角色权限管理、数据管理、日志管理、系统设置等。

3.3. HPC 作业调度系统集成

平台与高性能计算(HPC)集群深度集成，实现 MPI 并行作业调度与跨节点性能数据的同步采集、汇聚与管理。

作业调度方面，采用插件化适配与统一调度接口的集成方案。针对 Slurm、PBS Pro、LSF 等主流 HPC 调度系统[25]，开发适配插件，封装作业提交、状态查询、作业终止等功能，屏蔽底层调度差异；同时提供标准化 RESTful 调度接口，统一实现作业提交、全生命周期监控、优先级设置、资源分配与超时控制，保证调度逻辑的一致性与可扩展性。

跨节点性能数据采集方面，采用分布式采集代理 与中心化数据汇总架构，解决 MPI 并行程序多节点运行时的分布式数据采集、实时同步与汇总聚合。在各计算节点部署轻量级采集代理 PDC Agent，基于 MPI 内置性能接口与系统监控工具，采集进程耗时、通信延迟、CPU/内存利用率等关键指标；采集数据经消息队列实时推送至平台后端，按测试任务、计算节点、MPI 进程维度进行聚合、去重与缺失值补全。PDC Agent 内置本地缓存与断点续传容错机制，网络中断时缓存数据，链路恢复后自动补传，有效降低数据丢失风险，保障采集过程的稳定性与可靠性。

3.4. 平台适配性与可扩展性

为确保平台能够适配不同的工程数值仿真场景、运行环境、硬件架构，同时支持后续的功能扩展与技术迭代，提升平台的实用性与生命周期，重点从环配功能扩展两方面开展平台适配性与可扩展性设计。环境适配方面，平台采用跨平台、松耦合的技术架构，支持多操作系统、多 MPI 版本、多编译器、多硬件架构的适配，确保能够与工程数值仿真应用的开发、运行环境无缝衔接；功能扩展方面，采用模块化、插件化的设计思路，将平台的核心功能拆分为独立的模块，同时预留标准化功能扩展接口，支持新增功能模块的快速集成，无需修改平台核心代码，降低功能扩展的难度与成本。

4. 实验与结果分析

为验证本研究中的 MPI 参数与性能映射模型、自动调优算法、自动化回归测试与性能基线生成方法及一体化平台的科学性、有效性与实用性，本章设计系统性的实验方案，选取工程数值仿真实际场景中

的 MPI 并行应用作为测试对象，开展实验验证，对实验结果进行详细分析，验证研究成果的应用价值。

4.1. 实验环境

选取两台高性能服务器作为实验节点，每台服务器配置 64 核 Intel 8358P CPU (支持超线程)、256 GB DDR4 内存，总核数 128 核，总内存 512 GB，两台服务器使用 100Gbps HDR 网络互联，构成 MPI 并行计算集群，模拟工程数值仿真并行运行环境。

服务器安装工程数值仿真领域主流 64 位 CentOS 8.5 操作系统、OpenMPI 4.1.5 和 MPICH 4.3.1、及 GCC 9.4.0 和 Intel C++ Compiler 2021 编译器，安装 mpstat (CPU 监测)、free/ps (内存监测)、nmon (综合硬件监测)、perf (算力吞吐量监测)等性能监测工具。

选择 Java™ 11 LTS 为平台后端服务基础运行环境，Python 3.6.8 为机器学习脚本语言，选择 MySQL 8.0.31 (关系型)和 MongoDB 5.0.15 (非关系型)两款数据库管理系统，同时安装配置 Nacos 2.1.0 (服务注册)、RabbitMQ 3.10.0 (消息队列)、Redis 6.2.7 (缓存)、Celery 5.2.7 (任务调度)等中间件软件，搭建本平台的运行环境。

4.2. 测试场景设计

基于实验硬件环境，选取支持 MPI 并行计算的国产电磁仿真分析软件为待测仿真应用，分别使用网格规模为 2 千万(小)、4 千万(中)、2 亿(大)、4 亿(超大)的 ZTZ-99 坦克模型算例，计算其在 1GHz 下的双站 RCS。为每种规模算例设置多组测试用例，覆盖本研究的性能预测与调优、自动化回归测试与性能基线核心内容，分别验证 XGBoost 性能映射模型的预测精度与 IPSO 自动调优算法的优化效果、自动化回归测试方法的效率与覆盖率、以及性能基线生成与动态更新方法的有效性。

4.3. 实验内容与结果分析

(1) XGBoost 性能映射模型预测精度实验

分别为 4 种仿真规模随机生成 50 组不同的 MPI 参数组合，在实验环境中实际运行这 200 组参数组合，采集运行耗时、CPU 平均利用率、通信延迟三大核心性能指标的实际值；同时将 200 组参数组合与环境信息输入平台的 XGBoost 映射模型，获取性能预测值；计算预测值与实际值的平均绝对误差(MAE)、均方误差(MSE)、预测准确率，验证模型的预测精度。

如图 6 所示，实验结果显示，4 种仿真规模下模型的 MAE 均未超过 0.036，MSE 均未超过 0.0026，预测准确率均高于 95%，表明构建的 XGBoost 性能映射模型具有较高的预测精度，能够准确反映 MPI 参数组合与应用性能之间的非线性关联关系。

随着仿真规模的增大，模型的预测误差略有上升，预测准确率略有下降，主要原因是仿真规模越大，应用的运行过程越复杂，受硬件资源波动、进程间通信冲突等偶然因素的影响更大，导致实际性能数据的波动略大，但误差仍在可接受范围内。

实验结果还显示，模型对 CPU 利用率的预测精度最高，对通信延迟的预测精度略低，主要原因是通信延迟受网络环境的微小波动影响更大，而模型的环境特征中未完全涵盖所有网络细节参数，但整体预测精度仍满足工程实际需求。

(2) 自动调优效果实验

每种仿真规模采用默认 MPI 参数配置作为调优前的基准配置，将基准配置输入平台 IPSO 自动调优算法，设置调优目标为多目标加权优化，其中运行耗时权重 0.5、CPU 利用率权重 0.3、通信延迟权重 0.2，执行自动调优；调优完成后，在实验环境中实际运行最优参数组合，采集调优后的性能指标，与调优前的基准配置对比，计算性能指标提升幅度。

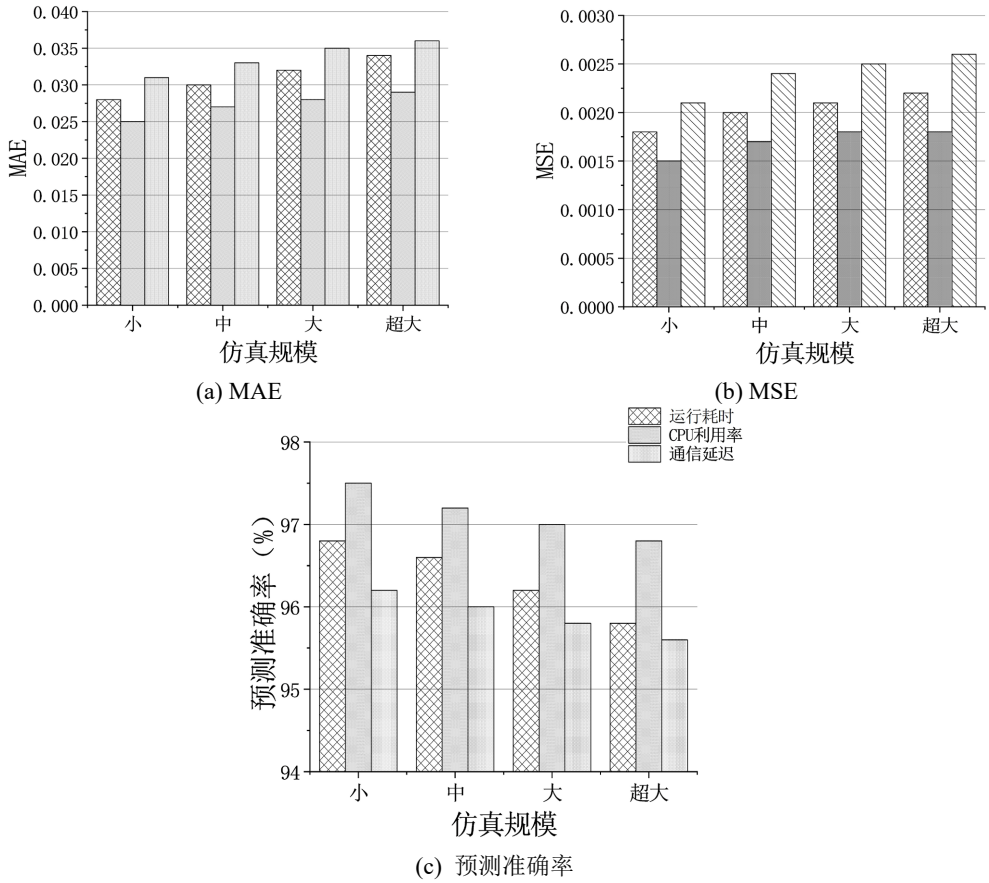

Figure 6. Model prediction error and accuracy under different scales
图 6. 不同规模下模型预测的误差与准确率

如图 7 所示，4 种仿真规模下应用的运行耗时、CP 利用率、通信延迟均得到明显优化，其中超大规模仿真场景下优化效果最佳，运行耗时缩短 23.2%，平均 CPU 利用率提升 14.1%，通信延迟降低 24.5%，表明 IPSO 自动调优算法能够有效搜索到适配不同仿真规模的最优参数组合。

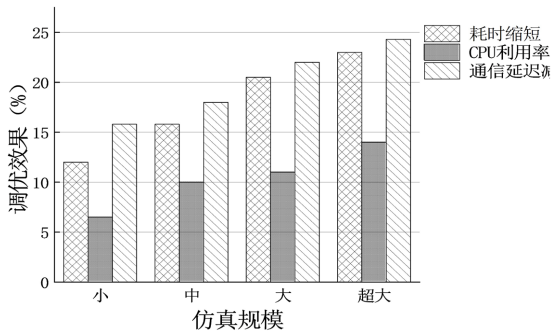

Figure 7. Performance improvement after automatic tuning under different scales
图 7. 不同规模下采用自动调优后性能提升

(3) 自动化回归测试实验

选取测试用的国产电磁仿真分析软件 CEES 的 3 个迭代版本：V1.0、V2.0、V3.0，其中 V3.0 为最新

版本，包含核心算法优化与功能模块新增。3 个版本分别采用自动化回归测试方法与传统人工测试方法进行回归测试，测试覆盖软件的所有核心功能与 MPI 参数配置场景；记录两种方法的总耗时与测试覆盖率。测试覆盖率判定标准为：基于需求用例与参数组合的覆盖情况，计算功能覆盖率与参数覆盖率的加权平均值，权重各为 0.5。

如图 8 所示，使用平台自动化测试的平均单版本测试耗时只有 11.2 小时，较人工测试 39.5 小时的平均时间缩短 71.6%。这是因为自动化测试实现了测试用例生成、测试环境部署、测试过程执行、测试结果分析全程自动化，无需人工干预，避免人工测试中的重复操作与等待时间；同时支持多任务并行测试，大幅提升测试效率。

同时，自动化测试的覆盖率达到 96.7%，较人工测试 83.3%覆盖率提升 16.1%。主要原因是自动化测试采用正交试验设计法生成测试用例，能够覆盖所有核心参数组合与功能模块，避免了人工测试中的遗漏问题。

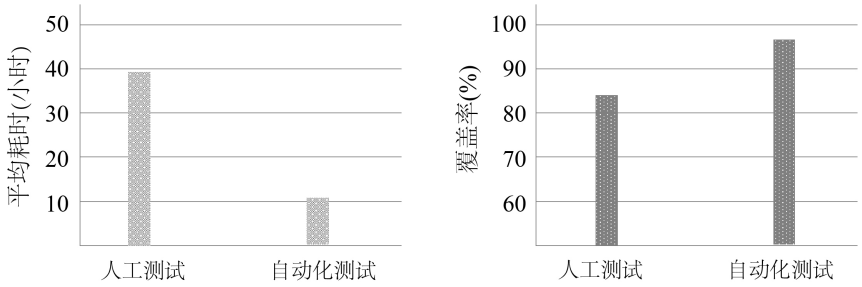

Figure 8. Comparison between automated and manual regression testing
图 8. 自动化回归测试与人工测试对比

(4) 性能基线有效性实验

基于软件 V2.0 版本的自动化回归测试性能数据，在平台中生成性能基线；随后分别将 V2.0、V3.0 版本的性能数据与基线对比，验证基线的基准性能表征能力；同时在 V3.0 版本中人为设置 MPI 通信参数配置不合理的故障以模拟性能退化，验证平台的异常预警准确性与问题定位能力。性能偏差率判定标准为：若性能数据偏差率 ≤10%，判定为性能正常；若偏差率 >10%，判定为性能退化，触发异常预警。

Table 1. Experimental results of performance baseline effectiveness
表 1. 性能基线有效性实验结果

测试版本	实验过程与结果
V2.0	性能数据的平均偏差率为 3.2%，无预警触发
V3.0 (正常版本)	平均偏差率为−8.5%(负号表示性能提升)
V3.0 (故障版本 1)	第一次测试，平均偏差率为 8.7%，触发一级预警(轻微异常)
V3.0 (故障版本 2)	调整故障参数使偏差率升至 15.3%，触发二级预警(一般异常)，并给出初步排查建议“通信延迟显著升高，建议检查 MPI 通信模式、缓冲区大小等通信类参数配置”

见表 1 所示，实验过程与结果表明，性能基线能够准确反映软件的基准性能水平，对软件正常迭代带来的性能变化具有良好的包容性，同时能够快速识别性能退化问题。平台的异常预警机制准确、有效，能够根据性能偏差率自动划分预警等级，并给出针对性的排查建议，帮助研发人员快速定位问题，提升问题解决效率。

5. 结束语

本研究针对工程数值仿真研发人员面临的 MPI 并行应用测试调优低效、性能管理缺失等核心问题，开展关键技术与一体化平台研发。系统梳理并构建了标准化 MPI 核心参数谱系，基于 XGBoost 算法建立 MPI 参数性能与映射模型，结合改进 IPSO 算法实现并行参数自动调优；同时研发自动化回归测试技术，提出融合统计分析与异常剔除相结合的性能基线生成方法，最终设计实现面向 MPI 并行仿真应用的参数调优与自动化性能测试平台。实验结果表明，该平台有效降低了研发人员的并行开发门槛，大幅提升研发与测试效率，性能测试调优效率提升 70%以上，测试覆盖率提升 16%以上。研究成果可为工程数值仿真应用提供一站式、全流程的测试调优解决方案，具备显著的工程应用价值与推广前景。

当前研究仍存在一定局限性：XGBoost 模型依赖大量历史数据训练，大规模场景下采集成本较高，小样本条件下模型泛化能力受限；同时，平台对集群负载、网络扰动等非确定性性能抖动仅采用简单异常剔除，易出现误判。在未来的工作中，可考虑引入迁移学习[26]与联邦学习[27]，通过跨场景预训练降低数据依赖、提升小样本精度；融合 LSTM 时序模型[28]学习正常抖动规律，区分随机波动与真实退化，降低预警误报率；进一步拓展平台对混合云、容器化 HPC 集群的适配能力，支撑跨云环境下 MPI 并行性能的一体化测试与调优。

参考文献

[1] 叶宁, 付康, 胡少文, 等. 基于 MPI 的异构算力资源融合调度平台[J]. 计算机与现代化, 2025(12): 38-45.

[2] 郑文旭. 并行计算机系统中 MPI 运行时参数调优方法与关键技术研究[D]: [硕士学位论文]. 长沙: 国防科技大学, 2019.

[3] 刘轶, 高玉林, 张国振. 并行程序运行故障原因识别[J]. 国防科技大学学报, 2022, 44(5): 45-52.

[4] 严畅, 朱杰. 基于 Vtune 的网络协议性能测试技术[J]. 信息技术, 2010, 34(6): 72-74.

[5] Wylie, B.J.N., Giménez, J., Feld, C., Geimer, M., Llort, G., Mendez, S., *et al*. (2025) 15+ Years of Joint Parallel Application Performance Analysis/Tools Training with Scalasca/Score-P and Paraver/Extrae Toolsets. *Future Generation Computer Systems*, **162**, Article 107472. https://doi.org/10.1016/j.future.2024.07.050

[6] Bader, M., Bode, A., Bungartz, H.J., *et al*. (2014) A Case Study: Holistic Performance Analysis on Heterogeneous Architectures Using the Vampir Toolchain. *Advances in Parallel Computing*, **25**, 793-802.

[7] Alghamdi, A.S.A., Alghamdi, A.M., Eassa, F.E. and Khemakhem, M.A. (2020) ACC_TEST: Hybrid Testing Techniques for MPI-Based Programs. *IEEE Access*, **8**, 91488-91500. https://doi.org/10.1109/access.2020.2994172

[8] Choi, J., Fink, Z., White, S., Bhat, N., Richards, D.F. and Kale, L.V. (2022) Accelerating Communication for Parallel Programming Models on GPU Systems. *Parallel Computing*, **113**, Article 102969. https://doi.org/10.1016/j.parco.2022.102969

[9] Hunold, S., Carpen-Amarie, A., Lübbe, F.D. and Träff, J.L. (2016) PGMPI: Automatic Verification of Self-Consistent MPI Performance Guidelines. In: *Lecture Notes in Computer Science*, Springer, 433-446. https://doi.org/10.1007/978-3-319-43659-3_32

[10] 曹亚浩, 吕云飞, 陈源宝, 等. 基于改进粒子群算法的 UUV 集群回收任务规划[J]. 舰船科学技术, 2025, 47(23): 92-97.

[11] 陈雪娟, 许欢欢, 杨泽. 机器学习驱动的智能视觉分析在目标检测与结构动力特性识别中的应用[J]. 无线互联科技, 2026, 23(4): 16-20.

[12] 舒予, 金昊, 江昊. 多维度特征增强的用户转发行为预测方法[J]. 中国电子科学研究院学报, 2025, 20(5): 529-538.

[13] 宋芮芮, 王雷春, 何运平, 等. 基于混合自注意力和差异归一化的长时间序列预测[J]. 计算机应用, 2026, 46(5): 1499-1506.

[14] 周子程, 张仰森, 王璞. 资源受限场景下大模型算术优化与分析方法[J/OL]. 小型微型计算机系统, 2025: 1-12. https://doi.org/10.20009/j.cnki.21-1106/TP.2025-0307, 2025-11-14.

[15] 马超, 王建明, 高华, 等. 一种深度神经网络 SAR 图像目标识别可视化方法[J]. 空天预警研究学报, 2023, 37(4):

295-300.

[16] 章祉瑶, 聂斌, 郑水飞. 交互作用特征选择研究综述[J/OL]. 计算机工程与应用, 2026: 1-30. https://link.cnki.net/urlid/11.2127.tp.20260314.1256.010, 2026-03-16.

[17] 徐洋, 罗润志, 许豪, 等. 基于数据驱动的自动泊车系统量化评价方法[J/OL]. 计算机工程与应用, 2026: 1-17. https://link.cnki.net/urlid/11.2127.TP.20260320.1131.008, 2026-03-20.

[18] 郑嘉伟, 王粉花, 赵波, 等. 基于多层次特征交互的点击率预测模型[J]. 实验室研究与探索, 2022, 41(5): 21-25+49.

[19] 张挺, 王宗锴, 林震寰, 等. 基于自动终止准则改进的 kd-tree 粒子近邻搜索研究[J]. 工程科学与技术, 2024, 56(6): 217-229.

[20] 丁承君, 耿宇坤, 胡健鑫, 等. 基于自适应时域 MPC 的无人车轨迹跟踪控制[J]. 科学技术与工程, 2025, 25(23): 9883-9891.

[21] 杨正, 周睿, 李鹏. 改进的 RSA 算法及在疫情传播 SVM 模型中的应用[J]. 计算机工程与设计, 2025, 46(10): 3016-3023.

[22] 付鹏斌, 陈帅帅, 杨惠荣, 等. 结合依存关系与同义词词林的相似度计算[J]. 计算机技术与发展, 2020, 30(1): 13-18.

[23] 韩世平, 李林, 陈杰, 等. 基于微服务架构的多源异构数据实时处理平台设计[J]. 国外电子测量技术, 2025, 44(11): 251-256.

[24] 陈阵, 蒋建民, 郭继文. 基于角色的访问控制研究综述[J/OL]. 信息安全学报, 2025: 1-22. https://link.cnki.net/urlid/10.1380.TN.20251211.1125.002, 2025-12-12.

[25] 沈瑜, 孙婧, 李娟. 基于 Slurm 的气象高性能计算资源调度管理及应用[J]. 计算机技术与发展. 2025, 35(11): 180-187.

[26] 韩培丽, 黄雯. 迁移学习在电力物联网数据异常检测中的应用[J]. 智能物联技术, 2025, 57(6): 120-124.

[27] 王宇, 唐小川. 基于贡献度的联邦学习模型聚合算法[J]. 信息技术. 2026(3): 1-6.

[28] 李思琪, 俞琨, 陈宇皓. 基于 ARIMA 和 LSTM 的高性能计算平台资源使用的预测研究[J]. 计算机科学, 2025, 52(9): 178-185.