Survey paper

What artificial intelligence can do for high-performance computing systems?

Pierrick Pochelu [a] [iD],*, Hyacinthe Cartiaux [b] [iD], Julien Schleich [b] [iD]

[a] *LuxProvide S.A., Bertrange, Luxembourg*
[b] *University of Luxembourg, Esch-sur-Alzette, Luxembourg*

ARTICLE INFO

ABSTRACT

High-performance computing (HPC) centers consume substantial power, incurring environmental and operational costs. This review assesses how artificial intelligence (AI), including machine learning (ML) and optimization, improves the efficiency of operational HPC systems. Approximately 1,800 publications from 2019 to 2025 were manually screened using predefined inclusion/exclusion criteria; 74 "AI for HPC" papers were retained and grouped into six application areas: performance estimation, performance optimization, scheduling, surrogate modeling, fault detection, and language-model-based automation.

Scheduling is the most active area, spanning research-oriented reinforcement-learning schedulers to production-friendly hybrids that combine ML with heuristics. Supervised performance estimation is foundational for both scheduling and optimization. Graph neural networks and time-series models strengthen anomaly detection by capturing spatio-temporal dependencies in production telemetry. Domain-specialized language models for HPC can outperform general-purpose LLMs on targeted coding and automation tasks. Together, these findings highlight integration opportunities such as LLM-based operating-system concepts and underscore the need for advances in MLOps, standardization of AI components, and benchmarking methodology.

1. Introduction

High-performance computing (HPC) has become an essential tool for advancing scientific research and industrial applications, providing the computational power needed to solve complex problems in fields such as simulation, machine learning (ML), and big data processing. However, the increasing complexity of these systems due to growing heterogeneity and scale to meet rising demand presents significant challenges in optimizing job scheduling (Li et al., 2022; Fan et al., 2022b; Zhang et al., 2020). Traditional algorithms like First-Come-First-Serve (Fan et al., 2022a; SchedMD, 2025), Shortest Job Next, and Round-Robin often fail to efficiently allocate jobs to resources, leading to sub-optimal performance and wasting power consumed (Malms et al., 2022). As HPC environments scale up, they also become more susceptible to hardware or software failures, disrupting ongoing tasks and necessitating re-scheduling or job restarts, further straining resources.

This complexity is further compounded by the threat of cyber-attacks (Lee et al., 2024), particularly in HPC systems exposed to public networks. Recent advances in Language Model (LM)-based tools show potential in supporting HPC cybersecurity engineers by automatically generating scripts and configuring firewall rules (Nowak et al., 2024). However, cybersecurity within HPC remains a niche research domain,

with limited empirical evidence on the deployment and effectiveness of such approaches in production environments.

Additionally, the rapid pace of innovation in areas like accelerator architectures and large-scale AI (Artificial Intelligence) applications, such as Large Language Model (LLM) training (Rasley et al., 2020), inundates HPC teams with vast amounts of information (Heldens et al., 2020), making automation increasingly necessary to support scientists and optimize system management.

Artificial Intelligence (AI) in this paper refers to the broad class of computational methods and algorithms that enable machines to perform tasks traditionally associated with human intelligence, such as learning, reasoning, decision-making, and language understanding. This definition encompasses machine learning (ML)—including supervised, unsupervised, reinforcement, and deep learning approaches. The definition is stretched to non-ML including optimization techniques such as evolutionary algorithms, Bayesian optimization, integer linear programming, and heuristics.

While machine learning currently dominates AI-for-HPC research, the definition of AI remains broader, encompassing both ML and non-ML approaches—from research prototypes to production systems. In practice, heuristic and rule-based algorithms often remain easier to deploy and maintain, as they avoid the complexities of building data

---

pipelines, handling dataset shift, or tuning hyperparameters. In contrast, ML methods can deliver strong performance but require significant overhead for data curation, model tuning, and robustness monitoring.

Two sub-fields illustrate this contrast. Tensor parallelism optimization frameworks like Alpa (Zheng et al., 2022) and Colossal-AI (Li et al., 2023) automate decisions using Integer Linear Programming and empirical profiling, without relying on internal ML models. Conversely, Colossal-Auto (Liu et al., 2023) employs a machine learning estimator to approximate performance and reduce search costs. In scheduling, many research systems adopt hybrid approaches. For example, GARL (Li et al., 2022) combines reinforcement learning with expert-driven heuristics, while other schedulers blend data-driven estimators with constraint-based optimization (Hu et al., 2021).

This review is primarily aimed at entry-level scientists and engineers who are new to the challenges of current HPC systems. In this review, the contributions of ML and heuristic optimization are emphasized in addressing problems that are difficult to solve with conventional methods, such as scheduling methods adapting to the dynamic nature of user activity with reinforcement learning (Li et al., 2022; Fan et al., 2022b; Zhang et al., 2020), automating decision-making in environments with massive, unstructured datasets (code operations (Munley et al., 2024; Godoy et al., 2023; Guo et al., 2024), graph dataset (Molan et al., 2024, 2023a), . . .). The application of ML in these areas serves as a complementary evolution of existing techniques, rather than a replacement.

As AI continues to evolve and mature, including ML algorithms and optimization methods, it presents promising opportunities to address the operational challenges inherent in HPC. The central research question guiding this review is: "What can AI do for HPC systems?" This question explores how AI techniques can be harnessed to support the HPC environment. The literature review aims to identify various areas for improvement.

"What can AI do for HPC systems?" This research question explores how AI methods can enhance the speed of applications, improve power efficiency, and reduce the operational costs of underlying HPC infrastructure. ML can distribute workloads more efficiently (Koslovski et al., 2024; Lubrano et al., 2024; Gu et al., 2023; Thanapol et al., 2023; de Freitas Cunha and Chaimowicz, 2023) than traditional schedulers that rely on First-Come-First-Serve heuristics. Additionally, AI can substitute long computations by learning from input parameters and their corresponding outputs (Miao et al., 2024; Radaideh and Kozlowski, 2020; Kadupitiya et al., 2019). Furthermore, it can monitor and automate infrastructure operations using statistical methods for anomaly detection (Wu and Guan, 2024; Molan et al., 2024, 2023a,b; Borghesi et al., 2022), as well as streamline scripting and coding processes.

Given the proliferation of research in AI and HPC, it is crucial to map out and synthesize the vast body of knowledge to understand current trends and identify future directions. This review aims to provide a comprehensive overview of the latest advancements in AI techniques applied to HPC optimization and operations. Specifically, this article aims to achieve three main objectives:

- Comprehensive Review: Provides a detailed analysis of research papers across various AI applications in HPC, summarizing goals and key methods.
- Evaluation of AI Techniques: Highlights the effectiveness of different AI methods, such as RL for scheduling, ML surrogates for simulation, and Language Model for coding, in improving HPC performance and efficiency.
- Identification of Trends: Identifies emerging trends in hardware and AI methods and current challenges in AI for HPC.

Section 2 reviews recent trends in HPC. Section 3 positions this work within prior surveys. Section 4 provides details of the methodology and the classification of collected papers across sub-fields. Section 5 summarizes key findings. Section 6 outlines a vision for potential AI integration in HPC. Finally, Section 7 concludes by synthesizing the current research directions and the integration challenges.

2. Background

2.1. Definition of key terms

AI refers to software capable of performing tasks requiring human intelligence, with methods including data-driven approaches, constraint satisfaction, and optimization algorithms (Turing, 1950). Recent advancements, particularly in deep learning, have broadened AI's scope across data applications supported by accelerators like GPUs and efficient tensor operations (Krizhevsky et al., 2012).

2.2. HPC/AI convergence

The convergence of HPC and AI can be framed through two main questions: "What can HPC do for AI?" and "What can AI do for HPC?" The first question focuses on how HPC can enhance AI by providing the computational power necessary to improve the accuracy of AI models, manage the memory requirements of large models with billions of parameters, and accelerate various stages of the AI lifecycle, including hyperparameter tuning, training, and inference. However, for the scope of this review, this question will be set aside to concentrate on the second question: "What can AI do for HPC?" This question explores how AI can be applied to optimize HPC infrastructure, including automatic parallelization and distribution of code, and how it can improve overall HPC system performance, efficiency, and sustainability (Malms et al., 2022).

The recent popularity of the fields of HPC and AI is reflected in the approximately 20,000 scientific Google Scholar entries responding to the queries with joint "HPC" and "AI" terms. Google Trends data in Fig. 1 shows the dynamics of the interest in such requests.

The first measurable number of requests appeared in July 2017, and the frequency increased significantly at the end of 2022, coinciding with the release of ChatGPT LLM. The interest in the intersection of HPC and AI is not only increasing, but the rate of increase itself is accelerating, driven by the rise of large AI applications. This growing interest suggests that the trend will continue to attract significant attention shortly. As more AI applications emerge that require high computational power, and as HPC systems are increasingly utilized, it becomes strategic to find smart ways to understand and manage these workloads. However, it is important to note that the general public may sometimes conflate single-node and multi-node HPC systems when querying about HPC/AI, which could lead to misunderstandings regarding the actual demand and interest in these infrastructures.

2.3. Environmental concerns and opportunities

Growing concerns around sustainability have led to increased emphasis on energy efficiency in HPC, as reflected in the Green-TOP500 list (TOP500 Project, 2025). While the adoption of accelerators has improved performance per watt (2, 3), overall energy consumption continues to rise due to increasing computational demands (4). To address this, AI-driven approaches such as intelligent scheduling, surrogate modeling, and real-time anomaly detection are being explored to enhance energy-aware resource management and system sustainability.

Despite the growing energy demands of HPC systems, their role in addressing environmental challenges is increasingly vital. For example, the Destination Earth initiative harnesses HPC to build a digital twin of the planet, supporting climate change adaptation and disaster risk management through advanced Earth system modeling (Hoffmann et al., 2023). In parallel, the U.S. Department of Energy's HPC4EnergyInnovation program (High performance, 2025) connects HPC resources with industry to accelerate the development of clean energy technologies and materials.
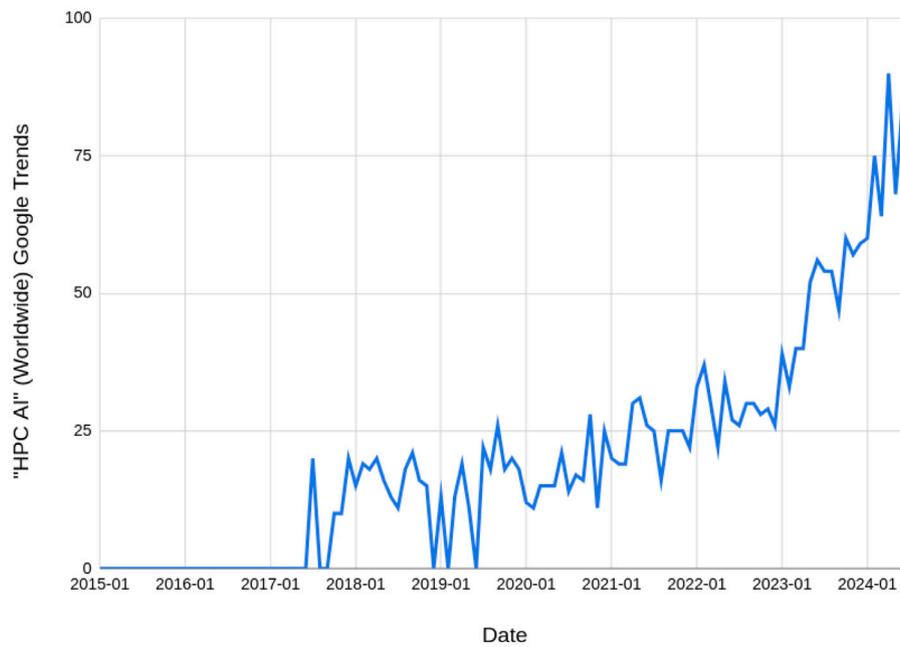
Fig. 1. World-wide Google queries with joint "HPC" and "AI" terms according to Google Trends. The measurement is sampled every month. The vertical axis represents relative search interest, normalized on a scale from 0 to 100.
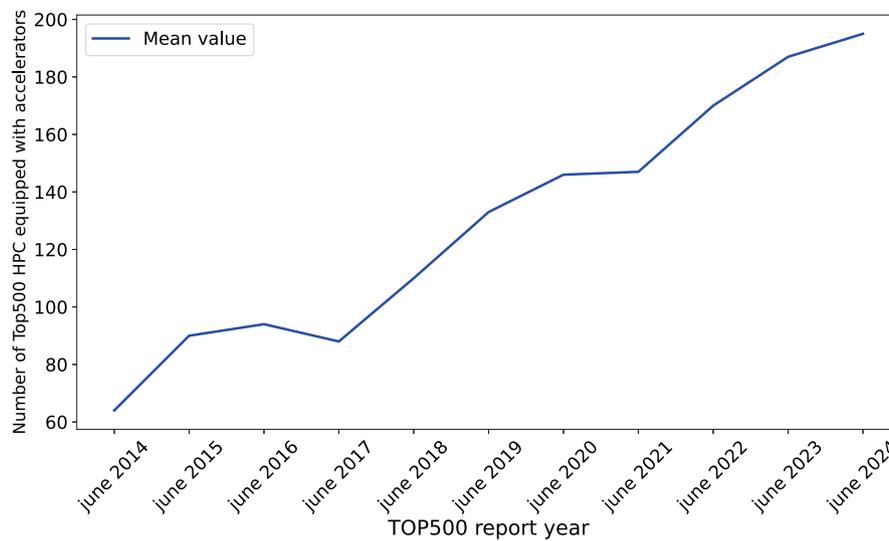

Fig. 2. Number of top 500 HPC systems (TOP500 Project, 2025) equipped with accelerators from June 2014 to June 2024. The number has increased from 64/500 in June 2014 to 195/500 in June 2024.

3. What can AI do for HPC? A review of existing surveys

Recent surveys have explored the potential of integrating AI into HPC. Notably, some works (Badia et al., 2024; Brewer et al., 2025) propose a conceptual taxonomy that distinguishes three modes of AI-HPC interaction:

- **AI-in-HPC**, where an AI model is used to replace a computationally intensive component, or the whole HPC simulation itself.
- **AI-out-HPC**, where AI is used to steer the HPC components or generate new data or parameterization.
- **AI-about-HPC**, where AI models are concurrent and coupled to the main HPC tasks and run synergistically with simulation.

This taxonomy emphasizes the physical or logical placement of AI within the HPC workflow rather than the specific use cases of AI models. While a general mapping between location and function is possible, ambiguity arises in scenarios in AI models addressing both application-level and system-level data sources and knobs (Menon et al., 2020; Borghesi et al., 2022).

Another relevant perspective is provided by the "Learning Everywhere" paper (Fox et al., 2019), which introduces another taxonomy:

- **MLautotuning**, which involves automated tuning of runtime systems.
- **MLcontrol**, where ML guides simulation execution to optimize objective and data collection.
- **MLafterHPC**, referring to post-simulation analysis using machine learning.
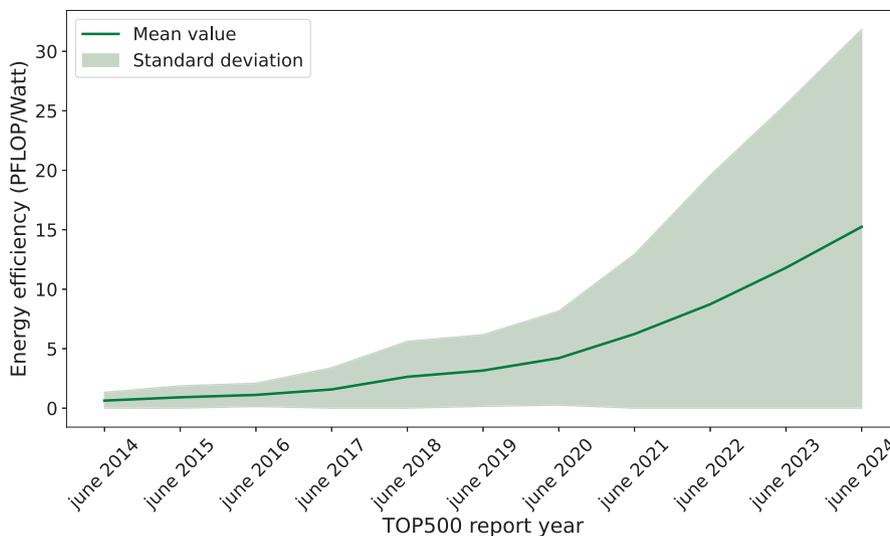
Fig. 3. Energy efficiency (PFLOPS/Watt) of top 500 HPC systems (TOP500 Project, 2025) equipped with accelerators from June 2015 to June 2024. A higher average is better. The average has moved from 0.6 in June 2014 to 15.2 in June 2024.
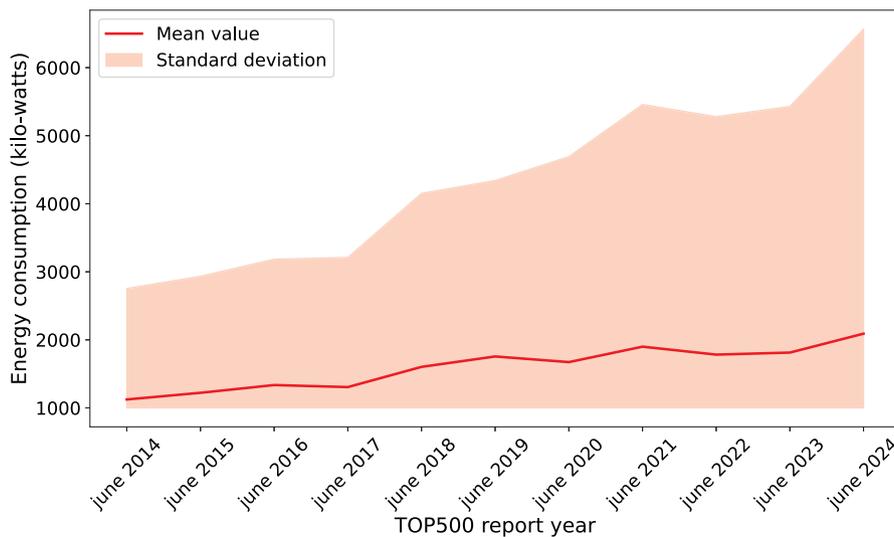

Fig. 4. Energy consumption (Kilowatts) of top 500 HPC systems (TOP500 Project, 2025) from June 2015 to June 2024. The average has increased from 1.1 MW in June 2014 to 2.1 MW in June 2024.

- **MLaroundHPC**, which includes surrogate modeling.

This categorization is primarily focused on improving simulation workflows rather than the HPC system infrastructure. Additionally, it centers exclusively on machine learning, omitting other AI paradigms such as heuristics and Integer Linear Programming, which are still widely used in HPC research. Finally, the split between runtime system tuning (MLautotuning) and application-level simulation optimization (MLcontrol) complicates the classification of papers that simultaneously optimize across both layers, such as HiPerBOt (Menon et al., 2020).

Other surveys concentrate on more specific aspects of AI in HPC. One survey (Kocot et al., 2023) explores energy-aware scheduling methods and emphasizes the increasing adoption of machine learning for multi-objective optimization. Another (Teranishi et al., 2025) examines the influence of AI, particularly large language models, on HPC software development, highlighting both productivity improvements and concerns around trustworthiness. A separate study (Merzky

et al., 2025) presents scalable architectures for hybrid ML-HPC workflows, with a particular focus on runtime resource management and orchestration.

While these surveys identify promising directions, they also reveal a lack of integration across AI-HPC use cases. Brewer et al. (2025) explicitly call for benchmarks to evaluate AI-enhanced workflows, and Badia et al. (2024) underscore the absence of shared APIs and data formats that would allow workflow components to interoperate. This fragmentation makes it difficult to compose, reuse, or benchmark AI methods across different HPC subsystems. As noted by Skvortsov and Stupnikov (2022), adopting FAIR (Findable, Accessible, Interoperable, Reusable) principles in HPC workflows could address many of these issues, but practical implementation remains limited.

In contrast to prior work, our survey provides a more comprehensive overview of both software and operating HPC in daily activities. It includes HPC components for scheduling, performance estimation, fault detection, performance optimization, and scripting/code automation.

By mapping interdependencies between AI applications and analyzing integration challenges, this survey aims to support a unified and intelligent HPC ecosystem.

4. AI for HPC methods

The research question is given by **"What artificial intelligence can do for high-performance computing systems?"**, the following section discusses the methodology employed to research this question, and then, the capabilities of AI are discussed.

4.1. Research methodology

The research scope of this review is narrowed towards AI for HPC. To cover relevant literature, searches were conducted in engines such as IEEE Xplore, Scopus, Google Scholar, and Google.

A comprehensive search strategy was employed using a combination of keywords with AND and OR logic to ensure comprehensive and relevant studies were captured.

The requests are the conjunction of 2 parts:

- The AI-related method terms: "artificial intelligence" OR "machine learning" OR "deep learning" OR "reinforcement learning" OR "LLM".
- The HPC goal aimed: "HPC" OR "software performance" OR "accelerator".

The following exclusion criteria were applied:

- Papers published before 2018. While valuable, these works were considered out of scope due to the rapid advancement of AI capabilities and the ongoing evolution of HPC architectures toward the exascale era.
- Papers lacking HPC- or AI-related keywords. Papers that did not mention these terms in their title or abstract were considered outside the scope of this review.
- "HPC for AI" papers, which are outside the scope compared to "AI for HPC". In some cases, reading the full article was necessary to determine the research question addressed.
- Papers on AI for micro-architecture design. This includes studies on chipset placement, logic synthesis, or quantum gate compilation. By contrast, such works fall within the scope of HPC if they address how AI improves the design of interconnected computing units or the synthesis of parallel computing architectures. This constraint allows the analysis to remain focused on operational aspects, rather than diverging into low-level physical design of computers.
- Only peer-reviewed documents were retained. Non–peer-reviewed sources may be cited for complementary discussion of trends or methodology, but were excluded from the core set of 74 reviewed papers.

In this study, approximately 1,800 papers were manually reviewed, focusing on the intersection of AI and HPC. Among these, 823 were directly related to this intersection.

Approximately 91% of the papers were centered on "HPC for AI" (around 749 papers), while only 9% were focused on "AI for HPC" (74 papers), with minimal overlap between the two categories due to aiming at different goals. It is acknowledged that the specific focus of this review may be on a more limited subset of the HPC/AI area, but the potential impact of this research is significant. "HPC for AI" generally improves the HPC utilization at the scale of a specific application set, and "AI for HPC" focuses on improving the overall HPC infrastructure to support a broader range of applications. The growing number of AI applications utilizing HPC, the exponential increase in demand for computational resources, and the environmental concerns related to HPC systems make research in this area increasingly urgent.

This review was conducted manually after finding that machine learning-based approaches were insufficient for accurately categorizing scientific literature or extracting meaningful insights. Public LLMs, such as ChatGPT, often exhibit bias and produce out-of-context interpretations due to the predominance of "HPC for AI" publications over "AI for HPC" in the current literature. Additionally, machine learning methods that rely on word frequency and conventional search engines necessitate manual analysis to accurately classify documents, particularly in distinguishing between "HPC for AI" and "AI for HPC". Scientific impact was also prioritized by considering journal and conference metrics as well as document citations per year.

Finally, to disambiguate papers spanning multiple subfields, an explicit tie-breaking rule was applied: classify by the last AI component in the end-to-end workflow (the "terminal-component rule"). For example, for a pipeline comprising (i) an AI-based performance estimator, (ii) an AI-driven performance optimizer, and (iii) an AI-enhanced scheduler, the paper is assigned to "AI scheduler". This convention yields unique and reproducible assignments. The common synergies between subfields are the topic of Section 6.2.

4.2. Methodology limitations

Some limitations characterize the methodology. First, despite efforts to systematize the process, it is not fully automatic due to varying query capabilities and the maximum number of entries per page across digital libraries. Second, cross-validation was performed among co-authors to mitigate classification error or bias, but complete elimination of interpretative variance is not feasible. Third, the continuous evolution of the literature implies that a reproduction of the same search protocol at a later time may yield different results. This limitation is intrinsic to any snapshot-based survey but remains important to recognize in the context of a rapidly developing domain such as AI for HPC.

4.3. Overview

Among the papers that were collected addressing the research question, the aim was to identify appropriate groups to categorize the findings.

To provide a more detailed and organized analysis, the "AI for HPC" research was further divided into 6 categories:

- **AI for estimating the computing performance of a workload**. Papers in this category focus on leveraging AI to predict key performance metrics for jobs running on HPC systems, such as job duration, memory consumption, CPU usage, and overall resource demands. Accurate predictions of these metrics are crucial for efficient job scheduling, resource allocation, and system management. AI models can analyze historical data and job characteristics to provide detailed job profiles, which can then be used to anticipate how long a job will take to complete or how much memory it will require. These insights help optimize the scheduling of jobs, avoid resource contention, and improve the overall throughput of the system. Additionally, these predictions can inform other algorithms within the HPC environment, such as those used for load balancing, fault tolerance, or energy management.
- **AI for computational performance optimization of applications**: Papers in this category focus on using AI to enhance the computational performance of applications running on HPC systems. This involves optimizing various aspects of the application, such as computing speed, efficiency, and resource utilization. AI-driven techniques can automatically adjust application parameters, select the most suitable run-time libraries, and fine-tune compiler settings to maximize performance. By analyzing the specific needs of an application and the characteristics of the underlying hardware, AI can identify bottlenecks, predict optimal configurations, and adapt the execution environment dynamically.

- **AI for Job/computer node Scheduling**: Papers in this category focus on using AI to optimize the scheduling of jobs on computing resources. In HPC, a scheduler is responsible for managing the queue of jobs that need to be executed on the available compute nodes (the individual units within an HPC system that perform the computations). The scheduler's role is to allocate these jobs to the compute nodes in a way that maximizes resource utilization, reduces job wait times, and balances the workload across the system. AI techniques can enhance this process by predicting the most efficient allocation of resources, dynamically adjusting schedules based on real-time conditions, and minimizing idle time on the compute nodes.
- **Machine Learning Surrogate Models for Long-Running Job Prediction**: Papers in this category investigate the use of machine learning surrogate models to predict the outcomes of long-running and computationally intensive jobs in HPC environments. These surrogate models act as simplified representations of the actual computational processes, allowing for faster computations without needing to run the full, resource-intensive HPC computations.
- **AI for Fault Detection**: This research area focuses on the application of AI techniques to identify and diagnose faults within the HPC infrastructure or distributed software systems. Faults in such environments can range from hardware malfunctions, such as failing compute nodes or storage devices, to software bugs or misconfigurations. AI models are trained to detect anomalies by analyzing vast amounts of operational data, including system logs, performance metrics, and sensor data, to identify patterns indicative of potential failures. Beyond just detecting faults as they occur, some studies advance the field by using AI to predict and anticipate faults before they happen, based on early warning signals, and some others attempt to provide a diagnosis of the fault. This proactive approach enables preemptive maintenance, therefore minimizing downtime and improving the overall reliability and resilience of the HPC system.
- **Language Model (LM) for Automation**: This area focuses on leveraging LM techniques to automate scripting, coding, and other text-based tasks within HPC environments. LM-powered tools can assist various communities working in HPC centers. For instance, HPC administrators can use these tools to automatically generate scripts for system management, monitoring, and configuration tasks, reducing the manual effort required for routine operations. HPC researchers benefit from LM by using it to scrape, organize, and summarize multidisciplinary research articles, which helps them stay updated with the latest developments across various fields. Meanwhile, HPC scientists can utilize LM to streamline the development of complex codes, including those that employ distributed computing frameworks. By automating these tasks, LM enhances productivity, reduces errors, and allows HPC professionals to focus on more strategic and innovative aspects of their work.

Fig. 5 illustrates the distribution frequency of the 74 recent papers into 6 main categories addressing "What can AI do for HPC?". AI for performance estimation is grouped into a single category due to its extensive coverage and debatable split of some papers between infrastructure (HPC jobs computing time prediction) and the application-level view (specific application computing time prediction).

To complement the quantitative overview of the research output and its impact, the following statistics have been gathered. According to SJR Scopus data, of the 74 papers analyzed, 38% are published in journals and conferences beyond the H-100 index, 40% fall between the H-100 and H-10 range, 10% are cited below H-10 but remain indexed, and 12% are not indexed. The number of papers published between 2024–2023 appears to be relatively stagnant compared to those published during 2022–2020, indicating no clear trend among

the six identified areas. However, the dataset of papers may be too limited to reliably identify or confirm such trends.

In the next section, the articles are sorted by publication date to highlight the progression and recent advancements in the field.

4.4. AI for performance estimation

Computational performance encompasses various metrics, such as computing time and resource utilization. Accurate performance prediction before executing experiments in HPC is critical for avoiding the submission of inefficient or poorly performing workloads (e.g., anticipating resource exhausted error (Gao et al., 2020)), inform a scheduler based on estimated completion times (Bharti and Ranjan, 2019) (e.g., Shorter-Job-First, Shortest-Queue-Time heuristics), informing of the infrastructure load (Vercellino et al., 2023; Menear et al., 2024). AI methods, particularly machine learning and deep learning, have demonstrated significant promise in accurately predicting these performance metrics. The Tables 1 and 2 review several key studies that have applied AI techniques to performance estimation in HPC environments.

Performance estimation may play a pivotal role in making go/no-go decisions before executing computing tasks. It raises essential questions such as: is the workload worth the computing time, budget (AIPACA MLOps, 2022), and power consumption? Moreover, understanding whether a model can effectively fit the data, as discussed in Gao et al. (2020), can introduce uncertainty, leading to time wasted in trial-and-error processes and delays caused by unnecessary memory allocation and crash report diagnostics.

Given the high dimensionality and nonlinearity involved in characterizing workloads, supervised ML and unsupervised ML methods can effectively leverage data from previously submitted jobs to make predictions. A notable challenge researchers face is predicting computing jobs with significantly varying durations, ranging from seconds to days. Although this is fundamentally a regression problem, some discretize predictions into ranges (Bharti and Ranjan, 2019; Vacchetti et al., 2024), and some Vercellino et al. (2023) cascades both a classifier for range selection and regressors for finer-grain prediction.

Some authors tackle the complexity of performance estimation by focusing on uncertainty estimation (Vercellino et al., 2023; Menear et al., 2024). Understanding these uncertainties can lead to more robust scheduling and resource allocation strategies by providing decision-makers and algorithms with optimistic and pessimistic value ranges.

Finally, interpretable AI methods, such as those utilizing SHAP (Lundberg and Lee, 2017) (Shapley Additive exPlanations), enhance the understanding of the contributions of input features to the model's predictions (Gebreyesus et al., 2024). This interpretability is crucial for refining performance estimation models and gaining insights into the factors influencing computational efficiency.

Work with performance estimation as the main objective accounts for 19% of the papers collected in this study. However, AI performance estimation allows for extracting key performance characterization needed by AI Scheduler, AI anomaly detector, and AI task optimizer computational tasks.

4.5. AI for performance optimization

Performance optimization in HPC involves improving the efficiency and effectiveness of computational tasks. Research in AI for optimization is divided according to the goal: general-purpose software and deep learning applications. General-purpose software optimization includes tuning compilers, runtime environments, and application parameters. Deep learning applications require optimizing batch sizes, determining the optimal distribution of tensor computations across accelerators, and reducing communication overhead between accelerators.

Performance optimization is the goal of 19% of the papers collected in this study. Tables 3 and 4 review several key studies that have applied AI techniques to performance optimization in HPC environments.
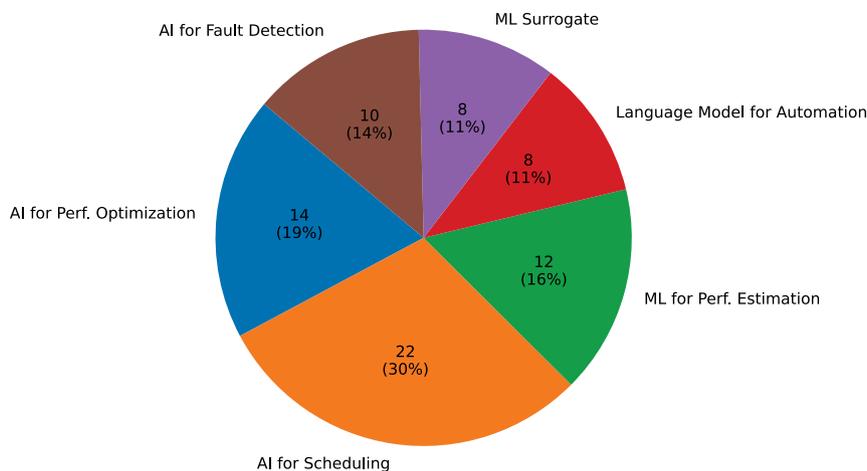
Fig. 5. Distribution of 74 recent papers identified answering "What AI can do for HPC systems?" into 6 main categories.

Table 1
Studies on AI methods for performance prediction in HPC between 2019 to 2022.

Ref	HPC goal	AI method	Data source	Outcomes
Bharti and Ranjan (2019)	Anticipating CPU utilization utilizing HPC Scheduler data	Ensemble of ML classifier	Generated with LSF, 10,000 entries, features: user id, resource requirement: CPU cores, memory	N/A
Pham et al. (2019)	Classifying jobs for future scheduler	LSTM (Hochreiter and Schmidhuber, 1997)	Synthetic jobs (with more than 500 runs)	100% accuracy on short jobs, 96% on long jobs, report challenge in tuning hyperparameters
Gao et al. (2020)	GPU memory consumption	Deep learning regressor	5 models with different hyperparameters under 3 frameworks (TensorFlow, PyTorch, and MXNet)	Average errors of 11.8% (TensorFlow), 13.85% (PyTorch), and 8.9% (MXNet)
Weihua Liu et al. (2021)	Ensemble to predict DSP software performance based on code	Ensemble of ML regressor	The execution time of a set of functions selected from PHY DSP Benchmark and run them on TIC64 DSP processor	Average absolute relative prediction error is less than 10%

The collected research addresses three complex optimization problems: multi-objective computational performance and application-specific score functions (e.g., Menon et al. (2020) and Pochelu et al. (2022)) where AI generates objectives to follow (e.g., validation score, uncertainty measurements). Optimization of ensembles of medium-sized tasks (e.g., Ward et al. (2021), Peterson et al. (2022), Guan et al. (2020), Bian et al. (2021) and Pochelu et al. (2022)) or large models distributed over multiple accelerators (Zheng et al., 2022; Li et al., 2023), where AI allows tuning the distribution of computations, accelerator micro-code performance optimization (e.g., Menon et al. (2020), Kaltenecker et al. (2020) and Willemsen et al. (2024)).

4.6. AI for scheduling

HPC scheduling involves the allocation and management of computational tasks on HPC resources to maximize the utilization of computational resources and minimize wait times. This includes deciding the order and timing of jobs to optimize resource utilization and performance. AI methods, particularly machine learning and reinforcement learning, have been increasingly applied to improve HPC scheduling, addressing challenges such as large input/output sizes, dynamic states, and the need for accurate predictions.

AI for scheduling constitutes 30% of the papers collected in this study. Tables 5, 6, 7 review several key studies that have applied AI techniques to scheduling in HPC environments.

Out of the collected papers in this category, half (Fan and Lan, 2021; Fan et al., 2021; Zhang et al., 2020; Wang et al., 2021; Fan et al., 2022b; Yang and Shen, 2022; Li et al., 2022; de Freitas Cunha and Chaimowicz, 2023; Koslovski et al., 2024) highlight the potential of Reinforcement Learning (RL) to transform automatic scheduling by outperforming traditional heuristics such as First-Come-First-Serve (FCFS). RL algorithms can learn from historical patterns, including the arrival times of previous jobs, job characteristics (such as CPU cores required, memory requests, and reservation times), and the current state of the infrastructure (including available memory and CPU cores). By leveraging this information, RL can adaptively optimize scheduling decisions, leading to more efficient resource utilization and improved performance in HPC environments.

However, the challenges in applying AI to HPC scheduling are multifaceted. Large input and output sizes and the dynamic nature of input/output characteristics can hinder the learning process. RL techniques face significant hurdles in implementation due to the large-scale nature of input/output. The modeling of the dynamic learning should support the dynamic dimensionality of the action space and state (e.g., adding/removing a server should not lead to retraining a new agent). Those systems based on RL often rely on the need for precise time predictions from other ML estimators and an accurate simulator to be trained before being deployed in real infrastructure.

Simpler scheduling techniques informed by machine learning have shown promise and are easier to integrate into HPC schedulers such as SLURM, PBS, and LSF. Many employ greedy algorithms enhanced with ML predictions for job characterization (Hu et al., 2021; Li et al., 2021; Betting et al., 2023) or comparisons between similar jobs and hardware to infer computation times (Thanapol et al., 2023). Additionally, some research focuses on energy-driven scores to schedule workloads, as seen in Syrigos et al. (2023).

Finally, interpretability in AI (Gebreyesus et al., 2024) using algorithms like SHAP allows for a better understanding of the contribution of input values to the scheduling decisions, which is critical for trust and transparency in AI-driven scheduling systems.

Table 2
Studies on AI methods for performance prediction in HPC between 2023 to 2025.

Ref	HPC goal	AI method	Data source	Outcomes
Cengiz et al. (2023)	GPU code performance prediction	Bespoke deep learning regressor	SPEC 2017 benchmark	R2 score 0.98
Vercellino et al. (2023)	Jobs queuing time prediction	PCA-based features aggregation, Features selection, classifier/regressor for time prediction, Bayesian Neural Network classifier, aleatoric/epistemic uncertainty	34 days of data collected from PBS, 1 year and 8 months on the KIT FHII System	Challenges: context shift, intrinsic data noise
AIPACA MLOps (2022)	Cloud computing deep learning task cost	Deep learning regressor	The tool contains random neural network sampling	N/A
Gebreyesus et al. (2024)	XAI for HPC Operations	HPC operating values, SHAP (Lundberg and Lee, 2017)	Historical dataset from onboard sensors: 12.5M samples, 26 features from energy and data. 310k samples, 9 features from the cooling system. 35.5k samples 22 features from environmental data	Global feature impact analysis, waterfall and force plots for local and global prediction, and dependency plots to capture feature interactions
Vacchetti et al. (2024)	ML classification to estimate jobs computing time, data drift estimation	XGBoost (Chen and Guestrin, 2016) classification, K-means	200 jobs executed using Altair Radioss solver, 150 jobs executed using the Altair Abaqus solver	Integrated tool within the Altair Access Portal, Frechet Inception distance measurement
Menear et al. (2024)	Quantifying uncertainty in job queue time predictions	XGBoost (Chen and Guestrin, 2016)	2.2 million jobs submitted to Eagle supercomputer operated by the National Renewable Energy Laboratory (NREL) in 2023	Interval Conformance Rate of 16 cluster partitions is 0.646
Hu et al. (2024)	Device-model agnostic framework for latency prediction of tensor programs	Representation of tensor named as Compact Abstract Syntax Tree, Transformer (Vaswani et al., 2017), KMeans-based sampling, Central Moment Discrepancy regularization	TenSet (Zheng et al., 2021) + custom collection of 120 deep learning. Experiments on 5 GPU models	14.03% prediction error for cross-model. 10.85% prediction error for cross-device prediction. Available open source.
Bai et al. (2025)	Multimodal task resource prediction	Informer (Zhou et al., 2021), LSTM, GNN	12 months production data sampled every 5 min. Input: CPU util., mem. usage, storage, network, task exec. time, task queue length, node temperature, system-level failure, restart events	Achieved up to 89.9% accuracy on runtime, CPU, memory, and storage predictions

4.7. ML surrogate

ML surrogates consist of machine learning models that approximate the behavior of more complex and computationally expensive simulations. These surrogates have emerged as powerful tools for avoiding unnecessary computation. They are especially useful within the HPC domain, where long and expensive computations can be avoided with a supervised learning approach.

Table 8 summarizes works found in this field.

Surrogate models have been extensively used as replacements for physics-based computing, such as in molecular dynamics (Kadupitiya et al., 2019; Radaideh and Kozlowski, 2020) and computational fluid dynamics (Sarma et al., 2024). They are also increasingly adopted to accelerate deep learning workloads, including hyperparameter tuning (Hu et al., 2023a), training (Ren et al., 2024; Hu et al., 2023a), and inference (Miao et al., 2024). Furthermore, surrogates serve as efficient substitutes for computationally expensive code patterns in scientific applications (Dong and Ren, 2023).

Recent AI techniques have significantly advanced the development of surrogate models. Convolutional autoencoders enable the extraction of meaningful features from high-dimensional data (Wang et al., 2016), while diffusion models support data enhancement tasks such as denoising and super-resolution (Shu et al., 2023; Sohl-Dickstein et al., 2015). Deep Gaussian Processes provide a principled approach to uncertainty quantification in surrogate modeling (Radaideh and Kozlowski, 2020). Additionally, transfer learning techniques, such as zero-shot hyperparameter transfer, generalize surrogate performance across tasks and model scales, offering efficient surrogates for deep learning computations (Yang et al., 2021).

4.8. AI for fault detection

Fault Detection or Anomaly Detection consists of automatically detecting abnormal behavior in monitoring data. Recent advances hold the potential to absorb different data and different levels of desired supervision according to the availability of manual labels. The Tables 9 and 10 summarize this body of research.

Most of the collected papers aim to detect a fault in multiple variables produced by the HPC system (network, CPU, power consumption, ...). Some others are focused on a particular element, such as communication traces (Alawneh and Hamou-Lhadj, 2022) or disk failures (Wu and Guan, 2024).

We identify the author model with different levels of supervision:

- **Supervised multi-class classifier** (Mohammed et al., 2019) allows to fit a model on previous data to detect coarse-grained anomalous patterns (hardware, software, network, human error). It relies on monitoring HPC logs and associated manual labels to tag the different types of errors.
- **Supervised binary-classifier** (Molan et al., 2023a, 2024; Wu and Guan, 2024). This method trains a model to detect anomalous patterns with binary labels ("abnormal" and "normal").
- **Semi-supervised** (Borghesi et al., 2022) is used when a large amount of unlabeled data is available, with a small subset containing binary labels.
- **Offline Unsupervised Anomaly Detection** (Molan et al., 2023b) is employed when large amounts of data without anomalies are available to learn before anomaly detection is deployed. No anomaly patterns are provided for learning, instead, it identifies statistical deviations from normal behavior.

Table 3

Studies on AI methods for performance optimization in HPC between 2020 and 2022.

Ref	HPC goal	AI method	Data source	Outcomes
Menon et al. (2020)	Tune compiler, run-time, application	Bayesian optimization, Transfer learning	Performance code including Kripke, Hypre, LULESH, and OpenAtom	50% fewer samples for Kripke in comparison to the compared approach, 2x the number of good configurations for LULESH.
Guan et al. (2020)	Optimization of ML ensembles training on HPC	Greedy algorithm optimizing computer time	50 models derived from DenseNets and 50 from ResNets	Speed up the ensemble training by 1.12-1.92X over the default training method, and 1.23–1.97X over the state-of-the-art framework designed for homogeneous ensemble training
Zheng et al. (2020)	Optimize tensor computation speed on Heterogeneous system	Heuristic, machine learning, custom scheduling	12 different kinds of tensor computations with total hundreds of test cases	In average 1.83x performance speedup on NVIDIA V100 GPU compared to cuDNN; 1.72x performance speedup on Intel Xeon CPU compared to MKL-DNN for 2D convolution; 1.5x performance speedup on Xilinx VU9P FPGA compared to OpenCL baselines; 2.21x speedup on NVIDIA V100 GPU compared to the state-of-the-art.
Bian et al. (2021)	Deep learning task scheduling with online batch size optimization	Online evolutionary scheduling algorithm	64 GPUs on TACC's Longhorn supercomputers. AlexNet, ResNet, VGG, GoogleNet, Inception, and BERT with different numbers of samples. From 3.6k to 20k samples.	Job completion time reduced by up to 45.6% compared to Deep RL
Ward et al. (2021)	Optimization of an ensemble of electrolyte simulations design	Upper confidence bound, Proxy-ML	10^5 molecules (SMILES strings format) from the QM9 dataset	Scales to 65,536 CPUs and accelerates the discovery rate for high-performance molecules by a factor of 100 over unguided searches. Available open-source.
Peterson et al. (2022)	Ensemble of simulations for building surrogate models	Data collection tool for surrogate ML	N/A	To enqueue 40 million simulations in 100 s, with a 30-millisecond per-task overhead that is independent of ensemble size. Open source and in production.
Pochelu et al. (2022)	Optimization of ML ensembles inference on HPC	Greedy algorithm optimizing validation score and inference time	CIFAR100, ImageNet	Scalable number of GPUs to vary the inference throughput
Zheng et al. (2022)	Automatic deep learning parallelism	Dynamic programming, Integer linear programming	N/A	Alpa generalizes to models with heterogeneous architectures. Alpa achieves a 3.5x speedup on 2 nodes and a 9.7x speedup on 4 nodes. Alpa also generalizes to models without manual strategies and achieves an 80% linear scaling efficiency on Wide-ResNet with 4 nodes. Available Open-source and in production.

- **Unsupervised** clustering algorithms enable easy visualization of time series by identifying patterns and normal data trends. DBSCAN-based clustering methods, which include a "reject" cluster for anomalies (Anton et al., 2023), as well as online unsupervised anomaly detection techniques (Alawneh and Hamou-Lhadj, 2022), are commonly used for this purpose.

Finally, some studies adopt hybrid approaches that combine different modes of supervision. For instance, Molan et al. (2023b) suggests switching between different levels of supervision to balance between high-human intervention with high-prediction quality and no labels, but lower precision. Others, such as Molan et al. (2023a), use an ensemble of two different AI models predicting at the same time and combine their advantages: one is self-supervised training on a large quantity of unlabeled data, and the second is a supervised model trained on a smaller quantity of data but labeled by a human expert.

Various data structures can be ingested by AI models. Recurrent models, such as Long Short-Term Memory (Molan et al., 2023b) networks, have proven to be well-suited for extracting information from temporal data. More recently, Graph Neural Networks (Molan et al., 2024) have demonstrated success in processing graph data by learning from different interconnected components within a network. GNNs have shown superior performance in detecting anomalies in computer networks compared to traditional neural networks that handle tabular data, as they effectively capture neighborhood contextual information between components.

4.9. LM for HPC scripting/coding automation

Language Model (LM) may be used to automate various aspects of information systems such as virtual assistance, search engines, and document summarization. Those approaches are especially useful in the highly technical and scientific context of the HPC. Table 11 summarizes recent research in this field.

Below, the literature is summarized according to the different goals addressed in the research papers.

- **Information retrieval in big data**. The HPC application and infrastructures are gaining in complexity due to the heterogeneity of platforms. AI has been useful in detecting the trends in the thousands of research papers published every year (Heldens et al., 2020).
- **Cybersecurity Script Generation**. HPC environments, accessible through public networks, are often prime targets for a wide range of cyberattacks. To improve security, machine learning algorithms (Al-Jody, 2021; Lee et al., 2024) have been proposed to design and assist HPC administrators in writing firewall rules and scripts. They can analyze historical logs to identify patterns and vulnerabilities, enabling them to build new rules.
- **Parallel code generation for HPC operations and coding**. This approach involves automatically designing software or assisting HPC developers through automatic code generation (Nan et al., 2022; Munley et al., 2024). The advancement of these methodologies is particularly beneficial in the context of complex systems,

Table 4

Studies on AI methods for performance optimization in HPC between 2023 and 2025.

Ref	HPC goal	AI method	Data source	Outcomes
Luszczek et al. (2023)	Multi-task HPC applications optimization	DeepGP (Damianou and Lawrence, 2013), transfer learning, EGO algorithm	2 applications: DGEQRF routine of the PLASMA library, the DGEMM routine of the SLATE library	Better optima found on DGEQRF compared to OpenTuner, HpBandster, EGO-GP, LCM
Li et al. (2023)	Automatic deep learning parallelism	Greedy algorithm	GPT-2 model with 10 billion parameters on the Wikipedia dataset	2.76x training speedup on large-scale models. Open source and in production.
Liu et al. (2023)	Deep-learning intra-operator parallelism; activation checkpointing	Symbolic profiling with Torch.fx (Reed et al., 2022); Cluster Detector (models bandwidth/latency/topology); intra-op parallelism ILP solver adapted from Alpa (Zheng et al., 2022); activation-checkpointing solver (Beaumont et al., 2021)	N/A	GPT-2 14.5B: 0.824 PFLOPS vs. 0.728 (Megatron-LM) and 0.715 (3D tensor parallelism)
Willemsen et al. (2024)	Auto-tuning CUDA kernels on a set of modern GPUs	Methodology for GPU auto-tuning design	N/A	Challenge: It uses random search as a baseline, the search spaces must be well-formed
Oh et al. (2024)	Tensor optimization	Branch-and-bound	N/A	Compared to FasterTransformer, ExeGPT achieves up to 15.2x improvements in throughput and 6x improvements in latency. Overall, ExeGPT achieves an average throughput gain of 2.9× across twenty evaluation scenarios.
Yang et al. (2024)	Inference of multiple DNN	Multi-task learning addresses this problem by designing a multi-task model that shares parameters across tasks based on a single backbone DNN	N/A	Inference latency of multi-DNNs reduced by 1.1-3x. Available open-source.

Table 5

Studies on AI methods for scheduling in HPC between 2020 and 2021.

Ref	HPC goal	AI method	Data source	Outcomes
Corbalan et al. (2020)	Power accounting, energy control/optimization	Linear model (Brochard et al., 2011)	8 applications, 26 nodes at Leibniz Supercomputing. Input: iteration time, power, memory transition per sec., cycles per sec.	From +2% to +12% Gflops/watt. Tested on 6480 nodes. Open source and SLURM plugin
Li et al. (2021)	General improving the scheduling system, Auto-correction of the user-requested time	Online learning, KNN, Linear Regression	Custom HPC simulator HPCsim implemented based on real system workload inputs. SimGrid (Casanova et al., 2014).	Can be used to enhance primary prioritizing and backfilling methods without being restricted by specific scheduling methods
Fan and Lan (2021)	HPC jobs/Resource scheduling	RL framework for HPC (A2C, SAC, PPO)	CQSim simulator (Yang et al., 2013)	DRAS-CQSim encapsulates simulation environments, agents, hyperparameter tuning options, and different reinforcement learning algorithms. Open source.
Hu et al. (2021)	Analysis of job traces and time prediction	ML duration prediction, Quasi-Shortest-Service-First (QSSF)	Real-world job traces from SenseTime. publicly available.	QSSF service to improve the average JCT by up to 6.5× . Cluster energy saving service, which improves overall cluster utilization by up to 13%.
Fan et al. (2021)	HPC jobs/Resource scheduling	Hierarchical Neural Network, Policy gradient, Deep Q-learning	Real jobs : Theta at ALCF (ALCF, 2024) 122k (max 1 day length), Cori at NERSC (NERSC, 2025) 2.6M (max 7 days). Train on real and synthetic jobs.	Outperforming existing scheduling policies by up to 45%
Zhang et al. (2020)	HPC jobs/Resource scheduling	Actor-Critic, Kernel-based Neural Network	Publicly available Parallel Workloads Archive (Feitelson et al., 2014): SDSC-SP2, HPC2N, PIK-IPLEX, ANL Intrepid, Lublin-1 (Lublin and Feitelson, 2003), Lublin-2	Learned models perform stably even when applied to unseen workloads, making them practical for production use. Open source
Bian et al. (2021)	Deep learning task scheduling with online batch size optimization	Online evolutionary scheduling algorithm	AlexNet, ResNet, VGG16, GoogleNet, Inception V3, pre-trained BERT	ONES can reduce the average JCT by up to 45.6%, compared to state-of-the-art (DRL) methods. In addition to that, it allows preemption and elastic batch size. Open source.
Wang et al. (2021)	RL for Jobs scheduling	Proximal policy optimization, RNN-based remaining time predictor, kill policy	The Vienna Ab Initio simulation package (Hafner and Kresse, 1997)	Average Slowdown Time is lower 8.41 compared to state-of-the-art (DeepRM Mao et al., 2016) 9.83. The wait time for long jobs using RL is longer than DeepRM, and the wait time for short jobs is shorter, indicating that short jobs are more likely to be backfilled. Open source

Table 6
Studies on AI methods for scheduling in HPC between 2022 and 2023.

Ref	HPC goal	AI method	Data source	Outcomes
Fan et al. (2022b)	RL for Jobs scheduling	Hierarchical Neural Network	Theta at ALCF (ALCF, 2024) and Cori at NERSC (NERSC, 2025)	Outperforms the existing heuristic and optimization approaches by up to 50%. Open source.
Yang and Shen (2022)	Online scheduling of task batches on heterogeneous servers	Greedy algorithms, RL	Google cluster trace and real application environment of 3000 face detection tasks submitted randomly in batches using from 20 to 100 cloud servers.	The algorithm improves system gain (measured as the value of completed tasks minus system operation costs) by approximately 10% to 30% compared to state-of-the-art methods (optRA, OSSA, HAMO, DRL). Provides a worst-case performance guarantee.
Li et al. (2022)	Large-scale scheduling	Generative adversarial deep reinforcement learning, PPO-Clip, Rule-based scheduling	Lublin-256 (Lublin and Feitelson, 2003) (10k tasks), SWF archive (Feitelson et al., 2014): HPC2N (527k tasks), SDSC-BLUE (250k tasks), SDSC-SP2 (73.5k tasks)	Learned models can perform stably even when applied to invisible workloads
Gu et al. (2022)	GPU resource management platform for DL jobs	Regression model-based method for job resource requirement estimation to avoid users over-allocating computing resources.	CNN for MNIST dataset, NeuMF (Xue et al., 2017) with MovieLens 20M dataset, ResNet-50, Inception V3, VGG-16.	Accelerate the job execution speed by 18% on average and shorten the average job completion time (JCT) by 21% compared with the existing solution (Kubernetes, Kubeflow, and HiveD)
de Freitas Cunha and Chaimowicz (2023)	Online scheduling, Large-action space, varying sizes without retraining	Maskable PPO (Huang and Ontañón, 2022)	Smdp branch of the sched-rl-gym package, with the CompactRM-v0 OpenAI Gym environment	Outperforms learning models from the literature and classic heuristics. Robust to changes in workload and cluster sizes, showing transfer works with changes of cluster size of up to 10×. Open source.
Thanapol et al. (2023)	Deep learning training Jobs/Resource placement	Similarity-Based policy	21 common CNN on ImageNet, 1680 jobs. Features: workload characteristics (number of layers, parameters, FLOPS, …), number of GPUs	Reducing makespan
Syrigos et al. (2023)	ML Inference scheduling in HPC-Edge for Latency/Battery optimization	Greedy algorithm driven by MIPS/Watt	NITOS testbed part of the SLICES-RI	Overall energy efficiency of over 41.8% compared to the off-the-shelf K8s scheduler
Hu et al. (2023b)	Interpretable deep learning workload scheduler	Primo (Hu et al., 2022), Decision tree, GA2M (Lou et al., 2013)	3 production-level traces collected: 101K jobs with average duration 13k sec, 23.9k jobs with avg. duration 5.4 s, and 12.4k jobs with avg duration 25.5 s The first two are SenseTime trace and the third is Microsoft trace.	Reduces the average job completion time by up to 1.3x compared with state-of-the-art pre-emptive scheduler Tiresias. Open source.
Gu et al. (2023)	Elastic resource management for distributed deep learning	Greedy algorithm, buddy allocation	6 DL tasks: ResNet50, VGG16, InceptionV3, BERT, GPT-2, deep speech 2	Evaluation results on a 128-GPU cluster show that ElasticFlow can increase the number of jobs that can meet their deadlines by 1.46–7.65x compared to existing state-of-the-art solutions. Integrated with PyTorch. Open source.
Betting et al. (2023)	Resource-recommendation system for HPC applications in the cloud	Multi layer perceptron	8 Amazon EC2 instance types, 2 have a GPU available. 8 applications: 5 from ARCHER (Turner, 2019), 3 Tensorflow.	The optimal instance type was chosen in 90% of the cases for seven out of eight applications, scoring a Mean Absolute Percentage Error (MAPE) consistently below 20%.

driven by the recent development of diverse accelerators and their intricacies.

5. AI for HPC summary

A deeper literature analysis uncovers interdependencies between these objectives. AI techniques for performance estimation can be applied in various contexts such as workload scheduling, anomaly detection, and performance optimization. These applications benefit both HPC users (through optimization, performance estimation, and AI surrogates) and HPC administrators (through anomaly detection, automation of operations, and workload scheduling). By leveraging AI, both users and administrators can achieve more efficient, reliable, and effective HPC systems.

The future of High-Performance Computing (HPC) is being shaped by a paradigm shift toward heterogeneity, encompassing a broader variety of accelerators and CPU architectures. This trend makes efficient scheduling and diverse computation acceleration more critical than ever before.

5.1. Category-specific challenges

While several overarching barriers affect AI integration in HPC, each application domain presents unique challenges:

- **Performance estimation.** Model generalizability remains a major difficulty due to hardware and workload heterogeneity—models trained on one system often underperform on others. This limits their portability across HPC centers. Finally, a lack of standards for HPC workloads makes estimation and comparison difficult. This is not true in the pure deep learning task, due to their nature to be a directed acyclic graph of tensor computation, some estimator may be designed (Li et al., 2023). A universal prediction of software execution time is theoretically impossible in the general case, as it depends on undecidable properties of programs (e.g., the halting problem), so practical performance estimation must rely on statistical models and empirical domain-specific measurement.

- **Performance Optimization.** Several works motivate new optimization designs by pointing out the scalability limits of prior

Table 7

Studies on AI methods for scheduling in HPC in 2024.

Ref	HPC goal	AI method	Data source	Outcomes
Dakić et al. (2024)	Kubernetes jobs placement	ML-Based workload placement	3-tier web application scenario	Integration of various components: PDUs to track energy usage, remote management interfaces to track servers' health state, hardware testing score
Lubrano et al. (2024)	Reserve in advance resources for optimal job execution, based on duration, dependencies, and machine characteristics	Mix of Unsupervised and Supervised ML for predicting	3 independent probability distributions to emulate the job submissions distribution: exponential for arrival, multinoulli for the number of nodes, gaussian mixture for wall time	Optimized work-flow has a makespan of 16 min, compared to 46 min of the non-optimized one. Integration to SLURM
Koslovski et al. (2024)	Graph of jobs placement	Actor-Critic RL, heuristics selection among several: First-Come-First-Serve, Shortest processing time, and Shortest queue time, …	Randomly generated DAG. Scenario 1: The jobs (100 or 300) were generated using a normal distribution for defining the arrival time of each composing task, Scenario 2: Each DAG is composed of up to 10 tasks, and the DAGs are generated with an interconnection probability of 0.2 or 1.0 between DAG layers	Dynamically adapted the policies selection for different workload contexts, eventually outperforming the counterparts schedulers (FCFS, SQF, SPF, SAF, F1, F2, F3, F4, AC)
Wang et al. (2025)	Intelligent job scheduling and node allocation	Maskable PPO (Huang and Ontañón, 2022), hierarchical RL	Data from HPCSim (open source)	Outperformed 27 scheduling strategies; Reduced job wait time. Available open source.

Table 8

Studies on machine learning surrogate model.

Ref	HPC goal	AI method	Data source	Outcomes
Kadupitiya et al. (2019)	ML surrogate of MPI/OpenMP simulation	Neural network regression	The training dataset consisted of 4804 simulation configurations, with 2060 input parameters	ANN ANN-based regression model predicted Contact density (MSE = 0.0000718), midpoint density (MSE = 0.0002293), and peak density (MSE = 0.0002306). Simulation time (around 12 h) and ANN surrogate model (around 0.25 s). Open source.
Radaideh and Kozlowski (2020)	ML surrogate of high-fidelity multi-physics simulation	DeepGP (Damianou and Lawrence, 2013), variational inference, variance decomposition with Sobol indices, parameter screening with Morris method	4 custom datasets: input dimension from 9 to 91, training samples from 120 to 200	Small MAE and MSE. DeepGP also accurately captures the uncertainty (about 0.6%). DeepGP successfully explains 95% or more of the variance in all 12 outputs.
Dong and Ren (2023)	Replace code region with a faster ML model	Neural Network	LLVM instructions to generate a tree-based data dependency graph. Evaluated on 11 functions: 3 from numerical solvers, 5 from PARSEC (Bienia, 2011), 3 from Exascale computing project (Messina, 2017).	There is 1.89×–16.8× speedup with a harmonic mean of 5.50× across all three types of applications, compared with the application performance on CPU (using all 40 cores).
Hu et al. (2023a)	Optimize tuning workloads in both the job-level and cluster-level granularities	Model Shrinker, Maximal update parameterization (Yang et al., 2021)	Evaluated on 6 models: GPT-3XL, Transformer, MobileNetV3 Large, VGG-11, ResNet-18	It reduces tuning makespan by up to 78.5× compared with Ray Tune and no reduction in tuning quality.
Sarma et al. (2024)	Library for large-scale training of AI models for computational fluid dynamics	Convolution auto-encoder (Wang et al., 2016), Convolution Defiltering Model (Shu et al., 2023) based on Diffusion probability model (Sohl-Dickstein et al., 2015) for super-resolution	Turbulent boundary layer (Albers et al., 2019) simulation dataset (8.3 TB)	Training scaling efficiency of 96% with 3664 GPUs compared to the performance on 1 node. Mean drag prediction error of ≈5%. With the diffusion model: ≈4%.
Ren et al. (2024)	Training more efficiently dynamic neural networks (DyNN) (Han et al., 2022) by predicting access order of tensors	Three parallel multilayer perceptrons and an output layer	Training on 24,000 samples from six diverse models (Transformer, CNN, LSTM). Tested on 2000 samples from 8 models.	The proposed model outperforms unified virtual memory (UVM) and dynamic tensor rematerialization (DTR) by 3× and 2.1× respectively in terms of maximum batch size.
Miao et al. (2024)	Faster and smaller LLM to mimic larger LLM	Speculative, Tree-based predicted tokens, verification in parallel	Prompts/questions from: Chatbot Instruction Prompts (Palla, 2023), ChatGPT Prompts (Rashad, 2023), WebQA (Chang et al., 2022), Alpaca (Taori et al., 2023), and PIQA (Bisk et al., 2020)	SpecInfer outperforms existing LLM serving systems by 1.5–2.8× for distributed inference and 2.6–3.5× for offloading-based inference, while preserving generative performance. Open source.
Bogale et al. (2025)	Performance modeling	Polynomial regression, Least squares regression	RAJAPerf benchmark data (Pearce et al., 2024)	54% better than kNN with 33% less data

Table 9

Studies on AI for automatic fault detection in HPC between 2019 and 2022.

Ref	HPC goal	AI method	Data source	Outcomes
Mohammed et al. (2019)	HPC fault and Diagnostics	Supervised multi-class ML, Ensemble	Computer Failure Data Repository (CFDR) (Schroeder and Gibson, 2006) at NERSC from 2001 to 2006	90% accuracy
Alawneh and Hamou-Lhadj (2022)	Communication Fault	Akaike Information Criterion, Analytic Hierarchy Process, Z-score, Median Absolute Deviation	5 traces generated from three HPC systems: SMG2000 (Brown et al., 2000), AMG2013 (ASC, 2013), NAS BT (NAS, 1994). Traces generated with Score-P (VI-HPS, 2022)	Accurately identify execution phases based on the detected communication patterns, it is capable of identifying slow patterns within each phase, and determines whether the latency is caused by late senders or receivers. These inefficient patterns are categorized based on their severity and complexity levels to help an analyst make decisions on where to start the inspection. Open source.
Borghesi et al. (2022)	HPC fault	Semi-supervised, Auto-encoder deep neural network	The data set combines sensor measurements coming from a variety of HW sensors and program counters, information from the job scheduler (SLURM), and reports on the system availability and status updates collected by Nagios (Barth, 2008). Aggregated sampling of 5-minute interval: mean and standard deviation. 12 874 samples.	Anticipating the actual labels on average 45 min in advance. F-score is 85%
Molan et al. (2023b)	HPC fault	LSTM, Auto-encoder, reconstruction error	The dataset contains 462 features. 10 months of operations. Sampling every 15 min and each feature: minimum, maximum, average, and variance by Nagios (Barth, 2008).	The proposed model architecture achieves the highest AUC of 0.77 compared to 0.75, which is the highest AUC achieved by the dense autoencoders.

Table 10

Studies on AI for automatic fault detection in HPC between 2023 and 2025.

Ref	HPC goal	AI method	Data source	Outcomes
Anton et al. (2023)	HPC fault & monitoring	HDBSCAN, t-SNE (van der Maaten and Hinton, 2008)	3 million entries or events and 89 features (Example: external temperature, PDUs measurement. Power from compute, storage, network, cooling units. Workload intensity, power usage effectiveness, ...)	Three phases methodology: raw data cleaning, ML-based processing and visualization
Molan et al. (2023a)	HPC fault	GNNs, Binary classification	31 months of data	Compared to SoTa Recurrent Unsupervised Anomaly Detection (RUAD) (Molan et al., 2023b). 1.5 h ahead: AUC is 0.8826 against 0.8699 with RUAD. 72 h ahead: 0.6219 against 0.5639 with RUAD
Molan et al. (2024)	HPC fault	Graph Convolutional Network, rack-level and room-level GNNs	31 months. 15-minute sampling frequency. It uses ExaMon (Borghesi et al., 2023), which aggregates a wide set of telemetry data.	It comprises three main parts: The monitoring subsystem, the MLOps subsystem with Kubernetes, and the anomaly the GRAAFE GNN model. It achieves an area under the curve (AUC) from 0.91 to 0.78, surpassing the state-of-the-art (SoA), achieving AUC between 0.64 and 0.5. GRAAFE sustains the anomaly prediction for all the Marconi100 nodes every 120 s, requiring an additional 30% CPU resources and less than 5% more RAM w.r.t. monitoring only. Open source.
Wu and Guan (2024)	Disk Fault	Ensemble, SMOTE Oversampling	Data from Self-Monitoring Analysis and Reporting Technology (SMART) of the disk.	Compared with traditional methods, the F1-score of disk fault prediction is improved by 6%, and the model training time is also greatly reduced
Xia et al. (2025)	Node-level anomaly detection	HAC (Zhang et al., 2022), Transformer, Mixture-of-Experts (Xie et al., 2023), Silhouettes (Rousseeuw, 1987)	Two real-world custom datasets	F1-score > 0.876; 0.560 better than baselines; 45.69% training overhead reduction. Available open source.
Sencan et al. (2025)	Anomaly detection in contaminated data	VAE with iterative anomaly removal	Production HPC system logs, Hpas (Ates et al., 2019)	F1-score 0.88

Table 11
Studies on Language Model (LM) for HPC operations automation.

Ref	HPC goal	AI method	Data source	Outcomes
Heldens et al. (2020)	Summarization of scientific papers on exascale computing topic	Big Data workflow, T-SNE (van der Maaten and Hinton, 2008), Clustering	Scopus (Elsevier, 2018)	Methodology with three-stage approach: (1) the challenges and opportunities of exascale based on various landmark studies are identified. (2) Data-driven techniques are used to analyze the large collection of related academic literature. (3) Eight research themes are identified
Al-Jody (2021)	Supervised NN for cyber-attack	Neural network, anomaly detection	30,000 activities from the users. They contain the occurrence of the command ran and the occurrences	Toolbox of reusable Ansible roles and playbooks to configure a cluster software environment described by the freely available OpenHPC recipes. Open source.
Nan et al. (2022)	LM to design HPC workflow	LM for code generation HISyn (Nan et al., 2020)	HPC-Fair (Verma et al., 2021) methodology. 60 custom natural language queries: 17 data manipulation, 25 ontology-interactive queries, 18 combined queries.	80% synthesis accuracy. The interactive design achieves 95% accuracy for extracting ontology information inside the NL query
Ding et al. (2023)	Managing AI models and datasets for HPC, and data race detection	LLaMA-based model	DataRaceBench V1.4.0 (Chen et al., 2023) containing 177 C/C++ test programs and 166 Fortran test programs. Among these, 88 C/C++ and 84 Fortran test cases exhibit data races, while 89 C/C++ and 82 Fortran test cases are free from data races.	The proposed model exhibits the ability to retrieve and extract HPC-related datasets and models based on human expressions. Good performance compared to existing data-race detectors.
Godoy et al. (2023)	Automatic parallel code writing with diverse programming models, and diverse languages	OpenAI Codex (OpenAI, 2024), GitHub Copilot (GitHub, 2025)	Six fundamental kernels in HPC: HPC AXPY, GEMV, GEMM, SpMV, Jacobi Stencil, CG	An average of novice level, for the languages and kernels tested. There is a slightly higher quality in the automatically generated C++ and Python languages rather than in Julia and Fortran.
Lee et al. (2024)	ML classifier for Firewall Rule	KNN, Neural Network	4.6 billion samples with 289,320 instances of identified firewall rule-policy errors. 10 features: source IP (4 parts), destination IP (4 parts), source port, and destination port	The results indicate that the KNN and NN models exhibited an accuracy of 97%. Additional training and feature refinement led to even better improvements, increasing the accuracy to 98%
Munley et al. (2024)	OpenACC, OpenMP automatic test-suite generation	Retrieval-Augmented Generation (Lewis et al., 2020), ChatGPT (Brown et al., 2020), DeepSeek-Coder (Guo et al., 2024), fine-tuning, prompt engineering	1335 training examples, 351 test examples	Deepseek-Coder-33b-Instruct produced the most passing tests of any LLM: 170/351
Rahman et al. (2025)	Parallel code generation	Multi-agent LLM: code generation, performance estimation, web-search	75 LeetCode problems	14.6% average runtime reduction compared to Claude 3.5 Sonnet

optimizers. Another open challenge is the reuse of knowledge across tasks, addressed in Luszczek et al. (2023) through transfer learning and multitasking, where insights from one optimization process accelerate learning on related tasks. Finally, a third challenge lies in the growing complexity of the parameter space: optimizers must handle high-dimensional search spaces combining system- and application-level settings (Menon et al., 2020).

- **Scheduling.** First, scalability: many approaches, such as RL-based schedulers (Fan et al., 2022b; Li et al., 2022) show promising results in simulations, but have not been demonstrated at scale in production systems. Second, trust remains limited: ML-based schedulers often act as black boxes, making them harder to explain and less reliable than heuristics like SLURM. Finally, schedulers must balance multiple and sometimes conflicting objectives—such as makespan, queue waiting time, fairness, and energy cost (Hossain et al., 2025).
- **Fault detection.** Techniques often rely on black-box models, raising concerns about interpretability and trustworthiness (Kadirvel et al., 2013). Providing explainable anomaly scores and actionable diagnostics remains an open problem.
- **Surrogate modeling.** While effective in reducing computational cost, surrogates often lack systematic approaches to quantify uncertainty in high-stakes simulations (Menon et al., 2020; Luszczek et al., 2023). Moreover, there is no consensus on standardized

datasets or evaluation metrics, limiting transfer learning techniques and cross-study comparability.
- **Code and scripting automation.** General-purpose LLMs exist and enable comparison with research works in the HPC context. However, papers in this area are rare and pursue heterogeneous objectives (e.g., parallel code generation Rahman et al., 2025, testing Godoy et al., 2023, bug detection Ding et al., 2023, ...) without shared benchmarks across studies. There is no documented evidence of production-level deployment to date, and the reliance on simplified tasks (e.g., LeetCode problem sets Rahman et al., 2025) does not reflect the skills required for developing and maintaining large-scale HPC codebases in production

These domain-specific issues demonstrate that generalized integration of AI into HPC remains challenging unless each category first resolves its internal limitations.

5.2. Platform-related challenges

Challenge: Accelerator heterogeneity.

While the majority of GFLOPS in modern HPC systems are delivered by AI accelerators, primarily GPUs with SIMT architectures, the challenge of scheduling across heterogeneous AI accelerator types remains largely unaddressed. Recent studies on alternative architectures,

such as Graphcore IPUs with MIMD architecture, have demonstrated substantial performance gains for specific workloads, with speedups reaching several factors over traditional GPUs (Ltaief et al., 2023; Burchard et al., 2023). Despite this, current HPC systems often place the burden of accelerator selection, code adaptation, and performance experimentation entirely on users. In contrast, AI-based schedulers, similar to those emerging in cloud environments (Betting et al., 2023), could provide intelligent guidance by selecting the most suitable accelerator dynamically, leveraging workload characteristics, real-time system states, and historical performance profiles.

Challenge: MLops of AI-based scheduler.

"AI for scheduling" remains the most active subfield in addressing the broader research question, "What can AI do for HPC systems?". Furthermore, most AI models for performance estimation are designed specifically to assist scheduling. The complexity of AI workloads — due to architectural diversity, computational constraints, and workload variability — requires advanced scheduling techniques. Efficient resource allocation in AI infrastructures increasingly depends on intelligent workload characterization and predictive models that can adapt to evolving usage patterns.

Literature reviewed in Tables 5, 6, 7 shows that heuristic-based scheduling often leads to sub-optimal decisions compared to AI-based schedulers, with significant potential for cost and energy savings. However, AI-based schedulers have not been implemented in MLOps pipelines, primarily due to challenges such as the lack of standardized benchmarks, the complexity of integrating learning models into real-time scheduling systems, concerns about model interpretability and trust, and the overhead of maintaining and retraining AI models in dynamic production environments.

Challenge: Increasingly distributed workflows.

The growing demand for heterogeneous computing is enabling new forms of inter-center collaboration, with the goal of maximizing accelerator utilization by intelligently matching user workloads to the most suitable hardware resources. Recent research emphasizes the development of heterogeneous computing networks that go beyond isolated HPC centers. Initiatives such as Grid'5000 (Bolze et al., 2006) and SLICES-RI (Fdida et al., 2022) are pioneering the creation of a next-generation computing ecosystem that supports distributed HPC, AI acceleration, advanced networking, and shared storage infrastructure. SLICES-RI, a collaboration among 16 European countries, provides access to a diverse array of resources — including HPC nodes, AI accelerators, IoT devices, and networking equipment — designed to facilitate geographically distributed workflows and multidisciplinary research.

This shift brings both opportunities and challenges for scheduling, programming, and detecting anomalies in multi-site HPC systems, HPC/IoT integration, and bridging HPC with cloud platforms. As a result, future AI for HPC research must consider this increasing heterogeneity, not only in terms of processor capabilities, but also in communication bandwidth and latency between distributed processing units.

5.3. AI trends

AI methods in the context of HPC are rapidly evolving, with deep learning approaches making significant strides. Deep Gaussian Processes (DeepG) have proven effective for quantifying uncertainties related to performance optimization (Luszczek et al., 2023) and surrogate modeling (Radaideh and Kozlowski, 2020), enabling robust decision-making in high-dimensional nonlinear systems. Graph Neural Networks (GNNs) have emerged as a powerful tool for capturing computer information and their interdependencies in networked machines (Molan et al., 2023a, 2024). Additionally, innovative applications are being developed that leverage natural language processing to streamline the creation and scaling of complex parallel code (Nan et al., 2022; Godoy

et al., 2023; Munley et al., 2024), enhance cybersecurity scripts (Lee et al., 2024). Finally, actor-critic reinforcement learning has consistently outperformed heuristic approaches, as demonstrated by the studies in Tables 5, 6, 7 despite the challenge of handling large-scale action spaces required for assigning jobs to resources. Achieving strong results with a large action space has been made possible through techniques such as action masking (de Freitas Cunha and Chaimowicz, 2023) or by employing reinforcement learning to select optimal heuristics from a predefined pool rather than making direct predictions (Koslovski et al., 2024).

AI methods in HPC are advancing rapidly, with several key trends emerging. Natural Language Processing is being explored for a variety of HPC applications, such as generating kernel codes, creating tests, and orchestrating complex workflows (Munley et al., 2024; Godoy et al., 2023; Nan et al., 2022). Recent research focuses on replacing large LLMs with smaller, more efficient models for inference. While the autoregressive nature of LLMs presents challenges for parallelizing inference, studies have demonstrated a 2× speedup when substituting large models with smaller ones in specific scenarios (Miao et al., 2024). This highlights the potential for optimizing inference procedures to achieve greater efficiency.

Advances in AI have significantly improved the ability to interpret unstructured data, enabling better contextual understanding. Graph Neural Networks (GNNs) are emerging as a cutting-edge approach in anomaly detection, effectively capturing dependencies in computer networks and outperforming traditional models (Molan et al., 2024, 2023a). LLMs are also proving invaluable for consuming and generating code. Tools like DeepSeek-Coder (Munley et al., 2024) push the boundaries toward Artificial General Intelligence (AGI), automating the creation of code and tests via intuitive prompting.

Alongside model development, research is increasingly emphasizing Machine Learning Operations (MLOps) for HPC. These workflows focus on deploying AI predictions effectively and include tasks such as retraining models in production environments, conducting hyperparameter searches, and monitoring input data and predictions (Anton et al., 2023; Molan et al., 2024). Additionally, methodological efforts aim to establish standardization in AI workflows (Verma et al., 2021) and facilitate the collection and organization of scientific literature (Heldens et al., 2020).

Reinforcement Learning has consistently demonstrated superior performance in task scheduling when compared to traditional heuristics, such as First-Come-First-Serve with backfilling queues (Li et al., 2022; Fan et al., 2022b; Zhang et al., 2020). Despite this, heuristics remain a widely used approach in many studies due to their ability to explore large decision spaces and produce relevant decisions without requiring prior interactions or extensive training (Syrigos et al., 2023; Gu et al., 2023; Pochelu et al., 2022). These two families of methods — RL and heuristics — continue to coexist, with hybrid approaches seeking to combine the strengths of both to achieve more effective and adaptable solutions (Koslovski et al., 2024).

The future of AI in HPC likely involves the convergence of techniques like scripting/coding generation, scalable reinforcement learning, uncertainty estimation, and graph-based information extraction. Integrating these methods could significantly enhance job characterization and control, leading to more efficient and sustainable computing infrastructures.

5.4. Meta-challenges in the literature

HPC centers worldwide often operate with distinct priorities and goals, making it difficult to design universal AI solutions. For example, job scheduling may be formulated to minimize average job completion time, ensure fairness among users, reduce energy consumption, or lower infrastructure costs (Li et al., 2022). Similarly, anomaly detection may target different priorities, such as optimizing F-score metrics (Borghesi et al., 2022), enhancing explainability

Table 12

Mapping between traditional OS elements and LLM-OS elements reported from Ge et al. (2023b).

Traditional OS	LLM-OS
Kernel	Core LLM-OS
Memory	Context window
Memory management	Context selection and management
File system	Retrieval storage (Packer et al., 2024)
Applications/Libraries	Software agents
Driver/API	Prompts/Instructions
OS SDK	LLM OS SDK (Ge et al., 2023b; Mei et al., 2024)
User interface	Natural language prompt
Applications	Agent applications

(Molan et al., 2023b), and diagnosing root causes (Alawneh and Hamou-Lhadj, 2022). These diverse priorities necessitate that administrators can customize AI models to balance priorities and priority shifts.

AI solutions for HPC often rely on datasets derived from hardware sensors, job schedulers, and system logs. However, these datasets are frequently unique to specific HPC centers, complicating efforts to develop AI models that generalize across different environments (Mohammed et al., 2019). Increasing the size and diversity of data inputs can improve model performance, but also risks overfitting to the idiosyncrasies of a single HPC center (Borghesi et al., 2022). The lack of standardized data sources further hinders the creation of versatile AI solutions applicable across heterogeneous systems.

Privacy concerns related to HPC user activity present additional barriers to sharing datasets. While publicly available datasets (Schroeder and Gibson, 2006; Feitelson et al., 2014; Lublin and Feitelson, 2003) exist, they often fail to capture recent trends, such as GPU-accelerated workloads and AI applications. Many AI models are trained using proprietary data or simulators, limiting the reproducibility and collaborative advancement of AI research in HPC (Molan et al., 2023b).

The challenge of generalization also affects AI models trained on homogeneous HPC systems, as they often fail to perform well in heterogeneous environments with varied nodes and accelerators (Fan et al., 2022b). Similarly, HPC traces from one center may not accurately represent the demand patterns or scheduling requirements of other centers. For instance, centers with high AI workloads face high accelerator demand compared to those focused on traditional CPU-based tasks.

While reinforcement learning has demonstrated its potential in HPC scheduling (Li et al., 2022), fully integrated workflows remain rare. These workflows must encompass diverse components, such as retraining mechanisms to address data shifts, hyperparameter optimization, administrator-facing dashboards, and explainability of RL-based scheduling decisions. The development of such comprehensive solutions is an ongoing challenge in advancing AI applications for HPC (Koslovski et al., 2024).

In addition to technical challenges, the literature itself presents difficulties in synthesis and comparison. The level of experimental detail varies significantly across papers. Some studies offer comprehensive performance metrics, while others omit critical information such as dataset sources, validation protocols, or evaluation baselines. For instance, several methods report no dataset at all, either because no historical data was needed, the method remains untested, or the authors did not disclose it. Furthermore, the use of heterogeneous evaluation metrics (e.g., speedup factors, percentage reductions, accuracy scores) without standardization complicates direct comparison and meta-analysis. These limitations hinder the establishment of benchmark-driven conclusions and highlight the need for better reporting practices in the HPC field.

To address these challenges, standardizing data collection and developing common frameworks for HPC datasets are crucial. Initiatives aimed at creating shared resources can facilitate collaboration and improve the generalizability of AI solutions. Additionally, customizable

AI methods that align with the specific goals of individual HPC systems hold promise for overcoming the limitations posed by diverse objectives and environments (Verma et al., 2021).

6. Integration of AI models into HPC systems

This section outlines the LLM-OS concept, its potential to integrate multiple "AI for HPC" models seen in previous sections (performance optimizers, surrogates, schedulers, fault detectors, and code generators) into a unified framework, and the key challenges it presents.

6.1. LLM-based operating systems

Traditional operating systems (OS) manage hardware and software resources to ensure efficient computing. With the emergence of large language models (LLMs), a new paradigm is being proposed in which LLMs operate not just as applications but as part of the core OS—managing memory, processes, and external tools (Mei et al., 2024; Packer et al., 2024; Ge et al., 2023b). This concept transforms OS architecture by embedding reasoning, task planning, and contextual awareness directly into the system's core.

Modern OSs typically adopt a layered and modular architecture, comprising components such as networking, storage, peripheral device support, interpreters, and development toolkits. These structures promote modularity, extensibility, and maintainability—properties that are equally desirable when integrating LLMs into the OS kernel.

Recent advances in AI technologies, including language reasoning, computer vision, and speech recognition, have significantly expanded human–computer interaction. Voice assistants like Apple's Siri[1] and Microsoft's Voice Access[2] demonstrate early forms of natural language control. However, these systems typically operate at the application level. In contrast, LLM-based OS architectures propose a deeper integration of AI at the core system level, enabling AI-native process management, tool orchestration, and interactive user experiences.

LLM-based Operating System (LLM-OS) architectures leverage recent advancements in large language models (LLMs) to mimic several OS modules. They can integrate other agents (the applications running above the OS). These systems aim to unify and orchestrate previously trained AI agents while maintaining a natural language interface. Table 12 highlights the mapping between traditional OS components and those of a typical LLM-OS, highlighting the feasibility and modularity of LLM-OS with the advantage of being powered by AI for further flexibility and adaptability.

MemGPT (Packer et al., 2024) lays foundational work for LLM-OS memory management. It adopts a hierarchical memory model to preserve the most important information (user intent, user profile, current in-context document), inspired by traditional OS principles allowed through Retrieved-Augmented Generator (Lewis et al., 2020), enabling long-term context management through:

- **Main context:** a limited but fast in-context window.
- **External context:** long-term memory, typically stored on disk or in external databases (Lewis et al., 2020).

This abstraction allows LLMs to handle extended document analysis and maintain coherent multi-session interactions by dynamically swapping memory, similar to virtual memory in classic OS design.

Several models have been proposed (Ge et al., 2023b; Mei et al., 2024; Zhang et al., 2024) that extend this concept further by proposing a system where LLMs function as the OS kernel. Within this framework, both software and AI agents act as applications, dynamically orchestrated by the LLM core based on user input, available resources, and context.

---

[1] https://www.apple.com/siri/.
[2] https://www.microsoft.com/en-us/windows/tips/voice-access.
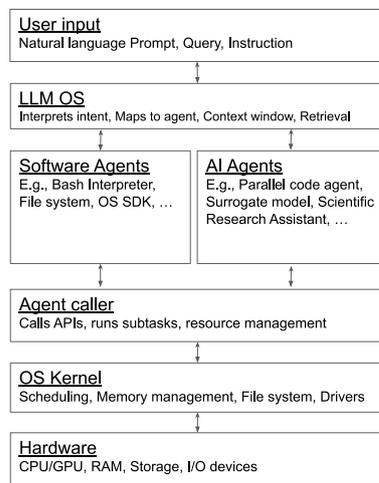
Fig. 6. Dataflow stack of LLM-OS inference from user intent to hardware.

Fig. 7. Synergies between the seven potential "AI for HPC" sub-domains. Each node represents a specific AI application area within HPC. The edges represent direct potential interactions and the information flows. Concrete examples of these synergies are discussed in the subsequent text.

A key feature of LLM-OS architectures is their ability to interpret a wide range of diverse user goals, decomposing them into a chain of thought (Wei et al., 2022) that translates high-level intents into a sequence of simpler tasks, such as API calls executed by agents (or AI or software agents) (Packer et al., 2024; Ge et al., 2023a). An illustrative example from Mei et al. (2024) involves a user planning a trip, which is broken down into software-level actions (e.g., uploading a photo ID, processing payments, calendar planning) and calls to specialized agents (e.g., retrieving user preferences, recommending flights and hotels).

These systems are described as containing the following components:

- LLM-OS: it interprets user intent and decomposes high-level goals into structured API calls. It requires access to agent APIs and, when needed, selects API calls demonstrations (Kong et al., 2024) that align user instructions with corresponding API calls.
- AI Agents: domain-specific models designed for tasks such as reasoning, data analysis, decision-making, and content generation.
- Software Agents: it is a wrapper for software, including desktop tools, software development toolkits (SDKs), interpreters, or simulators.
- Agent Caller: a scheduling and coordination module (also referred to as the AI-OS Kernel in Mei et al. (2024)). It manages agent execution and enforces access control. This component ensures safe operation by preventing unauthorized access and managing computational resources. It utilizes metadata such as resource requirements (e.g., compute time, memory usage, GPU availability), agent dependencies with versioning, and access rights to efficiently and safely schedule agent execution on the underlying hardware.

The Fig. 6 illustrates the dataflow of the inference of a standard LLM-OS described by Mei et al. (2024) and Zhang et al. (2024).

6.2. Opportunities

The true potential of AI in HPC lies not only in isolated innovations but in the strategic integration of complementary specialized AI models. LLM-OS interfaces provide a natural language layer that can unify control over these AI subsystems and collect and select the most relevant information generated by the agents.

Among the seven AI for HPC models and LLM-OS (not yet deployed for HPC), multiple synergies are identified and summarized in Fig. 7.

The conceptual links between subtopics are presented below with illustrative works. Software maturity is also discussed.
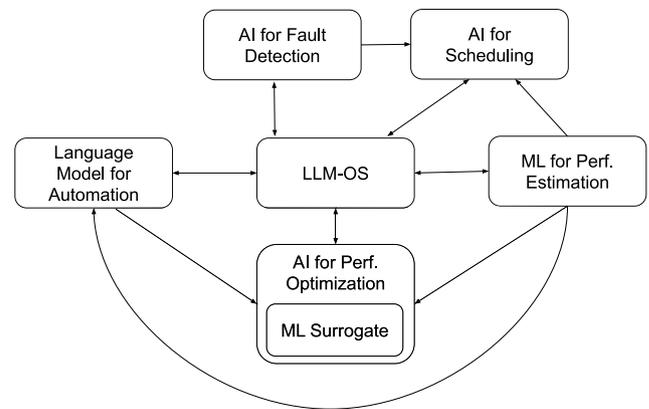
- **AI Performance Estimation → Scheduling**. Predicting job runtime, queue wait time, or power consumption is a proven way to improve HPC scheduling. By contrast, default systems such as SLURM often assume nothing about incoming jobs and rely on simple First-Come-First-Serve rules. Many workflows now combine estimators with schedulers. Some use supervised prediction to guide standard schedulers (Corbalan et al., 2020; Hu et al., 2021). Others link supervised estimators with RL schedulers (Wang et al., 2021). Hybrid designs combine supervised estimators with custom optimization policies (Hossain et al., 2025). In highly heterogeneous clusters, schedulers are often paired with recommender systems to guide job placement (Wyzykowski et al., 2024; Betting et al., 2023). A different line of work explores pure RL schedulers (Zhang et al., 2020; Fan et al., 2022b; Koslovski et al., 2024), which do not rely on explicit estimation. They can learn powerful strategies, but are usually less interpretable and remain mostly experimental. In practice, only the EAR system (Corbalan et al., 2020) demonstrates an estimator combined with a scheduler in production. Its plugin, *earplug.so*, integrates with SLURM and reduces power consumption using a supervised linear regression model (Brochard et al., 2011).
- **AI Performance Estimation → Performance Optimization**. Some performance optimization frameworks reduce sampling costs by incorporating data-driven performance estimation. For instance, Bayesian optimization (Menon et al., 2020; Luszczek et al., 2023; Willemsen et al., 2024) typically involves three phases: sampling, modeling, and searching. Transfer learning (Luszczek et al., 2023) can further accelerate this process by reusing knowledge from related tasks, thereby lowering the number of expensive evaluations needed. Among optimization systems deployed in production (Peterson et al., 2022, Alpa Zheng et al., 2022, and Colossal-AI Li et al., 2023), only Colossal-AI integrates an AI-based estimation component, Colossal-Auto (Liu et al., 2023), which performs prediction prior to optimization.
- **LM Code/Scripting Automation → Performance Optimization** The semantic link is straightforward between code/script automation in the HPC context and optimization. Most of the collected papers in this category aim at automation to best use resource and achieve performance, accelerate converge, or scalability. However, none of those papers' experiments have been assessed on a large scale.
- **Language Model → Performance Estimation + Performance Optimization**. LLM-based systems, such as MARCO (Rahman et al., 2025), perform joint tasks by generating optimized code/

scripts while also providing performance feedback. These models can estimate runtime from user-submitted code and constraints, and iteratively refine the code for better efficiency, thus bridging performance estimation and optimization in a unified automation loop.

- **AI for Fault Detection → Scheduling.** To improve the resilience of HPC systems, the fault detection mechanisms can be integrated into scheduling. A proactive approach can prevent workload failures by combining three key software components: monitoring, early fault detection, and preemption through task re-scheduling to healthy nodes. Such proactive fault-tolerant systems are well established in cloud computing (Mushtaq et al., 2024; Kadirvel et al., 2013), where virtualization facilitates VM migration (in case of hardware failure) or scaling (in case of resource exhaustion). Comparable frameworks in the HPC context remain rare in the literature.

- **ML Surrogate ⊂ ML Performance Optimization.** Surrogate modeling is a specific strategy within the broader field of performance optimization using machine learning. These models aim to approximate expensive computations in HPC workloads, enabling rapid evaluations during tuning or design space exploration.

- **LLM-OS and others** As outlined in Section 6.1, LLM-OS layers can serve as orchestration points of specialized AI models and components through shared knowledge. However, their application within HPC systems is still purely conceptual, as no empirical deployment or experimentation has yet been reported.

Despite these observed synergies, a major challenge remains: the lack of integration and standardization across AI components for HPC systems. As highlighted in the survey by Badia et al. (2024), the absence of standardized APIs, data formats, and interfaces impedes the seamless composition of AI models developed independently. This fragmentation undermines the reusability of components, the interface between user workflows and AI tools, the portability of AI solutions across domains, the reproducibility of experiments, and the ability to perform fair comparisons between models. While initiatives such as the FAIR principles, which promote workflows that are Findable, Accessible, Interoperable, and Reusable, have been proposed (Skvortsov and Stupnikov, 2022), their adoption remains limited. Consequently, even when technical synergies exist, practical integration is hard to achieve.

Furthermore, several conceptual links between AI tasks remain underexplored or missing. For instance, joint HPC code generation and runtime-time prediction LLM feedback loops are promising avenues but lack representation in current literature (Rahman et al., 2025). Some research, like SYNDICATE (Mahajan et al., 2023), proposes tightly coupled optimization of scheduling and execution planning, but its scope is limited to deep learning communication collectives, making generalization to all HPC workloads impossible.

A potential "ML for Performance Estimation" agent capable of providing accurate time and memory usage predictions could serve multiple roles. It could inform users about whether it is worthwhile to launch a job, assist the scheduler in deciding when and where to run jobs based on resource availability and expected load, and guide the code design and tuning process. These estimations could also be surfaced through LLM interfaces, enabling users to evaluate the cost-benefit ratio of launching a computing task.

However, a performance estimation agent is not always necessary to inform other AI models. For example, RL-based performance optimization often represents a model-free approach that bypasses prior estimation, learning instead through direct interaction with the environment. Similarly, some scheduling strategies or code tuning heuristics operate effectively using static rules or hardware-aware heuristics (e.g., cache hierarchies), rather than explicit performance predictions. Even in LM-based automation for code optimization, performance estimators are rarely used as intermediate models—most approaches rely on real

experimentations or hardcoded performance rules, though a hybrid system where estimators are used as reward signals for RL-based Language Model (LM) is a promising future direction.

The three subfields "ML Surrogate Models", "AI for Performance Optimization", and " Language Model for (Parallel Code) Automation", share a common goal: accelerating computation, albeit through different mechanisms. Surrogates aim to replace portions of numerical simulations (often with human-in-the-loop placement), optimizers fine-tune input parameters to maximize performance, and Language Model agents automatically generate or restructure code. Their functionalities are naturally interconnected.

For example, Language Model could generate code that is then further optimized by a performance optimizer (e.g., tuning message sizes, process-core mappings, or shared memory allocation). Conversely, profiling results from performance optimization tools could inform a Language Model code generator to revise suboptimal code segments. Furthermore, Language Model agents could identify recurring patterns in code (e.g., redundant functions or loops) and trigger supervised ML integration or surrogate replacement automatically.

ML surrogate models, designed to approximate costly simulations, also integrate with performance optimization workflows, since both aim to reduce the runtime or resource consumption of HPC applications. These interconnections support an emerging architecture of AI for HPC: a system of feedback-driven models that share data and reinforce one another. Whether through shared training pipelines, real-time inference loops, or coordination via LLM-based interfaces, these models amplify their impact when deployed in concert.

At the level of a single HPC job, similar synergies appear. Surrogate models can be embedded within applications to replace expensive computations, while performance optimizers can adapt parameters dynamically, informed by surrogate feedback. This allows faster iteration and reduced computational overhead in tuning complex workloads.

At first glance, several orthogonalities are also observed between AI models that necessitate no direct communication. In particular, models acting *within* jobs (such as performance optimizers and Language Model automation) address goals related to code execution and tuning, whereas models acting *outside* jobs (such as AI-based schedulers and fault detectors) operate at the system orchestration level. These models solve different problems and may evolve independently.

However, orthogonality in functionality does not imply isolation. Many of these models may consume overlapping contextual information, such as user behavior patterns, historical performance data, or hardware topology, and contribute outputs that are useful for other agents. A unifying, higher-level agent such as an LLM-OS can serve as an integration layer, allowing models to interoperate indirectly through shared knowledge representations. For example, a fault detection model might identify a degraded node and update a global status register. While the performance optimizer embedded could receive hints or modified execution recommendations from the LLM-OS, which aggregates infrastructure context. Similarly, a scheduler may use user or application profiles gathered during past optimization runs to make future scheduling decisions, even if those agents never directly interact.

This results in a complex flow of information, where some AI agents communicate directly, exchanging data and API calls, while others interact only indirectly by accessing and contributing to shared knowledge bases about users, applications, and hardware. The LLM-OS plays a central role in this architecture.

6.3. Challenges

LLM-OS Reasoning.

A core challenge in LLM-OS design lies in enabling effective interaction between language models and agents via API calls. This process demands a combination of Chain-of-Thought (CoT) reasoning (Wei et al., 2022), the ability to plan sequences of API calls, and the practical utilization of external tools. Decomposing user prompts into subtasks

using CoT has demonstrated strong performance in mathematical and symbolic reasoning tasks (Huang and Chen-Chuan Chang, 2023); however, it does not consistently improve outcomes across broader domains relevant to HPC, such as content generation, context-aware question answering, or tasks requiring expert-level knowledge.

Moreover, studies show that LLMs still struggle with planning and reasoning tasks that are trivial for humans (Huang and Chen-Chuan Chang, 2023). This highlights the need for more extensive fine-tuning (Kong et al., 2024), incorporating demonstrations of effective API usage and multi-step planning. For instance, evaluations on structured reasoning benchmarks reveal that even state-of-the-art models GPT-4 fail in abstract, multi-step planning tasks (Kambhampati et al., 2024).

LLM interactions.

Each AI agent (Han et al., 2024) must align its assigned tasks with overarching system objectives while also integrating contextual information from other agents. Ensuring consistency in shared goals and effectively leveraging distributed contextual knowledge remains a significant challenge in multi-agent LLM-OS environments. Recent research (Abbasiantaeb et al., 2024) shows that when two LLMs engage in cooperative problem-solving via iterative question-answer exchanges, the original objective often degrades over time. Lengthy prompt exchanges can cause the focus to drift, leading to less relevant or incoherent responses. However, emerging findings suggest that state-of-the-art models, such as GPT-4, are more resilient in maintaining coherence and goal alignment during extended interactions.

Effective communication in traditional HPC systems, between operating systems, applications, and orchestration layers, relies on standardized protocols and domain-specific languages (DSL) such as Terraform (Howard, 2022). Similarly, Large Language Model Operating Systems (LLM-OS) and AI agents are beginning to adopt standards for data and instruction exchange (Ge et al., 2023b). However, striking a balance between the expressive flexibility of natural language and the structural rigidity of DSLs remains a significant challenge. The inherently unstructured nature of LLM interactions complicates traditional security and compatibility frameworks, requiring robust oversight, validation mechanisms, and carefully designed interfaces to ensure smooth integration across heterogeneous platforms and versions.

System Integration.

In contrast to monolithic AI models, integrating AI models for HPC requires coordination between all aspects: updating the callable software/AI APIs, software, application data, AI model versioning and repository, and triggering of update of AI models and software when infrastructure changes.

The interaction between AI agents is far from easy. Even if multiple agents aim for the same goal (e.g., HPC job scheduling), their operational requirements may diverge significantly. For example, the collected tables of this article show that some AI-enhanced schedulers may require real logs, some synthetics from HPC digital twin, some consume the prediction of a performance estimator, and some consume application characteristics ("Data source column"). Some agents (e.g., job performance estimators, HPC fault detection) need frequent retraining when infrastructure and job profile changes, while rule-based AI models require no training phase.

Discrepancies in data sources, models' data sensitivity, and demand robust coordination and systematic MLOps workflows for potential integration. Recent studies are exploring best practices for integrating traditional software CI/CD pipelines with the machine learning and data lifecycle to develop comprehensive MLOps solutions (Liang et al., 2024). In particular, methods that detect data distribution shifts by comparing historical data with incoming data, using similarity metrics, can effectively trigger the re-training of models (Peixoto et al., 2025). However, the field of MLOps in the context of multi-heterogeneous agents is still relatively new.

They point to the growing necessity of standardization of MLOps practices and LLM-OS tailored for the large-scale nature of the HPC domain. Without such practices and systems, the promise of AI-enhanced HPC may be overshadowed by its operational overhead.

Debugging and Monitoring

Another critical challenge lies in the monitoring and reproducibility of AI agent interactions, particularly when those interactions span long timeframes or involve large, evolving datasets. As multiple agents exchange performance data, scheduling decisions, model predictions, and inferred states, the volume and variety of data traces quickly grow beyond what is human-auditable. Tracking who influenced what, when, and why becomes increasingly difficult.

This issue may become even more pronounced in scenarios involving interactions between two LLMs (Abbasiantaeb et al., 2024), such as in the case of an LLM-OS coordinating with an LLM for code generation or other specialized tasks. The current LLM has a tendency for lengthy interaction, which significantly increases the log size.

This finding suggests that for AI models and their interactions, maintaining effective communication and logging requires substantial storage and computational resources. These demands, in turn, raise questions about the overall cost-benefit ratio of integrating such complex systems. The trade-off between the performance and adaptability gains from AI integration and the resource overhead involved in managing these interactions becomes a critical consideration.

LLMS-OS Security

The cybersecurity of LLMs and MLOps is an active area of research, with ongoing efforts addressing both offensive and defensive aspects. An LLM-OS, like any LLM-based service, typically exposes a chatbot graphical user interface to its users. As with any IT system, it is susceptible to traditional cybersecurity threats (Hat, 2024) such as unauthorized access, denial-of-service (DoS) attacks, privilege escalation, and man-in-the-middle attacks. Additionally, it is exposed to LLM-specific risks, including misuse for generating spam, scams, misinformation, violations of safety guardrails (Welbl et al., 2021; Hackett et al., 2025), and potential data leakage (Kandpal et al., 2024). In addition, MLOps in high-performance computing (HPC) environments introduces further concerns due to the experimental and rapidly evolving software stacks used in AI workflows (Saxena et al., 2024) .

The risks of deploying LLM-OS are further amplified in HPC systems. HPC systems typically manage vast datasets and demand significant computational power, resulting in high operational and environmental costs. Consequently, a security breach or system failure can have wide-reaching implications for institutions relying on these infrastructures. Consequently, the security LLM-OS of HPC systems deserves further evaluations and mitigations.

7. Conclusion

This paper explores how various AI techniques can enhance High-Performance Computing (HPC). The analysis of 74 recent papers identified six key AI applications in HPC, including automatic job performance estimation, job scheduling, fault detection, performance optimization, surrogate models, and code/script automation through Natural Language Processing. Among these, job scheduling stands out as the most actively researched and in-demand area.

The key findings of this review are as follows: **Key findings:**

- **Performance estimation** is foundational, enabling informed scheduling, optimization, and even guiding code-generation agents.
- **Performance optimization** reduces runtime and energy while improving utilization across diverse workloads.
- **AI for job scheduling** is the most active topic, addressing distribution across increasingly heterogeneous resources. Approaches range from simulation-only research to production prototypes; while RL dominates, many systems combine supervised ML, recommendation, heuristics, and ILP.

- **AI-based surrogates** deliver substantial computational savings in simulations and DL, approximating HPC computations and, more recently, lower-level functions.
- **Fault detection and recovery** benefit from GNNs and time-series models (e.g., LSTM, Informer) that exploit spatial–temporal structure; most rely on minimal supervision but large production datasets.
- **Language Models for HPC automation** are advancing rapidly— LLMs enable test generation, code synthesis, and command-line automation. Models adapted to HPC contexts tend to outperform general-purpose ones on domain tasks.

The literature reveals interdependencies between several of these topics. For instance, AI-based performance estimators can support performance optimizers, inform job schedulers, and assist code/script generation tools. However, the integration of these individual AI applications into a unified and intelligent HPC framework has not yet been fully explored. This paper aims to contribute to that direction.

This analysis found that a particularly promising area of interest is the integration of Large Language Models (LLMs) into Operating Systems. This direction aims at more intuitive user interactions and allows the system to adapt dynamically to context, leading to more efficient resource utilization and potentially paving the way for the next generation of intelligent computing environments.

Despite its potential, integration raises several challenges:

- Lack of standardized metrics, datasets, and APIs hinders the integration and comparability of developed AI methods.
- AI components introduce new security concerns, particularly when integrated deeply into system-level infrastructure.
- Scaling MLOps workflows to accommodate frequent retraining and shifting data distributions.
- The live nature of data and models complicates debugging, reproducibility, and interpretability.
- Designing effective communication protocols among heterogeneous AI agents.
- Managing the computational and operational overhead introduced by logging, and coordination of large-scale AI interactions.

Given these challenges, the systematic benefits of such integration will need to be carefully evaluated in future research to determine whether the performance and adaptability gains justify the added complexity and risks.

HPC systems already enable major scientific breakthroughs in areas such as climate modeling, drug discovery, aerospace, automotive design, and energy optimization. By enhancing operational functions such as scheduling, anomaly detection, performance estimation and optimization, and surrogate modeling, AI can make these systems more efficient, reliable, and sustainable.

CRediT authorship contribution statement

Pierrick Pochelu: Writing – review & editing, Writing – original draft, Visualization, Validation, Supervision, Software, Resources, Project administration, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation, Conceptualization. **Hyacinthe Cartiaux:** Writing – review & editing. **Julien Schleich:** Project administration.

Declaration of competing interest

The authors declare that this manuscript has not been published previously, is not under consideration for publication elsewhere, and will not be submitted to another journal while under review by Engineering Application of Artificial Intelligence.

We have disclosed any potential conflicts of interest. There are no financial or personal relationships that could be construed as a conflict of interest.

We confirm that all data, models, and findings underlying the results are available upon reasonable request, and the manuscript adheres to the ethical guidelines. All necessary permissions for the use of any third-party material are obtained.

We agree to the terms and conditions of the Elsevier publication policy and confirm that all co-authors have reviewed and approved the final manuscript.

Data availability

No data was used for the research described in the article.

References

Abbasiantaeb, Z., Yuan, Y., Kanoulas, E., Aliannejadi, M., 2024. Let the llms talk: Simulating human-to-human conversational qa via zero-shot llm-to-llm interactions. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining. WSDM '24, Association for Computing Machinery, New York, NY, USA, pp. 8–17. http://dx.doi.org/10.1145/3616855.3635856.

AIPACA MLOps, 2022. ML training cost calculator. https://github.com/aipaca-mlops/ML-training-cost-calculator?tab=readme-ov-file. (Accessed 25 November 2025).

Al-Jody, T., 2021. Bearicade: A novel high-performance computing user and security management system augmented with machine learning technology. URL http://eprints.hud.ac.uk/id/eprint/35579/.

Alawneh, L., Hamou-Lhadj, A., 2022. Locating and categorizing inefficient communication patterns in hpc systems using inter-process communication traces. J. Syst. Softw. 194, 111494. http://dx.doi.org/10.1016/j.jss.2022.111494, URL https://www.sciencedirect.com/science/article/pii/S0164121222001704.

Albers, M., Meysonnat, P.S., Schröder, W., 2019. Actively reduced airfoil drag by transversal surface waves. Flow Turbul. Combust. 102 (4), 865–886. http://dx.doi.org/10.1007/s10494-018-9998-z.

ALCF, 2024. Alcf documentation. URL https://docs.alcf.anl.gov/. (Accessed 25 November 2025).

Anton, L., Willemot, S., Gougeaud, S., Zertal, S., 2023. Ml-based methodology for hpc facilities supervision. In: Bienz, A., Weiland, M., Baboulin, M., Kruse, C. (Eds.), High Performance Computing. Springer Nature, Switzerland, Cham, pp. 307–319.

ASC, 2013. Parallel algebraic multigrid solver for linear systems: The asc amg 2013 benchmark code. URL https://computing.llnl.gov/projects/co-design/amg2013. (Accessed 25 November 2025).

Ates, E., Zhang, Y., Aksar, B., Brandt, J., Leung, V., Egele, M., Coşkun, K., 2019. Hpas: An hpc performance anomaly suite for reproducing performance variations. pp. 1–10. http://dx.doi.org/10.1145/3337821.3337907.

Badia, R.M., Berti-Equille, L., Ferreira Da Silva, R., Leser, U., 2024. Integrating Hpc, Ai, and Workflows for Scientific Data Analysis: Report from Dagstuhl Seminar 23352. Tech. Rep. ORNL/TM–2024/3301, Oak Ridge National Laboratory (ORNL), Oak Ridge, TN, United States, http://dx.doi.org/10.2172/2341398, Sponsor Org.: USDOE Office of Science (SC), Advanced Scientific Computing Research (ASCR). Contract Number: AC05-00OR22725. URL https://www.osti.gov/biblio/2341398.

Bai, X., Zhou, J., Wang, Z., 2025. Hpc cluster task prediction based on multi-modal temporal networks with hierarchical attention mechanism. Computers 14 (8), http://dx.doi.org/10.3390/computers14080335, URL https://www.mdpi.com/2073-431X/14/8/335.

Barth, W., 2008. Nagios: System and Network Monitoring. No Starch Press.

Beaumont, O., Eyraud-Dubois, L., Shilova, A., 2021. Efficient combination of rematerialization and offloading for training DNNs. In: Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (Eds.), Advances in Neural Information Processing Systems. URL https://openreview.net/forum?id=BFYlnDtJSqW.

Betting, J.-H., Liakopoulos, D., Engelen, M., Strydis, C., 2023. Oikonomos: An opportunistic, deep-learning, resource-recommendation system for cloud hpc. In: 2023 IEEE 34th International Conference on Application-Specific Systems, Architectures and Processors. ASAP, pp. 188–196. http://dx.doi.org/10.1109/ASAP57973.2023.00039.

Bharti, P., Ranjan, R., 2019. Specifying cpu requirements for hpc applications via ml techniques. Int. J. Adv. Netw. Appl. 10 (5), 1–3, copyright - © 2019. This work is published under http://www.ijana.in/index.php (the License). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2023-11-25. URL https://www.proquest.com/scholarly-journals/specifying-cpu-requirements-hpc-applications-via/docview/2762022619/se-2 suppl.Special Issue.

Bian, Z., Li, S., Wang, W., You, Y., 2021. Online evolutionary batch size orchestration for scheduling deep learning workloads in gpu clusters. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '21, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3458817.3480859.

Bienia, C., 2011. Benchmarking Modern Multiprocessors (Ph.D. thesis). aAI3445564, USA.

Bisk, Y., Zellers, R., Gao, J., Choi, Y., et al., 2020. Piqa: Reasoning about physical commonsense in natural language. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34, pp. 7432–7439.

Bogale, B., Lumsden, I., Sukkari, D., Yokelson, D., Brink, S., Pearce, O., Taufer, M., 2025. Surrogate models for analyzing performance behavior of hpc applications using the raja performance suite. In: Lees, M.H., Cai, W., Cheong, S.A., Su, Y., Abramson, D., Dongarra, J.J., Sloot, P.M.A. (Eds.), Computational Science – ICCS 2025. Springer Nature Switzerland, pp. 327–335.

Bolze, R., Cappello, F., Caron, E., Daydé, M., Desprez, F., Jeannot, E., Jégou, Y., Lanteri, S., Leduc, J., Melab, N., Mornet, G., Namyst, R., Primet, P., Quétier, B., Richard, O., Talbi, E.-G., Touche, I., 2006. Grid'5000: A large scale and highly reconfigurable experimental grid testbed. Int. J. High Perform. Comput. Appl. 20, 481. http://dx.doi.org/10.1177/1094342006070078, 494.

Borghesi, A., Burrello, A., Bartolini, A., 2023. Examon-x: A predictive maintenance framework for automatic monitoring in industrial iot systems. IEEE Internet Things J. 10 (4), 2995–3005. http://dx.doi.org/10.1109/JIOT.2021.3125885.

Borghesi, A., Molan, M., Milano, M., Bartolini, A., 2022. Anomaly detection and anticipation in high performance computing systems. IEEE Trans. Parallel Distrib. Syst. 33 (4), 739–750. http://dx.doi.org/10.1109/TPDS.2021.3082802.

Brewer, W., Gainaru, A., Suter, F., Wang, F., Emani, M., Jha, S., 2025. Ai-coupled hpc workflow applications, middleware and performance. arXiv:2406.14315. URL https://arxiv.org/abs/2406.14315.

Brochard, L., Panda, R., DeSota, D., Thomas, F., Bell, R.H., 2011. Power and energy-aware processor scheduling. In: Proceedings of the 2nd ACM/SPEC International Conference on Performance Engineering. ICPE '11, Association for Computing Machinery, New York, NY, USA, pp. 227–234. http://dx.doi.org/10.1145/1958746.1958780.

Brown, P.N., Falgout, R.D., Jones, J.E., 2000. Semicoarsening multigrid on distributed memory machines. SIAM J. Sci. Comput. 21 (5), 1823–1834. http://dx.doi.org/10.1137/S1064827598339141, arXiv:https://doi.org/10.1137/S1064827598339141.

Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al., 2020. Language models are few-shot learners. Adv. Neural Inf. Process. Syst. 33, 1877–1901.

Burchard, L., Zhao, M.X., Langguth, J., Buluç, A., Guidi, G., 2023. Space efficient sequence alignment for sram-based computing: X-drop on the graphcore ipu. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '23, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3581784.3607094.

Casanova, H., Giersch, A., Legrand, A., Quinson, M., Suter, F., 2014. Versatile, scalable, and accurate simulation of distributed applications and platforms. J. Parallel Distrib. Comput. 74 (10), 2899–2917, URL http://hal.inria.fr/hal-01017319.

Cengiz, M., Forshaw, M., Atapour-Abarghouei, A., McGough, A.S., 2023. Predicting the performance of a computing system with deep networks. In: Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering. ICPE '23, Association for Computing Machinery, New York, NY, USA, pp. 91–98. http://dx.doi.org/10.1145/3578244.3583731.

Chang, Y., Narang, M., Suzuki, H., Cao, G., Gao, J., Bisk, Y., 2022. Webqa: Multihop and multimodal qa. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16495–16504.

Chen, T., Guestrin, C., 2016. XGBoost: A scalable tree boosting system. In: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '16, ACM, New York, NY, USA, pp. 785–794. http://dx.doi.org/10.1145/2939672.2939785, URL http://doi.acm.org/10.1145/2939672.2939785.

Chen, L., Wu, W., Siegel, S.F., Lin, P.-H., Liao, C., 2023. Dataracebench v1.4.1 and dataracebench-ml v0.1: Benchmark suites for data race detection. arXiv:2308.08473. URL https://arxiv.org/abs/2308.08473.

Corbalan, J., Alonso, L., Aneas, J., Brochard, L., 2020. Energy optimization and analysis with ear. In: 2020 IEEE International Conference on Cluster Computing. CLUSTER, pp. 464–472. http://dx.doi.org/10.1109/CLUSTER49012.2020.00067.

Dakić, V., Kovač, M., Slovinac, J., 2024. Evolving high-performance computing data centers with kubernetes, performance analysis, and dynamic workload placement based on machine learning scheduling. Electronics 13 (13), http://dx.doi.org/10.3390/electronics13132651, URL https://www.mdpi.com/2079-9292/13/13/2651.

Damianou, A., Lawrence, N.D., 2013. Deep gaussian processes. In: Artificial Intelligence and Statistics. PMLR, pp. 207–215.

de Freitas Cunha, R.L., Chaimowicz, L., 2023. An smdp approach for reinforcement learning in hpc cluster schedulers. Future Gener. Comput. Syst. 139, 239–252. http://dx.doi.org/10.1016/j.future.2022.09.025, URL https://www.sciencedirect.com/science/article/pii/S0167739X22003090.

Ding, X., Chen, L., Emani, M., Liao, C., Lin, P.-H., Vanderbruggen, T., Xie, Z., Cerpa, A., Du, W., 2023. Hpc-gpt: Integrating large language model for high-performance computing. In: Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis. SC-W '23, Association for Computing Machinery, New York, NY, USA, pp. 951–960. http://dx.doi.org/10.1145/3624062.3624172.

Dong, W., Ren, J., 2023. Autoconstruct: Automated neural surrogate model building and deployment for hpc applications. In: Proceedings of the 13th Workshop on AI and Scientific Computing At Scale using Flexible Computing. FlexScience '23, Association for Computing Machinery, New York, NY, USA, pp. 33–40. http://dx.doi.org/10.1145/3589013.3596677.

Elsevier, 2018. Scopus - the largest database of peer-reviewed literature. URL https://www.elsevier.com/solutions/scopus. (Accessed 25 November 2025).

Fan, Y., Lan, Z., 2021. Dras-cqsim: A reinforcement learning based framework for hpc cluster scheduling. Softw. Impacts 8, 100077. http://dx.doi.org/10.1016/j.simpa.2021.100077, URL https://www.sciencedirect.com/science/article/pii/S2665963821000257.

Fan, Y., Lan, Z., Childers, T., Rich, P., Allcock, W., Papka, M.E., 2021. Deep reinforcement agent for scheduling in hpc. In: 2021 IEEE International Parallel and Distributed Processing Symposium. IPDPS, pp. 807–816. http://dx.doi.org/10.1109/IPDPS49936.2021.00090.

Fan, Y., Lan, Z., Rich, P., Allcock, W., Papka, M.E., 2022a. Hybrid workload scheduling on hpc systems. In: 2022 IEEE International Parallel and Distributed Processing Symposium. IPDPS, pp. 470–480. http://dx.doi.org/10.1109/IPDPS53621.2022.00052.

Fan, Y., Li, B., Favorite, D., Singh, N., Childers, T., Rich, P., Allcock, W., Papka, M.E., Lan, Z., 2022b. Dras: Deep reinforcement learning for cluster scheduling in high performance computing. IEEE Trans. Parallel Distrib. Syst. 33 (12), 4903–4917. http://dx.doi.org/10.1109/TPDS.2022.3205325.

Fdida, S., Makris, N., Korakis, T., Bruno, R., Passarella, A., Andreou, P., Belter, B., Crettaz, C., Dabbous, W., Demchenko, Y., Knopp, R., 2022. Slices, a scientific instrument for the networking community. Comput. Commun. 193, 189–203. http://dx.doi.org/10.1016/j.comcom.2022.07.019, URL https://www.sciencedirect.com/science/article/pii/S0140366422002663.

Feitelson, D.G., Tsafrir, D., Krakov, D., 2014. Experience with using the parallel workloads archive. J. Parallel Distrib. Comput. 74 (10), 2967–2982. http://dx.doi.org/10.1016/j.jpdc.2014.06.013, URL https://www.sciencedirect.com/science/article/pii/S0743731514001154.

Fox, G., Glazier, J., Kadupitiya, J., Jadhao, V., Kim, M., Qiu, J., Sluka, J., Somogy, E., Marathe, M., Adiga, A., Chen, J., Beckstein, O., Jha, S., 2019. Learning everywhere: Pervasive machine learning for effective high-performance computation. In: Proceedings - 2019 IEEE 33rd International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2019, Proceedings - 2019 IEEE 33rd International Parallel and Distributed Processing Symposium Workshops. IPDPSW 2019, Institute of Electrical and Electronics Engineers Inc., pp. 422–429. http://dx.doi.org/10.1109/IPDPSW.2019.00081, publisher Copyright: © 2019 IEEE.; 33rd IEEE International Parallel and Distributed Processing Symposium Workshops, IPDPSW 2019 ; Conference date: 20-05-2019 Through 24-05-2019.

Gao, Y., Liu, Y., Zhang, H., Li, Z., Zhu, Y., Lin, H., Yang, M., 2020. Estimating gpu memory consumption of deep learning models. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2020, Association for Computing Machinery, New York, NY, USA, pp. 1342–1352. http://dx.doi.org/10.1145/3368089.3417050.

Ge, Y., Hua, W., Mei, K., Ji, J., Tan, J., Xu, S., Li, Z., Zhang, Y., 2023a. Openagi: when llm meets domain experts. In: Proceedings of the 37th International Conference on Neural Information Processing Systems. NIPS '23, Curran Associates Inc., Red Hook, NY, USA.

Ge, Y., Ren, Y., Hua, W., Xu, S., Tan, J., Zhang, Y., 2023b. Llm as os, agents as apps: Envisioning aios, agents and the aios-agent ecosystem. arXiv:2312.03815. URL https://arxiv.org/abs/2312.03815.

Gebreyesus, Y., Dalton, D., De Chiara, D., Chinnici, M., Chinnici, A., 2024. Ai for automating data center operations: Model explainability in the data centre context using shapley additive explanations (shap). Electronics 13 (9), http://dx.doi.org/10.3390/electronics13091628, URL https://www.mdpi.com/2079-9292/13/9/1628.

GitHub, 2025. Github copilot: Your ai pair programmer. https://copilot.github.com/. (Accessed 25 November 2025).

Godoy, W., Valero Lara, P., Teranishi, K., Balaprakash, P., Vetter, J., 2023. Evaluation of openai codex for hpc parallel programming models kernel generation. http://dx.doi.org/10.1145/3605731.3605886, URL https://www.osti.gov/biblio/2000371.

Gu, R., Chen, Y., Liu, S., Dai, H., Chen, G., Zhang, K., Che, Y., Huang, Y., 2022. Liquid: Intelligent resource estimation and network-efficient scheduling for deep learning jobs on distributed gpu clusters. IEEE Trans. Parallel Distrib. Syst. 33 (11), 2808–2820. http://dx.doi.org/10.1109/TPDS.2021.3138825.

Gu, D., Zhao, Y., Zhong, Y., Xiong, Y., Han, Z., Cheng, P., Yang, F., Huang, G., Jin, X., Liu, X., 2023. Elasticflow: An elastic serverless training platform for distributed deep learning. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS 2023, Vol. 2, Association for Computing Machinery, New York, NY, USA, pp. 266–280. http://dx.doi.org/10.1145/3575693.3575721.

Guan, H., Mokadam, L.K., Shen, X., Lim, S.-H., Patton, R., 2020. Fleet: Flexible efficient ensemble training for heterogeneous deep neural networks. In: Dhillon, I., Papailiopoulos, D., Sze, V. (Eds.), Proceedings of Machine Learning and Systems. Vol. 2, pp. 247–261, URL https://proceedings.mlsys.org/paper_files/paper/2020/file/462211f67c7d858f663355eff93b745e-Paper.pdf.

Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y., Luo, F., Xiong, Y., Liang, W., 2024. Deepseek-coder: When the large language model meets programming – the rise of code intelligence. URL https://arxiv.org/abs/2401.14196.

Hackett, W., Birch, L., Trawicki, S., Suri, N., Garraghan, P., 2025. Bypassing prompt injection and jailbreak detection in llm guardrails. arXiv:2504.11168. URL https://arxiv.org/abs/2504.11168.

Hafner, J., Kresse, G., 1997. The Vienna AB-Initio Simulation Program VASP: An Efficient and Versatile Tool for Studying the Structural, Dynamic, and Electronic Properties of Materials. Springer US, Boston, MA, pp. 69–82. http://dx.doi.org/10.1007/978-1-4615-5943-6_10.

Han, Y., Huang, G., Song, S., Yang, L., Wang, H., Wang, Y., 2022. Dynamic neural networks: A survey. IEEE Trans. Pattern Anal. Mach. Intell. 44 (11), 7436–7456. http://dx.doi.org/10.1109/TPAMI.2021.3117837, URL https://doi.ieeecomputersociety.org/10.1109/TPAMI.2021.3117837.

Han, S., Zhang, Q., Yao, Y., Jin, W., Xu, Z., He, C., 2024. Llm multi-agent systems: Challenges and open problems. arXiv:2402.03578. URL https://arxiv.org/abs/2402.03578.

Hat, R., 2024. Llm and llm system risks and safeguards. URL https://www.redhat.com/en/blog/llm-and-llm-system-risks-and-safeguards. (Accessed 25 November 2025).

Heldens, S., Hijma, P., Werkhoven, B.V., Maassen, J., Belloum, A.S.Z., Van Nieuwpoort, R.V., 2020. The landscape of exascale research: A data-driven literature analysis. ACM Comput. Surv. 53 (2), http://dx.doi.org/10.1145/3372390.

2025. High performance computing for energy innovation (hpc4ei). https://hpc4energyinnovation.llnl.gov/. (Accessed 25 November 2025).

Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput. 9, 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hoffmann, J., Bauer, P., Sandu, I., Wedi, N., Geenen, T., Thiemert, D., 2023. Destination earth – a digital twin in support of climate services. Clim. Serv. 30, 100394. http://dx.doi.org/10.1016/j.cliser.2023.100394, URL https://www.sciencedirect.com/science/article/pii/S2405880723000559.

Hossain, A., Abdurahman, A., Islam, M.A., Ahmed, K., 2025. Power-aware scheduling for multi-center hpc electricity cost optimization. arXiv preprint arXiv:2503.11011.

Howard, M., 2022. Terraform – automating infrastructure as a service. arXiv:2205.10676. URL https://arxiv.org/abs/2205.10676.

Hu, Q., Nori, H., Sun, P., Wen, Y., Zhang, T., 2022. Primo: Practical learning-augmented systems with interpretable models. In: 2022 USENIX Annual Technical Conference. USENIX ATC 22, USENIX Association, Carlsbad, CA, pp. 519–538, URL https://www.usenix.org/conference/atc22/presentation/hu.

Hu, H., Su, J., Zhao, J., Peng, Y., Zhu, Y., Lin, H., Wu, C., 2024. Cdmpp: A device-model agnostic framework for latency prediction of tensor programs. In: Proceedings of the Nineteenth European Conference on Computer Systems. EuroSys '24, Association for Computing Machinery, New York, NY, USA, pp. 1054–1074. http://dx.doi.org/10.1145/3627703.3629572.

Hu, Q., Sun, P., Yan, S., Wen, Y., Zhang, T., 2021. Characterization and prediction of deep learning workloads in large-scale gpu datacenters. In: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis. SC '21, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3458817.3476223.

Hu, Q., Ye, Z., Zhang, M., Chen, Q., Sun, P., Wen, Y., Zhang, T., 2023a. Hydro: Surrogate-based hyperparameter tuning service in datacenters. In: 17th USENIX Symposium on Operating Systems Design and Implementation. OSDI 23, USENIX Association, Boston, MA, pp. 757–777, URL https://www.usenix.org/conference/osdi23/presentation/hu.

Hu, Q., Zhang, M., Sun, P., Wen, Y., Zhang, T., 2023b. Lucid: A non-intrusive, scalable and interpretable scheduler for deep learning training jobs. In: Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS 2023, Vol. 2, Association for Computing Machinery, New York, NY, USA, pp. 457–472. http://dx.doi.org/10.1145/3575693.3575705.

Huang, J., Chen-Chuan Chang, K., 2023. Towards reasoning in large language models: A survey. In: Findings of the Association for Computational Linguistics, ACL 2023, Proceedings of the Annual Meeting of the Association for Computational Linguistics. Association for Computational Linguistics (ACL), pp. 1049–1065. http://dx.doi.org/10.18653/v1/2023.findings-acl.67, we would like to thank Jason Wei (OpenAI) and Denny Zhou (Google DeepMind) for their valuable advice and constructive feedback on this work. This material is based upon work supported by the National Science Foundation IIS 16-19302 and IIS 16-33755, Zhejiang University ZJU Research 083650, IBM-Illinois Center for Cognitive Computing SystemsResearch (C3SR) and IBM-Illinois Discovery Accelerator Institute (IIDAI), gift grants from eBay and Microsoft Azure, UIUC OVCR CCIL Planning Grant 434S34, UIUC CSBS Small Grant 434C8U, and UIUC New Frontiers Initiative. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies. We would like to thank Jason Wei (OpenAI) and Denny Zhou (Google DeepMind) for their valuable advice and constructive feedback on this work. This material is based upon work supported by the National Science Foundation IIS 16-19302 and IIS 16-33755, Zhejiang University ZJU Research 083650, IBM-Illinois Center for Cognitive Computing Systems Research (C3SR) and IBM-Illinois Discovery Accelerator Institute (IIDAI), gift grants from eBay and Microsoft Azure, UIUC OVCR CCIL Planning Grant 434S34, UIUC CSBS Small Grant 434C8U, and UIUC New Frontiers Initiative. Any opinions, findings, and conclusions or recommendations expressed in this publication are those of the author(s) and do not necessarily reflect the views of the funding agencies.; 61st Annual Meeting of the Association for Computational Linguistics, ACL 2023 ; Conference date: 09-07-2023 Through 14-07-2023.

Huang, S., Ontañón, S., 2022. A closer look at invalid action masking in policy gradient algorithms. In: Barták, R., Keshtkar, F., Franklin, M. (Eds.), Proceedings of the Thirty-Fifth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2022, Hutchinson Island, Jensen Beach, Florida, USA, May (2022) 15-18. http://dx.doi.org/10.32473/flairs.v35i.130584.

Kadirvel, S., Ho, J., Fortes, J.A.B., 2013. Fault management in Map-Reduce through early detection of anomalous nodes. In: 10th International Conference on Autonomic Computing. ICAC 13, USENIX Association, San Jose, CA, pp. 235–245, URL https://www.usenix.org/conference/icac13/technical-sessions/presentation/kadirvel.

Kadupitiya, J., Fox, G.C., Jadhao, V., 2019. Machine learning for performance enhancement of molecular dynamics simulations. In: Rodrigues, J.M.F., Cardoso, P.J.S., Monteiro, J., Lam, R., Krzhizhanovskaya, V.V., Lees, M.H., Dongarra, J.J., Sloot, P.M. (Eds.), Computational Science – ICCS 2019. Springer International Publishing, Cham, pp. 116–130.

Kaltenecker, C., Grebhahn, A., Siegmund, N., Apel, S., 2020. The interplay of sampling and machine learning for software performance prediction. IEEE Softw. 37 (4), 58–66. http://dx.doi.org/10.1109/MS.2020.2987024.

Kambhampati, S., Valmeekam, K., Guan, L., Verma, M., Stechly, K., Bhambri, S., Saldyt, L., Murthy, A., 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In: Proceedings of the 41st International Conference on Machine Learning. ICML'24, JMLR.org.

Kandpal, N., Pillutla, K., Oprea, A., Kairouz, P., Choquette-Choo, C.A., Xu, Z., 2024. User inference attacks on large language models. In: Al-Onaizan, Y., Bansal, M., Chen, Y.-N. (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing. Association for Computational Linguistics, Miami, Florida, USA, pp. 18238–18265, URL https://aclanthology.org/2024.emnlp-main.1014/.

Kocot, B., Czarnul, P., Proficz, J., 2023. Energy-aware scheduling for high-performance computing systems: A survey. Energies 16 (2), http://dx.doi.org/10.3390/en16020890, URL https://www.mdpi.com/1996-1073/16/2/890.

Kong, Y., Ruan, J., Chen, Y., Zhang, B., Bao, T., Shiwei, S., d. G. Qing, Hu, X., Mao, H., Li, Z., Zeng, X., Zhao, R., Wang, X., 2024. TPTU-v2: Boosting task planning and tool usage of large language model-based agents in real-world industry systems. In: Dernoncourt, F., Preoţiuc-Pietro, D., Shimorina, A. (Eds.), Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: Industry Track. Association for Computational Linguistics, Miami, Florida, US, pp. 371–385. http://dx.doi.org/10.18653/v1/2024.emnlp-industry.27, URL https://aclanthology.org/2024.emnlp-industry.27/.

Koslovski, G.P., Pereira, K., Albuquerque, P.R., 2024. Dag-based workflows scheduling using actor–critic deep reinforcement learning. Future Gener. Comput. Syst. 150, 354–363. http://dx.doi.org/10.1016/j.future.2023.09.018, URL https://www.sciencedirect.com/science/article/pii/S0167739X23003485.

Krizhevsky, A., Sutskever, I., Hinton, G.E., 2012. Imagenet classification with deep convolutional neural networks. In: Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1. NIPS'12, Curran Associates Inc., Red Hook, NY, USA, pp. 1097–1105.

Lee, J.-K., Hong, T., Lee, G., 2024. Ai-based approach to firewall rule refinement on high-performance computing service network. Appl. Sci. 14 (11), http://dx.doi.org/10.3390/app14114373, URL https://www.mdpi.com/2076-3417/14/11/4373.

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., W.-t. Yih, Rocktäschel, T., Riedel, S., Kiela, D., 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA.

Li, S., Liu, H., Bian, Z., Fang, J., Huang, H., Liu, Y., Wang, B., You, Y., 2023. Colossal-ai: A unified deep learning system for large-scale parallel training. In: Proceedings of the 52nd International Conference on Parallel Processing. ICPP '23, Association for Computing Machinery, New York, NY, USA, pp. 766–775. http://dx.doi.org/10.1145/3605573.3605613.

Li, J., Zhang, X., Han, L., Ji, Z., Dong, X., Hu, C., 2021. Okcm: improving parallel task scheduling in high-performance computing systems using online learning. J. Supercomput. 77 (6), 5960–5983. http://dx.doi.org/10.1007/s11227-020-03506-5.

Li, J., Zhang, X., Wei, J., Ji, Z., Wei, Z., 2022. Garlsched: Generative adversarial deep reinforcement learning task scheduling optimization for large-scale high performance computing systems. Future Gener. Comput. Syst. 135, 259–269. http://dx.doi.org/10.1016/j.future.2022.04.032, URL https://www.sciencedirect.com/science/article/pii/S0167739X22001613.

Liang, P., Song, B., Zhan, X., Chen, Z., Yuan, J., 2024. Automating the training and deployment of models in mlops by integrating systems with machine learning. Appl. Comput. Eng. 67, 1–7. http://dx.doi.org/10.54254/2755-2721/67/20240690.

Liu, Y., Li, S., Fang, J., Shao, Y., Yao, B., You, Y., 2023. Colossal-auto: Unified automation of parallelization and activation checkpoint for large-scale models. arXiv:2302.02599. URL https://arxiv.org/abs/2302.02599.

Lou, Y., Caruana, R., Gehrke, J., Hooker, G., 2013. Accurate intelligible models with pairwise interactions. In: Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. KDD '13, Association for Computing Machinery, New York, NY, USA, pp. 623–631. http://dx.doi.org/10.1145/2487575.2487579.

Ltaief, H., Hong, Y., Dabah, A., Alomairy, R., Abdulah, S., Goreczny, C., Gepner, P., Ravasi, M., Gratadour, D., Keyes, D., 2023. Steering customized ai architectures for hpc scientific applications. In: High Performance Computing: 38th International Conference, ISC High Performance 2023, Hamburg, Germany, May (2023) 21–25, Proceedings. Springer-Verlag, Berlin, Heidelberg, pp. 125–143. http://dx.doi.org/10.1007/978-3-031-32041-5_7.

Lublin, U., Feitelson, D.G., 2003. The workload on parallel supercomputers: modeling the characteristics of rigid jobs. J. Parallel Distrib. Comput. 63 (11), 1105–1122. http://dx.doi.org/10.1016/S0743-7315(03)00108-4, URL https://www.sciencedirect.com/science/article/pii/S0743731503001084.

Lubrano, F., Vercellino, C., Vitali, G., Viviani, P., Scionti, A., Terzo, O., 2024. Advanced resource allocation in the context of heterogeneous workflows management. pp. 14–20. http://dx.doi.org/10.1145/3642978.3652835, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85191989496&doi=10.1145%2f3642978.3652835&partnerID=40&md5=defe58544aea315498139c785aa1609e.

Lundberg, S.M., Lee, S.-I., 2017. A unified approach to interpreting model predictions. pp. 4765–4774, URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf.

Luszczek, P., Sid-Lakhdar, W.M., Dongarra, J., 2023. Combining multitask and transfer learning with deep gaussian processes for autotuning-based performance engineering. Int. J. High Perform. Comput. Appl. 37 (3–4), 229–244. http://dx.doi.org/10.1177/10943420231166365, arXiv:https://doi.org/10.1177/10943420231166365.

van der Maaten, L., Hinton, G., 2008. Visualizing data using t-sne. J. Mach. Learn. Res. 9 (86), 2579–2605, URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Mahajan, K., Chu, C.-H., Sridharan, S., Akella, A., 2023. Better together: Jointly optimizing ML collective scheduling and execution planning using SYNDICATE. In: 20th USENIX Symposium on Networked Systems Design and Implementation. NSDI 23, USENIX Association, Boston, MA, pp. 809–824, URL https://www.usenix.org/conference/nsdi23/presentation/mahajan.

Malms, M., Cargemel, L., Suarez, E., Mittenzwey, N., Duranton, M., Sezer, S., Prunty, C., rosse laurent, P., Pérez-Harnandez, M., Marazakis, M., Lonsdale, G., Carpenter, P., Antoniu, G., Narasimharmurthy, S., Brinkman, A., Pleiter, D., Haus, U.-U., Krueger, J., Hoppe, H.-C., Haas, R., 2022. Etp4hpc's sra 5 - strategic research agenda for high-performance computing in europe - 2022.

Mao, H., Alizadeh, M., Menache, I., Kandula, S., 2016. Resource management with deep reinforcement learning. In: Proceedings of the 15th ACM Workshop on Hot Topics in Networks. HotNets '16, Association for Computing Machinery, New York, NY, USA, pp. 50–56. http://dx.doi.org/10.1145/3005745.3005750.

Mei, K., Zhu, X., Xu, W., Hua, W., Jin, M., Li, Z., Xu, S., Ye, R., Ge, Y., Zhang, Y., 2024. Aios: Llm agent operating system. arXiv:2403.16971. URL https://arxiv.org/abs/2403.16971.

Menear, K., Scully-Allison, C., Duplyakin, D., 2024. Quantifying uncertainty in hpc job queue time predictions. In: Practice and Experience in Advanced Research Computing 2024: Human Powered Computing. PEARC '24, Association for Computing Machinery, New York, NY, USA, http://dx.doi.org/10.1145/3626203.3670627.

Menon, H., Bhatele, A., Gamblin, T., 2020. Auto-tuning parameter choices in hpc applications using bayesian optimization. In: 2020 IEEE International Parallel and Distributed Processing Symposium. IPDPS, pp. 831–840. http://dx.doi.org/10.1109/IPDPS47924.2020.00090.

Merzky, A., Titov, M., Turilli, M., Kilic, O., Wang, T., Jha, S., 2025. Scalable runtime architecture for data-driven, hybrid hpc and ml workflow applications. arXiv:2503.13343. URL https://arxiv.org/abs/2503.13343.

Messina, P., 2017. The exascale computing project. Comput. Sci. Eng. 19 (3), 63–67. http://dx.doi.org/10.1109/MCSE.2017.57.

Miao, X., Oliaro, G., Zhang, Z., Cheng, X., Wang, Z., Zhang, Z., Wong, R.Y.Y., Zhu, A., Yang, L., Shi, X., Shi, C., Chen, Z., Arfeen, D., Abhyankar, R., Jia, Z., 2024. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS '24, Vol. 3, Association for Computing Machinery, New York, NY, USA, pp. 932–949. http://dx.doi.org/10.1145/3620666.3651335.

Mohammed, B., Awan, I., Ugail, H., Younas, M., 2019. Failure prediction using machine learning in a virtualised hpc system and application. Clust. Comput. Vol. 22 (2), 471–485. http://dx.doi.org/10.1007/s10586-019-02917-1.

Molan, M., Ahmed Khan, J., Borghesi, A., Bartolini, A., 2023a. Graph neural networks for anomaly anticipation in hpc systems. In: Companion of the 2023 ACM/SPEC International Conference on Performance Engineering. ICPE '23 Companion, Association for Computing Machinery, New York, NY, USA, pp. 239–244. http://dx.doi.org/10.1145/3578245.3585335.

Molan, M., Ardebili, M.S., Khan, J.A., Beneventi, F., Cesarini, D., Borghesi, A., Bartolini, A., 2024. Graafe: Graph anomaly anticipation framework for exascale hpc systems. Future Gener. Comput. Syst. 160, 644–653. http://dx.doi.org/10.1016/j.future.2024.06.032, URL https://www.sciencedirect.com/science/article/pii/S0167739X24003327.

Molan, M., Borghesi, A., Cesarini, D., Benini, L., Bartolini, A., 2023b. Ruad: Unsupervised anomaly detection in hpc systems. Future Gener. Comput. Syst. 141 (C), 542–554. http://dx.doi.org/10.1016/j.future.2022.12.001.

Munley, C., Jarmusch, A., Chandrasekaran, S., 2024. Llm4vv: Developing llm-driven testsuite for compiler validation. Future Gener. Comput. Syst. 160, 1–13. http://dx.doi.org/10.1016/j.future.2024.05.034, URL https://www.sciencedirect.com/science/article/pii/S0167739X24002449.

Mushtaq, S.U., Sheikh, S., Idrees, S.M., Malla, P.A., 2024. In-depth analysis of fault tolerant approaches integrated with load balancing and task scheduling. Peer-To-Peer Netw. Appl. 17 (6), 4303–4337. http://dx.doi.org/10.1007/s12083-024-01798-5.

Nan, Z., Dave, M., Shen, X., Liao, C., Vanderbruggen, T., Lin, P.-H., Emani, M., 2022. Interactive nlu-powered ontology-based workflow synthesis for fair support of hpc. In: 2022 IEEE/ACM International Workshop on HPC User Support Tools. HUST, pp. 29–40. http://dx.doi.org/10.1109/HUST56722.2022.00009.

Nan, Z., Guan, H., Shen, X., 2020. Hisyn: human learning-inspired natural language programming. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. ESEC/FSE 2020, Association for Computing Machinery, New York, NY, USA, pp. 75–86. http://dx.doi.org/10.1145/3368089.3409673.

NAS, 1994. Nas parallel benchmarks. URL https://www.nas.nasa.gov/publications/npb.html. (Accessed 25 November 2025).

NERSC, 2025. Nersc systems. URL https://docs.nersc.gov/systems/. (Accessed 25 November 2025).

Nowak, M., Frankowski, G., Meyer, N., Yilmaz, E., Erdogan, O., Nominé, J.-P., Robin, F., 2024. Security in hpc centres. https://prace-ri.eu/wp-content/uploads/wp79.pdf, partnership for Advanced Computing in Europe (PRACE), URL https://prace-ri.eu/wp-content/uploads/wp79.pdf.

Oh, H., Kim, K., Kim, J., Kim, S., Lee, J., Chang, D.-s., Seo, J., 2024. Exegpt: Constraint-aware resource scheduling for llm inference. In: Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS '24, Vol. 2, Association for Computing Machinery, New York, NY, USA, pp. 369–384. http://dx.doi.org/10.1145/3620665.3640383.

OpenAI, 2024. Openai codex. URL https://openai.com/blog/openai-codex/. (Accessed 25 November 2025).

Packer, C., Wooders, S., Lin, K., Fang, V., Patil, S.G., Stoica, I., Gonzalez, J.E., 2024. Memgpt: Towards llms as operating systems. arXiv:2310.08560. URL https://arxiv.org/abs/2310.08560.

Palla, A., 2023. Chatbot instruction prompts. https://huggingface.co/datasets/alespalla/chatbot_instruction_prompts.

Pearce, O., Burmark, J., Hornung, R., Bogale, B., Lumsden, I., McKinsey, M., Yokelson, D., Boehme, D., Brink, S., Taufer, M., Scogland, T., 2024. Raja performance suite : Performance portability analysis with caliper and thicket. In: SC24-W: Workshops of the International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1206–1218. http://dx.doi.org/10.1109/SCW63240.2024.00162.

Peixoto, E., Torres, D., Carneiro, D., Silva, B., Marques, R., 2025. Reusing ml models in dynamic data environments: Data similarity-based approach for efficient mlops. Big Data Cogn. Comput. 9 (2), http://dx.doi.org/10.3390/bdcc9020047, URL https://www.mdpi.com/2504-2289/9/2/47.

Peterson, J.L., Bay, B., Koning, J., Robinson, P., Semler, J., White, J., Anirudh, R., Athey, K., Bremer, P.-T., Di Natale, F., Fox, D., Gaffney, J.A., Jacobs, S.A., Kailkhura, B., Kustowski, B., Langer, S., Spears, B., Thiagarajan, J., Van Essen, B., Yeom, J.-S., 2022. Enabling machine learning-ready hpc ensembles with merlin. Future Gener. Comput. Syst. 131, 255–268. http://dx.doi.org/10.1016/j.future.2022.01.024, URL https://www.sciencedirect.com/science/article/pii/S0167739X22000322.

Pham, T.-T., Pister, M., Couvée, P., 2019. Recurrent neural network for classifying of hpc applications. In: 2019 Spring Simulation Conference (SpringSim). pp. 1–12. http://dx.doi.org/10.23919/SpringSim.2019.8732923.

Pochelu, P., Petiton, S.G., Conche, B., 2022. A deep neural networks ensemble workflow from hyperparameter search to inference leveraging gpu clusters. In: International Conference on High Performance Computing in Asia-Pacific Region. HPCAsia '22, Association for Computing Machinery, New York, NY, USA, pp. 61–71. http://dx.doi.org/10.1145/3492805.3492819.

Radaideh, M.I., Kozlowski, T., 2020. Surrogate modeling of advanced computer simulations using deep gaussian processes. Reliab. Eng. Syst. Saf. 195, 106731. http://dx.doi.org/10.1016/j.ress.2019.106731, URL https://www.sciencedirect.com/science/article/pii/S0951832019301711.

Rahman, A., Cvetkovic, V., Reece, K., Walters, A., Hassan, Y., Tummeti, A., Torres, B., Cooney, D., Ellis, M., Nikolopoulos, D.S., 2025. Marco: Multi-agent code optimization with real-time knowledge integration for high-performance computing. arXiv:2505.03906. URL https://arxiv.org/abs/2505.03906.

Rashad, M., 2023. Chatgpt prompts. https://huggingface.co/datasets/MohamedRashad/ChatGPT-prompts.

Rasley, J., Rajbhandari, S., Ruwase, O., He, Y., 2020. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. KDD '20, Association for Computing Machinery, New York, NY, USA, pp. 3505–3506. http://dx.doi.org/10.1145/3394486.3406703.

Reed, J., DeVito, Z., He, H., Ussery, A., Ansel, J., 2022. Torch.fx: Practical program capture and transformation for deep learning in python. Proc. Mach. Learn. Syst. 4, 638–651.

Ren, J., Xu, D., Yang, S., Zhao, J., Li, Z., Navasca, C., Wang, C., Xu, H., Li, D., 2024. Enabling large dynamic neural network training with learning-based memory management. In: 2024 IEEE International Symposium on High-Performance Computer Architecture. HPCA, pp. 788–802. http://dx.doi.org/10.1109/HPCA57654.2024.00066, URL https://ieeexplore-ieee-org.proxy.bnl.lu/stamp/stamp.jsp?tp=&arnumber=10476398.

Rousseeuw, P.J., 1987. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. J. Comput. Appl. Math. 20, 53–65. http://dx.doi.org/10.1016/0377-0427(87)90125-7, URL https://www.sciencedirect.com/science/article/pii/0377042787901257.

Sarma, R., Inanc, E., Aach, M., Lintermann, A., 2024. Parallel and scalable AI in HPC systems for CFD applications and beyond. http://dx.doi.org/10.3389/fhpcp.2024.1444337, missing Journal: Frontiers in High Performance Computing (Front. High Perform. Comput.) = 2813-7337 (import from CrossRef, Journals: juser.fzjuelich.de); Please add the journal to the list of journals. URL https://juser.fz-juelich.de/record/1031520.

Saxena, R., Baskar, A., Haroon, S., Hayat, S., Shcherbakov, O., Kayabay, K., Hoppe, D., 2024. Cybersecurity concerns of artificial intelligence applications on high-performance computing systems. In: AISyS 2024. Venice, Italy.

SchedMD, 2025. Slurm publications. URL https://slurm.schedmd.com/publications.html. (Accessed 25 November 2025).

Schroeder, B., Gibson, G., 2006. The computer failure data repository (cfdr): collecting, sharing and analyzing failure data. In: Proceedings of the 2006 ACM/IEEE Conference on Supercomputing. SC '06, Association for Computing Machinery, New York, NY, USA, pp. 154–es. http://dx.doi.org/10.1145/1188455.1188615.

Sencan, E., Lee, Y.-C., Casey, C., Schwaller, B., Leung, V.J., Brandt, J., Kulis, B., Egele, M., Coskun, A.K., 2025. Refine: A robust approach to unsupervised anomaly detection for production hpc systems. In: ISC High Performance 2025 Research Paper Proceedings (40th International Conference). pp. 1–12, URL https://ieeexplore.ieee.org/document/11018307.

Shu, D., Li, Z., Barati Farimani, A., 2023. A physics-informed diffusion model for high-fidelity flow field reconstruction. J. Comput. Phys. 478, 111972. http://dx.doi.org/10.1016/j.jcp.2023.111972, URL https://www.sciencedirect.com/science/article/pii/S0021999123000670.

Skvortsov, N.A., Stupnikov, S.A., 2022. A semantic approach to workflow management and reuse for research problem solving. Data Intell. 4 (2), 439–454. http://dx.doi.org/10.1162/dint_a_00142, apply FAIR principles to workflows, meaning that workflows should be findable (F), accessible (A), interoperable (I), and reusable (R). arXiv:https://direct.mit.edu/dint/article-pdf/4/2/439/2012427/dint_a_00142.pdf.

Sohl-Dickstein, J., Weiss, E.A., Maheswaranathan, N., Ganguli, S., 2015. Deep unsupervised learning using nonequilibrium thermodynamics. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37. ICML'15, JMLR.org, pp. 2256–2265.

Syrigos, I., Kefalas, D., Makris, N., Korakis, T., 2023. Eelas: Energy efficient and latency aware scheduling of cloud-native ml workloads. In: 2023 15th International Conference on COMmunication Systems & NETworkS. COMSNETS, pp. 819–824. http://dx.doi.org/10.1109/COMSNETS56262.2023.10041344.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., Hashimoto, T.B., 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Teranishi, K., Menon, H., Godoy, W.F., Balaprakash, P., Bau, D., Ben-Nun, T., Bhatele, A., Franchetti, F., Franusich, M., Gamblin, T., Georgakoudis, G., Goldstein, T., Guha, A., Hahn, S., Iancu, C., Jin, Z., Jones, T., Low, T.M., Mankad, H., Miniskar, N.R., Monil, M.A.H., Nichols, D., Parasyris, K., Pophale, S., Valero-Lara, P., Vetter, J.S., Williams, S., Young, A., 2025. Leveraging ai for productive and trustworthy hpc software: Challenges and research directions. arXiv:2505.08135. URL https://arxiv.org/abs/2505.08135.

Thanapol, P., Lavangnananda, K., Leprévost, F., Schleich, J., Bouvry, P., 2023. Scheduling deep learning training in gpu cluster using the model-similarity-based policy. In: Nguyen, N.T., Boonsang, S., Fujita, H., Hnatkowska, B., Hong, T.-P., Pasupa, K., Selamat, A. (Eds.), Intelligent Information and Database Systems. Springer Nature, Singapore, Singapore, pp. 363–374.

TOP500 Project, 2025. Top500 list. https://top500.org. (Accessed 25 November 2025).

Turing, A.M., 1950. Computing machinery and intelligence. Mind 59 (236), 433–460. URL http://www.jstor.org/stable/2251299.

Turner, A., 2019. Single node performance comparison report. http://dx.doi.org/10.5281/zenodo.2616549.

Vacchetti, B., Cerquitelli, T., Nosenzo, V., Capitelli, E., Chiosso, L., Trocano, M., 2024. Jem: An ai-based engine workflow to predict simulation's execution time on hpc cluster. In: 2024 International Conference on Control, Automation and Diagnosis. ICCAD, pp. 1–5. http://dx.doi.org/10.1109/ICCAD60883.2024.10553971.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I., 2017. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems. NIPS'17, Curran Associates Inc., Red Hook, NY, USA, pp. 6000–6010.

Vercellino, C., Scionti, A., Varavallo, G., Viviani, P., Vitali, G., Terzo, O., 2023. A machine learning approach for an hpc use case: the jobs queuing time prediction. Future Gener. Comput. Syst. 143, 215–230. http://dx.doi.org/10.1016/j.future.2023.01.020, URL https://www.sciencedirect.com/science/article/pii/S0167739X23000274.

Verma, G., Emani, M., Liao, C., Lin, P.-H., Vanderbruggen, T., Shen, X., Chapman, B., 2021. Hpcfair: Enabling fair ai for hpc applications. In: 2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments. MLHPC, pp. 58–68. http://dx.doi.org/10.1109/MLHPC54614.2021.00011.

VI-HPS, 2022. Scalable performance measurement infrastructure for parallel codes. URL https://www.vi-hps.org/projects/score-p. (Accessed 25 November 2025).

Wang, L., Rodriguez, M.A., Lipovetzky, N., 2025. Optimizing HPC] scheduling: a hierarchical reinforcement learning approach for intelligent job selection and allocation. J. Supercomput. 81 (8), 918. http://dx.doi.org/10.1007/s11227-025-07396-3.

Wang, Y., Xie, Z., Xu, K., Dou, Y., Lei, Y., 2016. An efficient and effective convolutional auto-encoder extreme learning machine network for 3d feature learning. Neurocomputing 174, 988–998. http://dx.doi.org/10.1016/j.neucom.2015.10.035, URL https://www.sciencedirect.com/science/article/pii/S0925231215014940.

Wang, Q., Zhang, H., Qu, C., Shen, Y., Liu, X., Li, J., 2021. Rlschert: An hpc job scheduler using deep reinforcement learning and remaining time prediction. Appl. Sci. 11 (20), http://dx.doi.org/10.3390/app11209448, URL https://www.mdpi.com/2076-3417/11/20/9448.

Ward, L., Sivaraman, G., Pauloski, J.G., Babuji, Y., Chard, R., Dandu, N., Redfern, P.C., Assary, R.S., Chard, K., Curtiss, L.A., Thakur, R., Foster, I., 2021. Colmena: Scalable machine-learning-based steering of ensemble simulations for high performance computing. In: 2021 IEEE/ACM Workshop on Machine Learning in High Performance Computing Environments. MLHPC, pp. 9–20. http://dx.doi.org/10.1109/MLHPC54614.2021.00007.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E.H., Le, Q.V., Zhou, D., 2022. Chain-of-thought prompting elicits reasoning in large language models. In: Proceedings of the 36th International Conference on Neural Information Processing Systems. NIPS '22, Curran Associates Inc., Red Hook, NY, USA.

Weihua Liu, B.S., Hu, Erh-Wen, Wang, J., 2021. Using machine learning techniques for dsp software performance prediction at source code level. Connect. Sci. 33 (1), 26–41. http://dx.doi.org/10.1080/09540091.2020.1762542, arXiv:https://doi.org/10.1080/09540091.2020.1762542.

Welbl, J., Glaese, A., Uesato, J., Dathathri, S., Mellor, J., Hendricks, L.A., Anderson, K., Kohli, P., Coppin, B., Huang, P.-S., 2021. Challenges in detoxifying language models. In: Moens, M.-F., Huang, X., Specia, L., t. Yih, S.W. (Eds.), Findings of the Association for Computational Linguistics: EMNLP 2021. Association for Computational Linguistics, Punta Cana, Dominican Republic, pp. 2447–2469. http://dx.doi.org/10.18653/v1/2021.findings-emnlp.210, URL https://aclanthology.org/2021.findings-emnlp.210/.

Willemsen, F.-J., Schoonhoven, R., Filipovič, J., Tørring, J.O., van Nieuwpoort, R., van Werkhoven, B., 2024. A methodology for comparing optimization algorithms for auto-tuning. Future Gener. Comput. Syst. 159, 489–504. http://dx.doi.org/10.1016/j.future.2024.05.021, URL https://www.sciencedirect.com/science/article/pii/S0167739X24002498.

Wu, S., Guan, J., 2024. Prediction of disk failure based on classification intensity resampling. Information 15 (6), 322, copyright - © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/). Notwithstanding the ProQuest Terms and Conditions, you may use this content in accordance with the terms of the License; Last updated - 2024-06-27. URL https://www.proquest.com/scholarly-journals/prediction-disk-failure-based-on-classification/docview/3072350799/se-2.

Wyzykowski, A.B.V., Sousa, G.M.C.D., Coelho, B.S., Santos, L.B., Anschau, D., 2024. Optimizing geophysical workloads in high-performance computing: Leveraging machine learning and transformer models for enhanced parallelism and processor allocation. http://dx.doi.org/10.1109/DCHPC60845.2024.10454084, URL https://www.scopus.com/inward/record.uri?eid=2-s2.0-85187778131&doi=10.1109%2fDCHPC60845.2024.10454084&partnerID=40&md5=e297a54e70ad8df38fc9ef40f2adc7ad.

Xia, S., Sun, Y., Pan, X., Yuan, Y., Zhang, S., Hu, S., Tao, L., Li, Y., Feng, J., 2025. Effective node-level anomaly detection in hpc systems via coarse-grained clustering and fine-grained model sharing. In: The International Conference for High Performance Computing, Networking, Storage and Analysis. SC '25, St. Louis, MO, USA, open-sourced code and anomaly labeling tool released;clustering tool available :contentReferenceindex=1. URL https://nkcs.iops.ai/wp-content/uploads/2025/07/NodeSentry.pdf.

Xie, Y., Huang, S., Chen, T., Wei, F., 2023. Moec: mixture of expert clusters. In: Proceedings of the Thirty-Seventh AAAI Conference on Artificial Intelligence and Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence and Thirteenth Symposium on Educational Advances in Artificial Intelligence. AAAI'23/IAAI'23/EAAI'23, AAAI Press, http://dx.doi.org/10.1609/aaai.v37i11.26617.

Xue, H.-J., Dai, X.-Y., Zhang, J., Huang, S., Chen, J., 2017. Deep matrix factorization models for recommender systems. In: Proceedings of the 26th International Joint Conference on Artificial Intelligence. IJCAI'17, AAAI Press, pp. 3203–3209.

Yang, G., Hu, E., Babuschkin, I., Sidor, S., Liu, X., Farhi, D., Ryder, N., Pachocki, J., Chen, W., Gao, J., 2021. Tuning large neural networks via zero-shot hyper-parameter transfer. In: Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W. (Eds.), Advances in Neural Information Processing Systems. Vol. 34, Curran Associates, Inc., pp. 17084–17097, URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8df7c2e3c3c3be098ef7b382bd2c37ba-Paper.pdf.

Yang, Y., Shen, H., 2022. Deep reinforcement learning enhanced greedy optimization for online scheduling of batched tasks in cloud hpc systems. IEEE Trans. Parallel Distrib. Syst. 33 (11), 3003–3014. http://dx.doi.org/10.1109/TPDS.2021.3138459.

Yang, Q., Yang, T., Xiang, M., Zhang, L., Wang, H., Serafini, M., Guan, H., 2024. Gmorph: Accelerating multi-dnn inference via model fusion. In: Proceedings of the Nineteenth European Conference on Computer Systems. EuroSys '24, Association for Computing Machinery, New York, NY, USA, pp. 505–523. http://dx.doi.org/10.1145/3627703.3650074.

Yang, X., Zhou, Z., Wallace, S., Lan, Z., Tang, W., Coghlan, S., Papka, M.E., 2013. Integrating dynamic pricing of electricity into energy aware scheduling for hpc systems. In: Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis. pp. 1–11.

Zhang, D., Dai, D., He, Y., Bao, F.S., Xie, B., 2020. Rlscheduler: An automated hpc batch job scheduler using reinforcement learning. In: SC20: International Conference for High Performance Computing, Networking, Storage and Analysis. pp. 1–15. http://dx.doi.org/10.1109/SC41405.2020.00035.

Zhang, S., Li, D., Zhong, Z., Zhu, J., Liang, M., Luo, J., Sun, Y., Su, Y., Xia, S., Hu, Z., Zhang, Y., Pei, D., Sun, J., Liu, Y., 2022. Robust system instance clustering for large-scale web services. In: Proceedings of the ACM Web Conference 2022. WWW '22, Association for Computing Machinery, New York, NY, USA, pp. 1785–1796. http://dx.doi.org/10.1145/3485447.3511983.

Zhang, Y., Zhao, X., Yin, J., Zhang, L., Chen, Z., 2024. Operating system and artificial intelligence: A systematic review. arXiv preprint arXiv:2407.14567.

Zheng, L., Li, Z., Zhang, H., Zhuang, Y., Chen, Z., Huang, Y., Wang, Y., Xu, Y., Zhuo, D., Xing, E.P., Gonzalez, J.E., Stoica, I., 2022. Alpa: Automating inter- and Intra-Operator parallelism for distributed deep learning. In: 16th USENIX Symposium on Operating Systems Design and Implementation. OSDI 22, USENIX Association, Carlsbad, CA, pp. 559–578, URL https://www.usenix.org/conference/osdi22/presentation/zheng-lianmin.

Zheng, S., Liang, Y., Wang, S., Chen, R., Sheng, K., 2020. Flextensor: An automatic schedule exploration and optimization framework for tensor computation on heterogeneous system. In: Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems. ASPLOS '20, Association for Computing Machinery, New York, NY, USA, pp. 859–873. http://dx.doi.org/10.1145/3373376.3378508.

Zheng, L., Liu, R., Shao, J., Chen, T., Gonzalez, J., Stoica, I., Haj-Ali, A., 2021. Tenset: A large-scale program performance dataset for learned tensor compilers. In: Vanschoren, J., Yeung, S. (Eds.), Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks. Vol. 1, URL https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/a684eceee76fc522773286a895bc8436-Paper-round1.pdf.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., Zhang, W., 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 35, pp. 11106–11115.