



Agentic AI for High-Performance Computing and Large Scale Science: A Narrative Review of Autonomous Orchestration, Coordination Challenges, and Evaluation Gaps

Inam Ullah^{1*}, Abdul Haseeb Malik¹, Najeebullah¹, Muhammad Nouman¹, Qazi Ejaz Ali¹, Waheed Ur Rehman¹

¹Department of Computer Science, University of Peshawar, Peshawar, Pakistan

ARTICLE INFO

Article History:

Received: July 17, 2026
 Revised: August 11, 2026
 Accepted: September 18, 2026
 Available Online: September 30, 2026

Keywords:

agentic AI
 large language models
 high-performance computing
 multi-agent systems
 scientific workflows
 parallel scheduling
 autonomous laboratories
 reproducibility

Classification Codes:

Funding:

This research received no specific grant from any funding agency in the public or not-for-profit sector.

ABSTRACT

Agentic artificial intelligence (AI), in which systems built on large language models (LLMs) plan, call tools and act in multi-step loops, is increasingly proposed as a control layer for high-performance computing (HPC) and large-scale scientific workflows. This narrative review asks which architectural patterns connect agentic reasoning to HPC and laboratory execution, which coordination and scheduling problems arise when agents act concurrently on shared resources, and how rigorously these systems are evaluated. We cite 104 sources, classify each of them in a supplementary inventory, and compare the 8 agentic systems that meet documented inclusion criteria. We propose a taxonomy of four agent roles (resource-request, workflow-steering, data-management and provenance, and experiment agents), keep it separate from infrastructure that schedules agent workloads, and place system families on an autonomy spectrum scored with an explicit rubric. We then synthesize six cross-cutting challenges: hallucination propagation, scheduling visibility, energy accounting, reproducibility, trust and authorization, and interoperability. Our central finding is methodological: no two tabulated systems share an evaluation protocol, and only 3 of the 8 compare against a baseline. We close with a concrete benchmark design and a research agenda.



© 2026 The authors published by JCIS. This is an Open Access Article under the Creative Common Attribution Non-Commercial 4.0

Corresponding Author's Email: inamkhattak1999@uop.edu.pk

1. Introduction

Large language models (LLMs) have moved from producing text to taking action. When an LLM is embedded in a loop that lets it plan, call tools, observe results, and revise its plan, the resulting system is commonly termed an agent [1,2]. Some foundational agents like ReAct [1], Toolformer [2], Reflexion [3] and Voyager [4] established that it is possible to embed both reasoning and acting in an LLM, and multi-agent frameworks like AutoGen [5] and MetaGPT [6] and Generative Agents [7] proved that many agents (with roles and memories) could collaborate on tasks that exceed the context window of a single agent.

This has come to high-performance computing (HPC) and computational science. An agent may now be tasked to perform a parameter sweep, track convergence, and determine what to run next [8,9,10] instead of relying on a human to write a batch script to submit to Slurm or the Portable Batch System (PBS), and then to manually check results. This

is now starting to be seen by the eScience and HPC communities as a new control plane that covers instruments, autonomous labs, simulations, data centers, cloud systems and leadership-class HPC [11]. Agent-driven materials screening on Aurora exascale system [10]; LLM agents dispatching Parsl tasks on Polaris [8]; hybrid AI-HPC workflow engines – combining simulation and inference at scale [12,13,14,15].

Concurrently, this new control plane brings with it a host of problems, which parallel and distributed computing has been exploring for decades: how to efficiently schedule interdependent tasks, how much communication and coordination does a distributed system need before it fails to pay off, how to guarantee the reproducibility of a computation, how to capture the energy cost of a decision-making process. What is new is that the entity making scheduling and resource decisions is a non-deterministic, occasionally hallucinating language model rather than a deterministic program [16], and LLMs on their own remain unreliable planners unless their plans are checked by an external solver [17].

This review focuses specifically on the intersection of agentic AI with HPC-scale and scientific orchestration (the area sometimes called agentic AI for large-scale science [11]) rather than agentic AI broadly, which has already been surveyed at a general level [18,19,20,21]. Our contribution is threefold. First, we propose a taxonomy of the roles agents play in HPC and scientific pipelines. Second, we synthesize six cross-cutting technical challenges that recur across an otherwise disparate literature, and we state which of the links between them are supported by evidence and which are hypotheses. Third, and most importantly, we show that evaluation methodology, not architecture, is the field's principal bottleneck, and we use this gap to motivate a concrete research agenda. We also document the corpus, the inclusion criteria and the publication status of every source, so that the evidence behind each claim can be checked (Section 2.3 and Supplementary Table S1).

2. Scope and Review Methodology

We define agentic AI for HPC and science as any system in which an LLM-driven agent autonomously makes decisions that affect the execution of computational or experimental work on shared, resource-constrained infrastructure, including HPC clusters, graphics processing unit (GPU) pools, cloud batch systems, and physically instrumented autonomous laboratories. We exclude agent applications that are purely conversational or that operate only on a single user's local machine without resource contention, since these do not raise the parallel-computing questions that are our focus.

Literature was identified through arXiv listings in Distributed, Parallel, and Cluster Computing (cs.DC), Multiagent Systems (cs.MA) and Artificial Intelligence (cs.AI), for example [8,10,12,22]; through the call for papers of the 2026 AGENT4SC workshop on Agentic AI for Large-scale Science [11] and the proceedings of SC workshops [15,23]; through journals covering autonomous laboratories in chemistry and materials science [24,25,26,27,28,29,30]; and through citation-chain snowballing forward and backward from these seed papers. We additionally consulted recent general surveys of multi-agent LLM systems to cross-check terminology and identify adjacent work [16,18,19,20,21,31,32,33,34]. Searches were run iteratively between June and July 2026 using term combinations such as "agentic AI" AND ("HPC" OR "high-performance computing" OR "scheduling" OR "self-driving laboratory" OR "scientific workflow"), with no fixed start date given how recent this sub-area is, and a practical cutoff of late July 2026. The primary query is the one given above. Because the further term combinations used during snowballing were not logged at the time, we report only this primary string and treat the missing search log as a limitation (Section 8.1). Web resources cited in the review were last checked in October 2026, during revision.

2.1 Relationship to Existing Surveys

General surveys of agentic AI already exist and should be distinguished from the present review. Systematic literature reviews of agentic AI across the software development life cycle [20] and broad systematic surveys covering multi-agent architectures, cognitive foundations, interaction, explainability, security, and performance evaluation [21] address agentic AI as a general-purpose paradigm, with software engineering, business process automation, and general reasoning as their primary application domains. Surveys of multi-agent collaboration mechanisms [31,32,33] similarly focus on communication protocols, failure attribution, and self-evolution in a domain-agnostic setting. None of these treats HPC-scale resource scheduling, energy accounting, or scientific reproducibility as a central concern, and none draws systematically on the cs.DC literature. Conversely, domain-specific surveys of agentic AI for scientific discovery [35] and of AI scientists more broadly [36] cover autonomous research and self-driving laboratories in depth

but give comparatively little attention to the underlying parallel and distributed systems substrate (schedulers, task graphs, cluster resource contention) on which these agents ultimately execute. General multi-agent collaboration surveys such as [18] likewise focus on coordination theory and applications rather than on the resource-constrained execution settings that are our concern here. The present review is positioned deliberately at the intersection of these two literatures, which is also why its corpus is smaller: it excludes work that is agentic-AI-general without an HPC or large-scale-science angle, and it excludes classical HPC workflow-management literature that predates the introduction of LLM-based agents.

2.2 Why a Narrative Review, Not a PRISMA-Style Systematic Review

This work is a deliberately written "narrative review," and we feel we need to emphasize why. A systematic review in the PRISMA (Preferred Reporting Items for Systematic Reviews and Meta-Analyses) sense [37] has to have a pre-registered protocol, a search of multiple bibliographic databases (not only arXiv, but also Scopus and Web of Science, IEEE Xplore and the ACM Digital Library), dual-reviewer screening with explicit inclusion/exclusion criteria, and reported flow of records from identification to screening to final inclusion. It was not a protocol-driven search, but rather an iterative, snowball search; and, of course, it was heavily dependent on arXiv and workshop venues because these are an area that is too new for much of the output to be found in Scopus or Web of Science yet. Presenting this work under a "systematic review" label without that underlying process would overstate its methodological rigor. We instead adopt the narrative/scoping review framing used elsewhere in fast-moving computer science sub-fields, and we treat a full PRISMA-style systematic review of this literature, with multi-database screening and a reported inclusion flow, as a valuable and currently missing piece of future work in its own right (Section 8).

2.3 Corpus Size, Inclusion Criteria and Construction of Table 1

To make the corpus checkable, every source cited in this review is listed in Supplementary Table S1 with its category, its publication status (peer reviewed, preprint, or grey literature such as web pages, repositories and documentation) and, for every source that describes a concrete system, the reason it was or was not tabulated. The review cites 104 sources. Of these, 8 describe agentic systems that meet the three inclusion criteria below, 7 describe infrastructure or general methods for running agent workloads, 8 describe autonomous laboratories whose decisions are made by Bayesian optimization, reinforcement learning or active learning rather than by an LLM agent, and 12 describe LLM agents for science outside our scope. The remainder are foundational agent methods and frameworks (23), surveys, perspectives and position papers (22), classical HPC and workflow systems (6), benchmarks and empirical studies (9), protocols and safety frameworks (4), and methodological, critical or documentation sources (5).

A system enters Table 1 only if it meets three criteria. (C1) An LLM agent makes decisions that affect the execution of computational or experimental work on shared, resource-constrained infrastructure, as defined at the start of this section. (C2) The source reports at least one evaluation of the system, either quantitative or a documented case study; position papers and conceptual frameworks are excluded. (C3) The source can be identified unambiguously by its authors and a stable identifier. We screened all 50 sources that describe a concrete system against these criteria, and 8 meet all three. Of the 42 excluded, 39 fail C1 because no LLM agent makes the execution decision or because the system has no shared HPC or laboratory substrate (for example general agent frameworks [5,38,39], infrastructure that schedules agent workloads [12,22], non-LLM autonomous laboratories [25,27,40] and research automation that does not execute work on shared infrastructure [41,42]); 2 fail C2 (for example the safety framework Safe-SDL [43]); and 1 falls outside the scope definition because it runs on a single local device (TinyAgent [44]). Table S1 gives the reason for each exclusion. For contrast, Table 1 also shows four excluded systems that report concrete metrics (Panel B).

Two limits of this audit should be stated plainly. The classification was carried out retrospectively, during revision, by the authors. The numbers of records retrieved and screened during the original search were not logged, so an identification and screening flow in the sense of PRISMA 2020 [37] cannot be reconstructed. We therefore report the composition of the final corpus and the inclusion flow for Table 1, both of which can be checked against Table S1, rather than retrieval counts that we could not verify.

3. A Taxonomy of Agentic Roles in HPC and Scientific Pipelines

Systems in our corpus consistently decompose into a small number of recurring agent roles, which we organize below. We derived these four categories inductively rather than adopting an existing scheme from the general multi-agent literature: after reading the corpus described in Section 2, we grouped systems by the single question each agent

is primarily answering on behalf of a human operator: should I request this resource, should I change what happens next in this campaign, should I move or record this data, or should I run this physical experiment, rather than by implementation detail (framework, model family, or protocol used), since the same underlying role recurs across otherwise very different technical stacks. Several systems combine two or more of these roles inside a single orchestrator or a hierarchy of specialized sub-agents [10,45,46]. The taxonomy classifies agents, that is, LLM components that make execution decisions. Infrastructure that schedules or serves agent workloads without handing those decisions to an agent is a different thing and is treated separately at the end of Section 3.1; Figure 1 shows it as a substrate layer beneath the four roles. Figure 1 summarizes the taxonomy and the section in which each role is discussed.

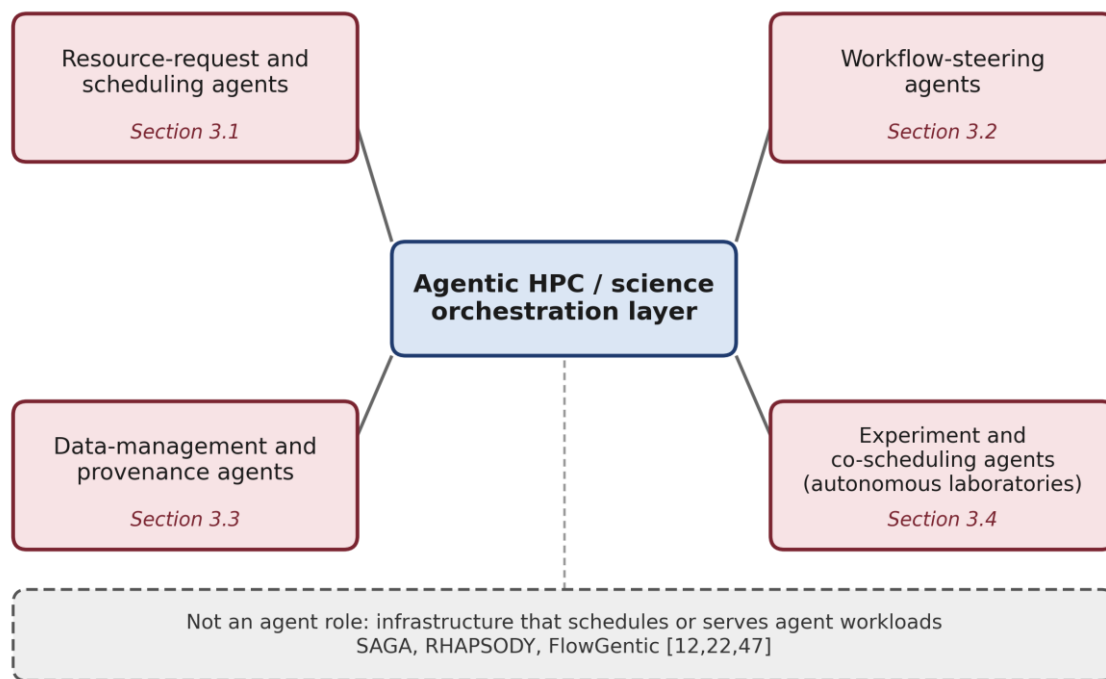


Figure 1. Taxonomy of agentic roles in HPC and scientific pipelines, derived inductively from the reviewed corpus (Section 2). The lower band is not an agent role: it shows infrastructure that schedules or serves agent workloads [12,22,47].

3.1 Resource-Request and Scheduling Agents

These agents translate a high-level scientific goal into concrete requests against a batch scheduler or cloud resource manager. Ma et al. [8] connect a LangChain/LangGraph agent to Parsl so that tool calls are dispatched either as concurrently queued Parsl tasks or as Parsl ensemble functions on the Polaris supercomputer, allowing a single LLM-issued instruction to fan out into many molecular dynamics simulations. Sochat and Milroy [9] give LLM agents the whole lifecycle of an HPC application run in the cloud: building containers, deploying them on Kubernetes, choosing among 21 instance types and orchestrating scaling studies for four HPC applications. In computational chemistry, an OpenClaw agent dispatches quantum chemistry and molecular dynamics jobs to a Slurm queue through the DPDispatcher library [46,48]. TinyAgent [44], which runs on a single local device and is therefore outside our scope, shows that small task-specific models can still call functions accurately when paired with a compiler-style planner, which suggests that resource-request agents need not always be large, cloud-hosted models. These agents differ sharply from a classical HPC scheduler, which optimizes over jobs with declared, static resource requirements; an agent issues requests that are generated dynamically and may change mid-execution, a dependency structure closer to the reasoning graphs of graph-of-thoughts prompting [49] than to a flat instruction sequence.

Scheduling for agents is a different problem. A separate line of work schedules the workloads that agents generate, rather than letting agents schedule. SAGA [22] treats a whole agent workflow, not each LLM call, as the unit

of scheduling on GPU clusters, and manages the key-value (KV) cache memory popularized by PagedAttention-style serving [50] jointly with the workflow’s control flow. RHAPSODY [12] is middleware that lets simulation, AI training, high-throughput inference and agent-driven control share one HPC allocation, and FlowGentic [47] is a library that connects agent frameworks such as LangGraph to HPC workflow engines. None of these lets an agent decide where its work runs, so we treat them as infrastructure beneath the four roles (Figure 1) and include SAGA and RHAPSODY in Table 1 only as contrast cases (Panel B).

3.2 Workflow-Steering Agents

A second class of agent monitors the intermediate state of a running scientific campaign and decides what to do next, for example, whether a simulation has converged and a new parameter regime should be explored, or whether a materials synthesis attempt failed and should be retried with a different recipe. Multi-agent orchestration for high-throughput materials screening on the Aurora exascale system [10] is a representative example. A central planning agent partitions the workload among parallel executor agents, which share a Model Context Protocol (MCP) server that runs tasks through Parsl, and the framework uses the open-weight gpt-oss-120b model to screen a metal-organic framework database [10]: because traditional HPC ensembles have static, predefined control flow, they cannot adapt to intermediate results without human intervention, whereas an agent layer can decide dynamically which candidate materials merit further simulation. In computational chemistry, El Agente [51] generates and runs quantum chemistry workflows from natural language and debugs failures in place, and its successor El Agente Gráfico [45] constrains the model’s decisions to checkpoints in a typed execution graph. Steering without being an agent is possible with related infrastructure: RHAPSODY [12] and FlowGentic [47] host the agent in the workflows of HPCs, and ROSE [23] steers the exploration by an LLM agent instead of the surrogate.

3.3 Data-Management and Provenance Agents

Scientific workflows increasingly move petabyte-scale datasets between instruments, edge devices and HPC storage tiers, and it is natural to expect agents to take on decisions about when and how to stage data. In the corpus reviewed here, however, we found no system in which an LLM agent autonomously controls bulk data movement; the in-transit transport strategies studied for coupled AI-simulation workflows [15] are designed and evaluated without an agent in the loop. The role is currently represented by data management and provenance. PROV-AGENT [52] extends the W3C PROV model to record the prompts, responses and decisions of agents within end-to-end workflow provenance, uses MCP and data observability for capture, and evaluates capture across edge, cloud and HPC environments. Its authors argue that the non-deterministic, potentially hallucinating nature of agent decisions makes queryable provenance more critical than in deterministic pipelines, not less. We therefore keep this role in the taxonomy as data management and provenance, and we flag autonomous data movement as an expected but not yet documented role.

3.4 Experiment and Co-Scheduling Agents in Autonomous Laboratories

Autonomous laboratories form a distinct but related lineage, in which decisions are embedded in a physical, robotic laboratory rather than a supercomputer. Most of these laboratories do not use LLM agents, and we separate them by decision method. Bayesian optimization or Bayesian active learning drives the mobile robotic chemist [25], CAMEO [28], the Ada thin-film platform [29], a fluidic laboratory for lead-free perovskite nanocrystals that uses ensemble neural network enabled Bayesian optimization [53], and the multi-robot Rainbow platform, which ranks candidates with Gaussian process models and a noisy expected hypervolume improvement criterion [54]. The Artificial Chemist uses machine learning based experiment selection [26]; AlphaFlow uses reinforcement learning [40]; and A-Lab combines natural-language models trained on text-mined synthesis literature with the ARROWS3 active-learning algorithm and neural network phase identification [27]. These systems belong to the broader materials-informatics trajectory toward autonomous discovery [24,55], but their results are not evidence about LLM agents, and we do not use them as such.

LLM agents come up in a limited number of lab-based systems. Coscientist [30] uses a GPT-4 planner that issues four kinds of command (web search, code execution, documentation retrieval and experiment execution) and drives liquid-handling robots and a cloud laboratory. El Agente [51], discussed in Section 3.2, applies the same pattern to computational rather than physical chemistry, so its constrained resource is compute rather than instrument time. Perspectives on managing materials laboratories with multi-agent AI [56] and work on few-shot grounded planning for embodied agents [57] suggest how LLM reasoning may be coupled to physical autonomy [36]. Laboratory agents face

the same co-scheduling problem as HPC agents, except that the constrained resource is instrument time and an incorrect decision can waste material or create an unsafe experiment rather than merely waste a job allocation [43].

4. Execution Substrates: From Lightly to Deeply Agentic

Before cataloging individual systems (Section 5), it is useful to place them on a single conceptual axis, since the degree of agentic autonomy a system grants turns out to predict much of what else is true about it. At one end, an LLM issues calls into an otherwise unmodified HPC workflow tool such as Parsl [8,58] or a graph-based scheduler such as Flux [59]; at the other, an autonomous laboratory lets an LLM propose, execute, and revise entire experimental protocols with minimal human oversight [30]. Figure 2 places the agentic system families discussed in Section 5 along this spectrum.

Rubric. Positions are assigned with a four-item rubric, each item scored 0, 1 or 2, so that a family's position is a score from 0 to 8. (R1) Plan origin: 0 if a human writes the workflow and the agent fills in parameters, 1 if the agent composes steps from a fixed tool set, 2 if the agent generates the plan and may rewrite it. (R2) Runtime control: 0 if the agent cannot change control flow after launch, 1 if it chooses among predefined branches, 2 if it can create new tasks or branches at run time. (R3) Reach of action: 0 for tool calls into a managed workflow system, 1 for direct job submission or resource requests on shared compute, 2 for physical actions on laboratory hardware. (R4) Human checkpoints: 0 if a human approves each consequential action, 1 if approval is given at campaign level, 2 if there is no routine approval within a campaign. The scores are our assessment from the descriptions in the sources, recorded per family in Table 2. Infrastructure that schedules agent workloads and non-LLM autonomous laboratories are not placed on the spectrum, because it measures the autonomy granted to an LLM agent.

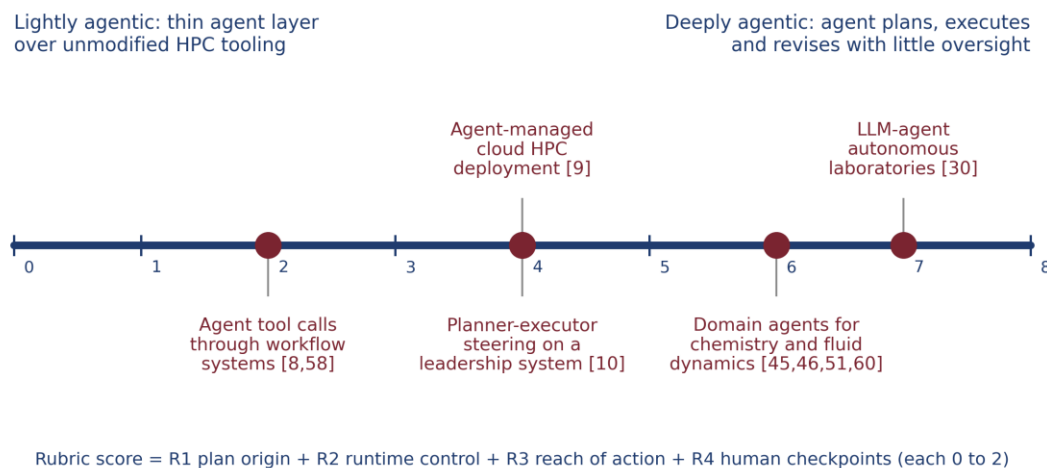


Figure 2. The lightly-agentic to deeply-agentic execution spectrum. Positions are rubric scores from 0 to 8 (Section 4; scores in Table 2): agent tool calls through workflow systems [8,58], agent-managed cloud HPC deployment [9], planner-executor steering on a leadership system [10], domain agents for computational chemistry and fluid dynamics [45,46,51,60], and LLM-agent laboratories [30].

This spectrum matters for the gap analysis in Section 7: the lightly agentic end inherits mature scheduling and reproducibility guarantees from the underlying classical system almost for free, because a thin agent wrapper around Parsl or Flux still benefits from decades of HPC scheduler engineering. The deeply agentic end must build these guarantees anew, since there is no equivalent classical system whose maturity an autonomous laboratory can borrow, which is where most of the open problems identified in Section 6 appear to concentrate. We therefore put forward, as a hypothesis rather than a finding, that position on this spectrum predicts which of the challenges in Section 6 a system will struggle with most. The hypothesis is testable: if systems were scored with the rubric and evaluated on a shared benchmark (Section 8), higher scores should go together with more frequent decision-level failures and weaker reproducibility guarantees. It has not been tested in the literature reviewed here.

5. Representative Systems

Table 1 compares the eight systems that meet the inclusion criteria of Section 2.3 (Panel A) and, for contrast, four adjacent systems that report concrete metrics but fall outside the scope because no agent makes the execution decision (Panel B). For each system the table gives the agentic role (Section 3), the execution substrate, the coordination interface, the evaluation actually reported, whether a baseline is compared, and the publication status of the source. In the baseline column, Yes means the source compares against an external system or a non-agentic alternative, and Partial means it compares only variants of its own design.

Table 1. Agentic HPC and science systems that meet the inclusion criteria (Panel A) and adjacent contrast cases (Panel B), by role, substrate, coordination interface, reported evaluation, baseline comparison and publication status. MOF: metal-organic framework; MD: molecular dynamics; SLO: service level objective; ALCF: Argonne Leadership Computing Facility.

System	Agentic role(s)	Execution substrate	Coordination interface	Reported evaluation	Baseline	Status
Panel A. In-scope agentic systems (meet C1, C2 and C3)						
Connecting LLM agent to HPC [8]	Resource request, task dispatch	Parsl on Polaris (ALCF) and a workstation	LangChain/LangGraph tools run as Parsl tasks	Qualitative: concurrent dispatch of molecular dynamics runs	No	Preprint
Agentic cloud HPC [9]	Resource request, deployment, scaling	Kubernetes cloud; 4 HPC applications, 21 instance types	Custom framework on the Gemini API	Figures of merit per application; scaling study to 5 nodes	Yes (LLM vs. user-provided strategy)	Preprint
Materials screening on Aurora [10]	Workflow steering (planner and executors)	Aurora exascale system, Parsl	Shared MCP server	Orchestration overhead and task completion on MOF screening	No	Preprint
PROV-AGENT [52]	Data management, provenance	Edge, cloud and HPC	MCP and data observability	Qualitative provenance capture case studies	No	Peer reviewed
El Agente [51]	Workflow steering, job execution	Quantum chemistry codes	Own multi-agent framework	Average task success above 87% on six course exercises; two case studies	No	Peer reviewed
El Agente Gráfico [45]	Workflow steering	Typed execution graphs with parallel execution	Typed execution graph runtime	About 80% lower model cost and over 4x lower wall-clock time on the same exercises	Yes (previous El Agente design)	Preprint
OpenClaw for chemistry [46]	Resource request, workflow steering	Slurm via DPDispatcher	Skills and tools	Qualitative reactive MD case study; about US\$7.6 of inference	No	Peer reviewed
Coscientist [30]	Experiment	Liquid-handling robots and a cloud laboratory	Four named commands	Case studies; planning scores across LLMs	Yes (other LLMs; Bayesian optimization)	Peer reviewed
Panel B. Contrast cases outside the scope (fail C1) that report concrete metrics						
SAGA [22]	Scheduling for agents	64-GPU cluster, vLLM	Not applicable	1.64x task completion time (geometric mean), 1.22x memory utilization, 99.2% SLO attainment, about 30% lower peak throughput	Yes (vLLM with prefix caching)	Preprint
RHAPSODY [12]	Runtime hosting agent control	Leadership-class HPC (Dragon, vLLM)	Not applicable	Overhead, inference scaling and coupling in case studies	Partial (runtime modes)	Preprint
LLMCompiler [61]	Parallel function calling	General tool-calling LLM	Planner and task-fetching unit	Up to 3.7x latency, 6.7x cost, about 9% accuracy vs. ReAct	Yes (ReAct)	Peer reviewed
ReWOO [62]	Plan then execute	General tool-calling LLM	Planner, workers, solver	Fewer LLM calls than ReAct	Yes (ReAct)	Preprint

Table 1 supports the argument of Section 7 in three ways. The metrics are heterogeneous by construction: task success rates, application figures of merit, cost and wall-clock reductions, and qualitative case studies. Only 3 of the 8 in-scope systems compare against any baseline [9,30,45], and no two share a workload, an LLM or a metric. And 4 of the 8 sources are preprints, so their numbers have not yet passed peer review and should be read as preliminary. This is not a criticism of individual papers, several of which evaluate carefully on their own terms, but of the absence of a shared protocol across the sub-field.

Interpretation of numbers from reports. The aggregate figures must be qualified by the conditions. SAGA's 1.64x reduction in task completion time is a geometric mean measured on a 64-GPU cluster serving SWE-bench coding agents and WebArena browser tasks, against vLLM v0.15.1 with prefix caching and affinity routing; the authors also report about 30% lower peak throughput than throughput-optimal batch scheduling [22]. The gain is therefore a trade between completion time and throughput, measured on software and web agents rather than scientific campaigns, and it comes from a preprint. LLMCompiler's speedups have passed peer review but were measured on general function-calling benchmarks, not on HPC workloads [61]. The cloud HPC agents of [9] report application figures of merit and a comparison between an LLM-chosen and a user-provided optimization strategy, but no quantitative human-expert baseline, and their scaling study reached 5 of a planned 32 nodes. El Agente reports task success above 87% on six university-level exercises without an external baseline [51], and El Agente Gráfico's cost and time reductions are relative to its own predecessor [45].

Even for a peer-reviewed result that is not part of the LLM, the details of the evaluation are important, as illustrated in A-Lab [27]. Its initial report of 41 novel compounds among 58 targets in 17 days was disputed because the automated X-ray diffraction analysis was unreliable and because atomic disorder was not considered [63] and an author correction was issued in 2026, claiming that the materials were not new compounds, but rather new compounds from the prediction platform [64]. The paper makes no comparison with other methods that are not autonomous, or other recipes created by humans, its comparisons are internal, such as comparing the success of active learning with that of literature-inspired recipes for nine targets [27]. So, the rate of synthesis cannot be compared with that of a classical Bayesian optimization campaign on the basis of the source and the lack of a baseline is not a problem in the original report but in our reading of it.

Table 2 groups the agentic system families discussed in this review, lists their members and gives their rubric scores (Section 4). Parsl- and workflow-manager-mediated systems [8,58] keep the agent layer thin; cloud-native orchestration [9] extends the same idea to Kubernetes; and domain agents in computational chemistry and fluid dynamics [45,46,51,60] combine LLM reasoning with electronic-structure codes, solvers and retrieval-augmented literature search [65,66]. Several further systems fall outside the spectrum because they do not execute work on shared infrastructure: hypothesis generation and research automation platforms [41,42,67,68,69,70,71], and retrieval and tool augmented assistants for scientific facilities [72]. They raise reproducibility and epistemic-validity questions that go beyond scheduling [73,74], and program-search systems that produce automatically verifiable artifacts, such as FunSearch-style mathematical discovery [75], are a useful contrast because their outputs sidestep much of that concern.

Table 2. Agentic system families, their members and their rubric scores on the autonomy spectrum (Section 4). Each item is scored 0, 1 or 2; the score is the sum.

Family	Members	R1 plan	R2 runtime	R3 reach	R4 checkpoints	Score
Agent tool calls through workflow systems	[8,58]	1	0	0	1	2
Agent-managed cloud HPC deployment	[9]	1	1	1	1	4
Planner-executor steering on a leadership system	[10]	2	1	0	1	4
Domain agents for computational chemistry and fluid dynamics	[45,46,51,60]	2	2	1	1	6
LLM-agent autonomous laboratories	[30]	2	2	2	1	7
Not placed on the spectrum: infrastructure for agent workloads [12,22,47]; non-LLM autonomous laboratories [25,26,27,28,29,40,53,54]; research automation without execution on shared infrastructure [41,42,67,68,69,70,71].						

6. Cross-Cutting Technical Challenges

Across otherwise unrelated systems, six challenges recur consistently enough that we treat them as the defining open problems of the field. The first four (Sections 6.1 to 6.4) concern how agents decide and compute; the last two (Sections 6.5 and 6.6) concern the boundaries and interfaces through which they act. Section 6.7 relates them and states which links are supported by evidence and which are hypotheses.

6.1 Hallucination Propagation Through Distributed Pipelines

Because agent decisions in an HPC or scientific pipeline are not isolated text outputs but triggers for real computation or physical experiments, a hallucinated intermediate judgment can propagate and compound across pipeline stages in a way that resembles error propagation in distributed systems more than it resembles hallucination in a single chatbot turn [16]. The Multi-Agent System Failure Taxonomy (MAST) [76] classifies 14 failure modes into three categories (system design issues, inter-agent misalignment and task verification) and concludes that failures are not only about the model capability, but often system design and coordination too, with more than 1600 annotated traces across seven frameworks. The Multi-Agent System Failure Taxonomy (MAST) [76] classifies 14 failure modes into three categories (system design issues, inter-agent misalignment and task verification) and argues that many failures are not about the model capability, but they are also about system design and coordination. [34] develops the same argument as an architectural argument. This distinction is particularly relevant for HPC and scientific applications, as it highlights the need to approach the issue differently from the standard one of just using a more powerful underlying LLM; when the most common failure mode is a misaligned handoff of tasks between the planning agent and the execution agent, or the execution agent silently misinterpreting the planner's response, then the fix must be implemented somewhere else—namely, in the coordination and verification layer between the two agents.

In essence, PROV-AGENT [52] argues from the provenance side, that provenance should be transparent and queryable because agents can hallucinate or reason incorrectly, and that one agent's output is frequently the other agent's input, making it impossible to see how far the hallucination traveled and when it was introduced. In an HPC context this adds to the scale of the underlying computation: a hallucinated judgement that a simulation has converged (or that a materials candidate is promising) can lead to the allocation of significant compute downstream (Section 6.3) towards a direction that a human might not notice for a while, and is a failure mode that has no obvious analogue in single-turn chatbot use.

6.2 Scheduling Visibility and Coordination Overhead

Scheduling visibility is listed as an open problem in the call for papers of the AGENT4SC workshop [11], and the systems in Table 1 show why it arises. In [8] and [10] the agent hands tasks to Parsl, which decides placement, and in [46] jobs reach a Slurm queue through DPDispatcher [48]; in each case the agent sees the outcome of a request but not the queue state that determines when it will be honoured. The clearest evidence on coordination overhead comes from a controlled study of 260 configurations covering six agentic benchmarks, five architectures and three LLM families: relative to a single agent, multi-agent coordination improved performance by up to 80.8% on decomposable tasks and degraded it by up to 70.0% on sequential ones, and architectures without centralized verification tended to propagate errors more [77]; a recent survey reaches a similar conclusion [31]. This is similar to the diminishing returns that Amdahl's law predicts of classical parallel computing, but with the added twist that the granularity of tasks and possibly even the dependency structure of the tasks could only become known once the planner has finished committing his plan [61, 62, 78].

Further complicating the situation is single-agent reasoning strategies. Chain-of-thought prompting [79] and tree-of-thoughts search [80] were developed to enhance the answer of one model by reasoning, but a deeper model reasoning increases planning latency and memory usage, which can be a negative effect on the efficiency of multi-agent scheduling when multiple agents are executed in parallel. Unlike Skeleton-of-Thought [81], which produces portions of a single answer concurrently, it generates multiple answers concurrently. It produces multiple answers concurrently, contrary to Skeleton-of-Thought [81] which produces portions of a single answer concurrently. Coordination benchmarks from the general multi-agent literature (e.g., [82, 83]) measure and quantify the degree of synchronization of agents, and a survey of open problems (e.g., [84]) identifies techniques for reducing the cost of achieving coordination. Table 3 contains a list of these techniques, along with the potential for using them in HPC; these techniques are not being widely used in the systems in Table 1 and represent a straightforward transfer opportunity with a low risk.

Table 3. Techniques for coordinating efficiency from the general multi-agent literature, and their potential applications for agentic HPC systems.

Technique	Mechanism	Source	Possible use in agentic HPC
Message pruning	Removes redundant inter-agent messages	[85]	Fewer coordination calls between planner and executors
Agent dropout	Drops agents that are not contributing	[86]	Shrink an executor pool in low-value phases
Asynchronous function calling	Non-blocking tool calls	[87]	Overlap job submission with planning
Parallel function calling	Calls planned as a dependency graph and run in parallel	[61]	Fan out independent simulations
Plan then execute	One planning pass, fewer LLM calls	[62]	Campaigns with a known structure
Allocation among agents	LLM-based allocation of tasks and resources to agents	[88]	Per-agent compute budgets
Topology optimization	Searches over agent organizations	[89]	Choose planner and executor fan-out

6.3 Energy Accounting

Energy Accounting is listed as an open problem in the 'open papers' call in AGENT4SC [11] and a recent survey [90] mentions it but without quantifying it in the same paper. A recent survey proposes a formal energy-accounting framework spanning the perception-reasoning-action cycle of an agent, distinguishing computational from communication energy costs, and argues that agentic AI faces compounding energy costs that traditional AI accounting based on counting floating point operations (FLOPs) does not capture, since every additional planning step, retry, or inter-agent message adds both inference and network energy that a single-pass model would never incur [90]. That survey also identifies mitigation levers, model simplification, computation control, and attention- or input-level pruning, but points out that virtually none of these have been explored in an HPC- or scientific-computing-type environment with limited resources; its proposed roadmap items (federated green learning, carbon-aware scheduling of agent workloads, and 6G-native agentic inference) are targeted to mobile-edge and wireless networking settings, not shared HPC clusters [90].

The energy-accounting problem has been refined in at least three respects when applied to HPC settings, aspects not fully captured by the energy-agency general literature. The first is that if an agent makes an imperfect causal judgement about the cause of a simulation or experiment failure, it can re-run the expensive job multiple times until it succeeds or a human corrects the mistake, but in a traditional, deterministically scripted HPC campaign a failure simply ends the pipeline. Second, if the agent is making exploratory branches, such as multiple possible parameter regimes that are evaluated in parallel by a planner-executor architecture like Aurora materials screening [10], the energy used by the branches that are never used is wasted even if most of them fail to yield a promising task, and this “failed-branch” energy is not captured in the performance metrics that such systems usually report, namely task completion rate. Third, messages of coordination between agents have to traverse a shared MCP server [10] or a peer-to-peer protocol [91] each, requiring at least one additional LLM inference pass, with the energy cost of coordination increasing with the number of messages exchanged (measured in joules not seconds). None of the systems in Table 1 report on the accounting of the three cost sources against the useful scientific output that they produce, which is just the type of cost-per-unit-of-science measure to which we will return in Section 7.

6.4 Reproducibility

Traditional HPC and scientific workflows enforce reproducibility through deterministic control flow and tightly specified inputs. Agentic workflows instead produce results through iterative, data-driven, and often stochastic decision processes, which means reproducing a workflow may require replicating the conditions and logic of a decision process rather than replicating an identical output trace [92]. Benchmarks such as CORE-Bench [93] and Repro-Bench [94] attempt to directly measure whether agentic systems can even reproduce existing, already-published computational results, and find this to be a nontrivial capability rather than a given: AI coding agents attempting to reproduce social-science findings show meaningfully imperfect success rates [95]. The most significant proposed remedy for the field is the so-called provenance-first approaches like PROV-AGENT [52] and the call for improved, feedback-aware provenance models [92] – but the latter ones aren't actually being used.

There are two meanings of reproducibility that are often mixed up in the HPC-agent literature. The first is output reproducibility, whether a re-run of the same agentic pipeline on the same input produces the same scientific output (the

traditional HPC notion, which directly threatens to agentic non-determinism). The second question is whether the series of intermediate decisions that an agent made in its execution – whether to simulate a next candidate, whether to request a new resource for the current candidate, whether to retry a given experiment – can be examined and, if needed, repeated – even if the final numerical output doesn't agree bit for bit. Classical HPC tooling was built almost entirely around the first sense, since deterministic scripts made the second sense trivially true; agentic systems break that equivalence, because two runs can reach the same final answer via very different, and differently defensible, chains of intermediate reasoning, or reach different final answers via superficially similar chains. CORE-Bench and Repro-Bench [93,94] currently evaluate only the first sense. AstaBench [96] and PaperBench [97] move slightly closer to the second by scoring intermediate research artifacts rather than only final answers, but neither was designed with HPC resource-decision logging in mind. We see decision reproducibility, not output reproducibility, as the harder and more distinctively agentic problem, and therefore as the more promising target for the provenance-and-replay research direction we return to in Section 8.

6.5 Trust, Authorization, and Safety Boundaries

One idea that threads themselves through both the HPC-cluster and the autonomous-laboratory arms of this literature is that there's no body systems of authorization between an agent's claimed intention and its implementation on a shared expensive infrastructure. The case of an over-allocated agent on a leadership-level system consumes queue time and energy that could otherwise be used by other users, and on a self-driving laboratory, the same kind of misallocation can harm the instruments or lead to a dangerous reaction [43]. Safe-SDL [43] is one of the first attempts to formalize safety boundaries and real-time control mechanisms specifically for AI-driven laboratories, and its authors explicitly note that existing operational frameworks such as UniLabOS and Osprey address system architecture and human oversight without providing a unified safety theory spanning design, real-time control, and transactional execution. No directly analogous safety framework yet exists for HPC-cluster-embedded agents, despite the fact that the failure modes, over-allocation, redundant execution, unsafe co-scheduling with other tenants, are conceptually similar.

What would an analogous layer for cluster-embedded agents contain? Drawing on the failure modes above and on mechanisms that schedulers already provide, we see five primitives. (i) Budgets: per-agent quotas in node-hours or GPU-hours, enforced by the scheduler's accounting rather than by the agent's prompt. (ii) Validation before execution: a request is checked before it is honoured, as Slurm already allows with `sbatch --test-only`, which validates a job script and estimates its start time without submitting it [98]. (iii) Reversibility classes: actions are tagged as read-only, cancellable or irreversible, and only the first two run without approval; dispensing a reagent is irreversible in a way that cancelling a queued job is not [43]. (iv) Limits on resubmission, which bound the redundant re-execution discussed in Section 6.3. (v) An approval and audit trail kept in the same provenance store that records agent decisions [52], so that every authorization decision can be replayed (Section 6.4).

These primitives connect the challenges of this section. Budgets and resubmission limits cap the compute that a hallucinated judgment can trigger (Section 6.1) and therefore its energy cost (Section 6.3); validation before execution partly compensates for weak scheduling visibility (Section 6.2); and typed protocols make the audit trail cheap to record (Section 6.6). We did not find such a layer described in the sources summarized in Table 1, and agents that operate computers directly through their graphical interfaces, as in Agent S [99], widen the set of actions that would need one.

6.6 Interoperability and Communication Protocols

As agentic HPC and scientific pipelines increasingly involve more than one agent, or agents built on different frameworks, a parallel thread of work has emerged around standardized communication protocols. The Model Context Protocol (MCP) provides a JSON-RPC client-server interface for tool invocation and typed data exchange, while the Agent2Agent (A2A) protocol targets peer-to-peer task delegation between otherwise opaque, independently developed agents, and the Agent Communication Protocol (ACP) and Agent Network Protocol (ANP) add, respectively, RESTful synchronous/asynchronous messaging and decentralized-identity-based peer discovery [91,100,101,102,103]. For HPC-scale orchestration specifically, an emerging instrument-facing analogue is the Agent-to-Instrument protocol, which aims to standardize how an agent addresses heterogeneous laboratory hardware without a hand-coded driver for every device and communication standard, a burden its authors describe as the bespoke-driver tax paid by essentially every self-driving laboratory built to date [104].

These protocols matter for our purposes for two concrete reasons beyond simple standardization. First, they specify the granularity of coordination overhead discussed in Section 6.2: If a protocol requires a full LLM call when each agent

sends another agent an inter-agent message, then coordinating the agents will require an LLM call for every message; however, if a protocol allows structured, non-generative message passing between agents (more akin to a typed remote procedure call than a natural-language exchange), the overhead can in principle be separated from the cost of LLM inference entirely. There are no protocols surveyed here that make this distinction an explicit part of the protocol - both MCP and A2A are still compatible with (but not with a requirement for) an LLM call on every message. Second, Protocol choice affects the reproducibility problem of Section 6.4: Protocol is structured and typed, so the complete, replayable trace of each invocation of each tool and each inter-agent request can be logged much more easily than it could be for a system using ad hoc, natural-language coordination between agents, making protocol adoption a more tractable near-term lever on reproducibility than post-hoc provenance tooling [52]. As of today, the coordination interface of each of these systems is shown in Table 1. The Aurora screening framework, whose executors share an MCP server [10] and PROV-AGENT [52] are two of the eight that use MCP. El Agente Gráfico uses typed execution graphs [45] to limit model decisions. The remaining five are run through specific interfaces such as LangChain/LangGraph tools [8] (custom framework for the Gemini API [9]), Skills and tools in OpenClaw [46] (dedicated multi-agent framework [51] and named commands in Coscientist [30]). This restricts the extent to which the lessons learned from the layer of coordination of one system can be transferred to another.

Adoption could be measured, and encouraged, in two simple ways. Papers can now report on which interface tool calls and inter-agent messages are carried, as Table 1 does. They can also provide reports on the completeness of their traces, which is the ratio of the number of agent actions in the trace that are consequential actions, that is, actions that could be replayed to produce the trace. Both might be necessary for a shared benchmark (Section 8) and make the protocol question measurable part of the reproducibility.

6.7 Comparative Summary of Open Problems

The challenges are interdependent, and the relationship between them is made explicit and clear about the evidential status of the challenges (Table 4). Agent error, whether it comes from hallucination (Section 6.1) or from coordination defects, which the available evidence suggests are the more common source of multi-agent failures [76], is what makes weak scheduling visibility (Section 6.2) and weak authorization (Section 6.5) costly: if agents never erred, these gaps would be inefficient rather than unsafe. We hypothesize that much of the avoidable energy cost of agentic campaigns (Section 6.3) comes from re-execution after incorrect decisions rather than from running the LLM itself; this remains a hypothesis because, as Section 6.3 shows, no study we reviewed measures these costs jointly. Reproducibility (Section 6.4) and interoperability (Section 6.6) are largely tooling and provenance problems that become tractable once the field agrees on what a replayable decision trace should contain. Self-resource-allocation mechanisms [88] and automated topology optimization [89] point to the same idea from another angle: coordination structure can be treated as a tunable, evaluable design parameter. A single provenance, authorization and resource-negotiation layer may therefore be a more efficient research investment than separate point solutions, which is again a proposal to be tested rather than a result.

Table 4. Evidential status of the main claims linking the challenges in Sections 4 and 6.

Claim	Section	Status	Evidence or test
Multi-agent coordination helps on decomposable tasks and hurts on sequential ones	6.2	Supported	260-configuration study [77]
Coordination and system-design defects are a major source of multi-agent failures	6.1, 6.7	Supported outside HPC	MAST, more than 1,600 traces [76]; not yet measured on HPC workloads
A hallucinated judgment in an HPC pipeline can trigger substantial wasted compute	6.1	Hypothesis	Inject faulty judgments in a benchmark campaign and meter the compute they trigger
Most avoidable energy comes from re-execution after agent errors	6.3, 6.7	Hypothesis	Requires joint energy accounting (Section 8)
Coordination overhead grows with message volume when every message needs an LLM call	6.3, 6.6	Reasoned, not measured	Count LLM calls and energy per message
Position on the autonomy spectrum predicts the dominant challenges	4	Hypothesis	Score systems with the rubric and compare failure profiles on a shared benchmark
Typed protocols make decision traces easier to replay than ad hoc coordination	6.6	Reasoned, not measured	Report trace completeness

7. Evaluation Methodology: The Central Gap

Having surveyed architectures and cross-cutting challenges, we now argue for a central, unifying thesis of this review: the agentic-AI-for-HPC-and-science literature has substantially more architectures than it has shared, comparable evaluation methodology, and this, not any single architectural shortcoming, is the field's most serious limitation.

This is made concrete by three symptoms. First, it is almost impossible to compare the performance claimed by reported systems (Table 1). While metrics like orchestration overhead, task completion rate, or latency speedup are reported in papers like [10,61] on different types of clusters, with different underlying LLM(s) and workloads – typically custom ones – it would not make sense to compare a claimed improvement in one paper to a claimed improvement in another. Second, benchmarks do exist (CORE-Bench [93], PaperBench [97], AstaBench [96], Repro-Bench [94]), but they are all for general autonomous-research applications in a broader context, and will not measure the quantities that are relevant to parallel computing – wall-clock time under contention, cost-per-unit-of-science, scheduling fairness, energy per decision – that would allow a systems researcher to compare two agentic HPC schedulers head to head. Third, open problems mentioned in paper introductions and conclusions, including the open problems of the field named in the AGENT4SC call for papers [11] and challenges of Section 6, are not really experimental issues that the paper measures, but rather are acknowledged as limitations..

This is where a background in parallel computing really pays off: there is a well-established methodology for benchmarking schedulers, for reporting communication-to-computation ratio, and for reporting energy per operation, which have yet to be systematically adopted into agentic-AI-for-science evaluation.

8. Research Agenda and Future Directions

Based on the gap analysis above, we lay out some specific paths for further research that can be acted upon by research communities with parallel or distributed computing experience.

Similar benchmark suites for HPC scheduling with agents. Four things should be grounded in a benchmark and five should be reported, as illustrated in Table 5. It fixes the substrate (a replayed production trace on a scheduler simulator, plus a small real allocation for validation), the workloads (three templates: a parameter sweep of independent tasks, an iterative simulation with sequential dependence, and an ensemble campaign whose next step depends on intermediate results), the model (one open-weight LLM pinned by version, with temperature and seed recorded), and the baselines (a scripted workflow on the same workflow engine, and a classical scheduler policy with no agent). It produces metrics for wall-clock time under contention, cost and energy per unit of scientific output, agent types and errors, completeness of traces (Section 6.6), and agent added value, which is the improvement over the scripted baseline. Governance can meet community expectations like SPEC/MLPerf, which is a public task specification, versioned releases, and submission rules that mandate submission of released traces.

Table 5. A concrete design for a shared agentic HPC benchmark.

Component	Proposal
Substrate	Replayed production job trace on a scheduler simulator; small real allocation to validate the simulator
Workload templates	Parameter sweep (independent tasks); iterative simulation (sequential); adaptive ensemble campaign
Model	One open-weight LLM pinned by version; temperature, seed and prompts released
Baselines	Scripted workflow on the same engine; classical scheduler policy without an agent
Metrics	Wall-clock time under contention; cost and energy per unit of scientific output; agent errors by type; trace completeness; agent added value over the scripted baseline
Fault injection	Planted faulty judgments and failed jobs, to measure detection and the compute each fault triggers
Governance	Public specification, versioned releases, submissions must include full decision traces

Agent-issued agent-directed acyclic graph (DAG) scheduling (dynamic and re-plannable). A natural extension of LLMCompiler's static task-fetching-unit model [61] and ReWOO's separation of plan and execution [62] is to have a scheduler that detects when a new intermediate result invalidates part of an already-committed execution graph and re-schedules only the affected subgraph, instead of re-planning all or just continuing a now-stale plan.

Formal cost models for agent parallelism granularity. A predictive model, similar to that used for estimating the granularity or Amdahl's-law of a classical parallel application in advance to decide whether to use a single agent to execute the application or multiple agents, would allow the orchestrators to make this decision without being required to experiment by trial and error with a particular application.

Accounting for agentic decisions, standardized energy and cost. Extending frameworks such as the perception-reasoning-action energy model [90] to the HPC setting, with explicit attribution of energy cost to agent-caused reruns, exploratory branches that are later discarded, and coordination messages between agents, would convert an acknowledged concern into a measurable and optimizable quantity.

Verifiable, replay able provenance for autonomous scientific decisions. Extending PROV-AGENT-style provenance capture [52] with a replay mechanism (one that can re-execute an agent's recorded decision logic, not merely its final output) would directly address the reproducibility gap identified in Section 6.4 and in [92].

Safety and trust boundaries for physically-embodied and cluster-embedded agents. Work such as Safe-SDL [43] for autonomous laboratories has begun to formalize safety boundaries for physically dangerous agent actions; an analogous, systematically developed safety and authorization layer for agent-issued HPC resource requests, bounding cost, blast radius, and reversibility before an agent's request is honored on a shared, expensive, multi-tenant cluster, remains comparatively underdeveloped; Section 6.5 sets out five primitives such a layer could start from.

Make agents do less, and prove that they help. A less obvious recommendation follows from Table 1 and Section 4. The most useful near-term work may be to make agents do less rather than more. Every baseline-free evaluation in Table 1 leaves open whether the agent added anything that a scripted workflow on the same engine would not have delivered, and the lightly agentic end of the spectrum inherits scheduling and reproducibility guarantees that the deeply agentic end must rebuild. We therefore suggest that papers report agent added value against a scripted baseline as a matter of course, that designers confine LLM calls to the decision points where they demonstrably add value and use typed, non-generative messages elsewhere (Section 6.6), and that a result showing an agent does not beat the scripted baseline be treated as a contribution rather than a failure.

8.1 Limitations of This Review

Three limitations of the present review should be stated plainly; in keeping with the emphasis we place elsewhere on rigorous self-assessment. First, because agentic AI for HPC and large-scale science is only two to three years old as a distinct sub-area, many of the systems discussed here are workshop papers, preprints or position papers rather than peer-reviewed, extensively validated studies; Table 1 and Table S1 now mark the publication status of every source, and a quantitative claim from a preprint should be read as an initial, not yet independently replicated, finding. Second, our search, while spanning arXiv listings, workshop proceedings and domain journals, was iterative, carried out by a single reviewer, and not logged; it cannot claim the exhaustiveness of a PRISMA-style systematic review [37], and the classification in Table S1 was made retrospectively. We have prioritized breadth across the taxonomy of Section 3 over completeness within any single sub-category. Third, the rubric positions in Table 2 and the evidential labels in Table 4 are our judgments, made explicit so that others can dispute them. These limitations leave a tension that a careful reader will notice: a review that calls for evaluation rigor is itself a narrative review. We have tried to narrow that gap by making the corpus auditable, and we regard a full systematic review of this literature as valuable future work.

9. Conclusion

Agentic AI is rapidly being adopted as a control layer for high-performance computing and large-scale scientific discovery across autonomous laboratories, hybrid AI-simulation workflow engines, and agent schedulers for cloud and cluster. This survey categorizes literature within it according to the roles agents play, the architectures that are used to map agentic reasoning to real execution substrates, and six common problems: hallucination propagation, scheduling visibility, energy accounting, reproducibility, trust and authorization, and interoperability. The near total absence of a shared and comparable evaluation methodology (3 of the 8 in-scope systems compare with some baseline) is the most important of our findings, and the lack of such methods is especially problematic in the field, since they are not the major missing components of the field of field computing; in fact, researchers with training in parallel and distributed systems benchmarking have a particularly strong opportunity to fill this void. We hope that the taxonomy, synthesis and research agenda provided here speed the next phase of development in the field.

10. Conflict of Interest

All of the authors declare that they have no competing financial interests or personal relationships that could have resulted in bias in this report.

11. References

1. S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, "ReAct: Synergizing reasoning and acting in language models," in Proc. ICLR, 2023. arXiv:2210.03629.
2. T. Schick, J. Dwivedi-Yu, R. Dessi, et al., "Toolformer: Language models can teach themselves to use tools," in Proc. NeurIPS, 2023. arXiv:2302.04761.
3. N. Shinn, F. Cassano, E. Berman, et al., "Reflexion: Language agents with verbal reinforcement learning," in Proc. NeurIPS, 2023. arXiv:2303.11366.
4. G. Wang, Y. Xie, Y. Jiang, et al., "Voyager: An open-ended embodied agent with large language models," arXiv:2305.16291, 2023.
5. Q. Wu, G. Bansal, J. Zhang, et al., "AutoGen: Enabling next-gen LLM applications via multi-agent conversation," arXiv:2308.08155, 2023.
6. S. Hong, X. Zheng, J. Chen, et al., "MetaGPT: Meta programming for a multi-agent collaborative framework," in Proc. ICLR, 2024. arXiv:2308.00352.
7. J. S. Park, J. O'Brien, C. J. Cai, et al., "Generative agents: Interactive simulacra of human behavior," in Proc. ACM UIST, 2023. arXiv:2304.03442.
8. H. Ma, A. Brace, C. Siebensschuh, G. Pauloski, I. Foster, and A. Ramanathan, "Connecting large language model agent to high performance computing resource," arXiv:2502.12280, 2025.
9. V. Sochat and D. Milroy, "Agentic orchestration of HPC applications in cloud," arXiv:2607.02925, 2026.
10. T. D. Pham, H. Tummalapalli, F. H. Bhuiyan, A. Vazquez Mayagoitia, C. Simpson, R. Balin, V. Vishwanath, and M. Kececi, "Multi-agent orchestration for high-throughput materials screening on a leadership-class system," arXiv:2604.07681, 2026.
11. AGENT4SC 2026: 1st Workshop on Agentic AI for Large-scale Science, co-located with IEEE eScience 2026, <https://agent4sc.github.io/> (accessed Oct. 2026).
12. A. Alsaadi, M. Hooten, M. Goliyad, A. Merzky, A. Shao, M. Titov, T. Wang, Y. Chen, M. Kalantzi, K. Lee, A. Park, I. Pimpalkhare, N. Radcliffe, C. Wahl, P. Mendygral, M. Turilli, and S. Jha, "RHAPSODY: Execution of hybrid AI-HPC workflows at scale," arXiv:2512.20795, 2025.
13. W. Brewer, A. Gainaru, F. Suter, F. Wang, M. Emani, and S. Jha, "AI-coupled HPC workflow applications, middleware and performance," arXiv:2406.14315, 2024.
14. S. Jha, V. Pascuzzi, and M. Turilli, "AI-coupled HPC workflows," in Artificial Intelligence for Science: A Deep Learning Revolution, A. Choudhary, G. Fox, and T. Hey, Eds. Singapore: World Scientific, 2023, ch. 28, pp. 515-534.
15. H. Tummalapalli, R. Balin, C. Simpson, A. Park, A. Alsaadi, A. E. Shao, W. Brewer, and S. Jha, "In-transit data transport strategies for coupled AI-simulation workflow patterns," in Proc. SC'25 Workshops, 2025, pp. 985-996.
16. X. Lin, Y. Ning, J. Zhang, Y. Dong, Y. Liu, Y. Wu, X. Qi, N. Sun, Y. Shang, K. Wang, et al., "LLM-based agents suffer from hallucinations: A survey of taxonomy, methods, and directions," arXiv:2509.18970, 2025.
17. B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "LLM+P: Empowering large language models with optimal planning proficiency," arXiv:2304.11477, 2023.
18. X. Zhang et al., "A survey of multi-AI agent collaboration: Theories, technologies and applications," in Proc. 2nd Guangdong-Hong Kong-Macao Greater Bay Area Int. Conf. Digital Economy and Artificial Intelligence, ACM, 2026.
19. H. Zhang, Y. Li, W. Huang, Z. Hou, Y. Song, X. Liu, F. Effaty, J. Jiang, S. Wu, Q. Ding, et al., "Towards agentic intelligence for materials science," arXiv:2602.00169, 2026.
20. S. Alvanakis Apostolou, J. Bosch, and H. Holmström Olsson, "Assistance to autonomy: A systematic literature review of agentic AI across the software development life cycle," arXiv:2605.15245, 2026.
21. S. Khapre et al., "Agentic AI systems: A systematic survey of multi-agent architectures, cognitive foundations, interaction, explainability, security, and performance evaluation," Neurocomputing, 2026.

22. D. Guo, J. Wu, and S.-M. Yiu, "SAGA: Workflow-atomic scheduling for AI agent inference on GPU clusters," arXiv:2605.00528, 2026.
23. A. Alsaadi, T. Wang, A. Park, et al., "ROSE: Radical orchestrator for surrogate exploration," in Proc. SC'25 Workshops, 2025, pp. 61-70. doi:10.1145/3731599.3767347.
24. F. Hase, L. M. Roch, and A. Aspuru-Guzik, "Next-generation experimentation with self-driving laboratories," Trends in Chemistry, vol. 1, no. 3, pp. 282-291, 2019.
25. B. Burger, P. M. Maffettone, V. V. Gusev, et al., "A mobile robotic chemist," Nature, vol. 583, pp. 237-241, 2020.
26. R. W. Epps, M. S. Bowen, A. A. Volk, et al., "Artificial chemist: An autonomous quantum dot synthesis bot," Advanced Materials, vol. 32, p. 2001626, 2020.
27. N. J. Szymanski, B. Rendy, Y. Fei, et al., "An autonomous laboratory for the accelerated synthesis of inorganic materials," Nature, vol. 624, pp. 86-91, 2023.
28. A. G. Kusne, H. Yu, C. Wu, et al., "On-the-fly closed-loop materials discovery via Bayesian active learning," Nature Communications, vol. 11, p. 5966, 2020.
29. B. P. MacLeod, F. G. L. Parlane, T. D. Morrissey, et al., "Self-driving laboratory for accelerated discovery of thin-film materials," Science Advances, vol. 6, p. eaaz8867, 2020.
30. D. A. Boiko, R. MacKnight, B. Kline, and G. Gomes, "Autonomous chemical research with large language models," Nature, vol. 624, pp. 570-578, 2023.
31. S. Qi, J. Ma, R. Xing, W. Guo, X. Huang, Z. Gao, J. Deng, J. Liu, L. Zhang, B. Wei, et al., "Beyond individual intelligence: Surveying collaboration, failure attribution, and self-evolution in LLM-based multi-agent systems," arXiv:2605.14892, 2026.
32. Y. Zhu, L. Liu, J. Yu, and D. Zhang, "LLM-based multi-agent orchestration: A survey of frameworks, communication protocols, and emerging patterns," Future Internet, vol. 18, no. 6, p. 326, 2026.
33. T. Guo, X. Chen, Y. Wang, R. Chang, S. Pei, N. V. Chawla, O. Wiest, and X. Zhang, "Large language model based multi-agents: A survey of progress and challenges," arXiv:2402.01680, 2024.
34. M. Nechepurenko and P. Shuvalov, "Coordination as an architectural layer for LLM-based multi-agent systems," arXiv:2605.03310, 2026.
35. M. Gridach, J. Nanavati, K. Zine El Abidine, L. Mendes, and C. Mack, "Agentic AI for scientific discovery: A survey of progress, challenges, and future directions," arXiv:2503.08979, 2025.
36. G. Tie, P. Zhou, and L. Sun, "A survey of AI scientists," arXiv:2510.23045, 2025.
37. M. J. Page, J. E. McKenzie, P. M. Bossuyt, et al., "The PRISMA 2020 statement: An updated guideline for reporting systematic reviews," BMJ, vol. 372, p. n71, 2021. doi:10.1136/bmj.n71.
38. Q. Wang, T. Wang, Z. Tang, Q. Li, N. Chen, J. Liang, and B. He, "MegaAgent: A large-scale autonomous LLM-based multi-agent system without predefined SOPs," in Findings of ACL, 2025, pp. 4998-5036.
39. W. Zhang, C. Cui, Y. Zhao, Y. Liu, and B. An, "AgentOrchestra: A hierarchical multi-agent framework for general-purpose task solving," arXiv:2506.12508, 2025.
40. A. A. Volk, R. W. Epps, D. T. Yonemoto, B. S. Masters, F. N. Castellano, K. G. Reyes, and M. Abolhasani, "AlphaFlow: Autonomous discovery and optimization of multi-step chemistry using a self-driven fluidic lab guided by reinforcement learning," Nature Communications, vol. 14, p. 1403, 2023.
41. A. Ghafarollahi and M. J. Buehler, "SciAgents: Automating scientific discovery through bioinspired multi-agent intelligent graph reasoning," Advanced Materials, vol. 37, p. e2413523, 2025.
42. J. Gottweis, W.-H. Weng, A. Daryin, et al., "Towards an AI co-scientist," arXiv:2502.18864, 2025.
43. Z. Zhang, H. Que, J. Chang, X. Zhang, H. Wei, and T. Zhu, "Safe-SDL: Establishing safety boundaries and control mechanisms for AI-driven self-driving laboratories," arXiv:2602.15061, 2026.
44. L. E. Erdogan, N. Lee, S. Jha, et al., "TinyAgent: Function calling at the edge," arXiv:2409.00608, 2024.
45. J. Bai, A. Aldossary, T. Swanick, et al., "El Agente Grafico: Structured execution graphs for scientific agents," arXiv:2602.17902, 2026.
46. M. Ding, C. Huang, Y. Hu, Y. Li, Z. Lu, X. Yu, D. Zhang, W. Zhai, T. Zhu, Q. Gu, and J. Zeng, "Automating computational chemistry workflows via OpenClaw and domain-specific skills," J. Chem. Theory Comput., vol. 22, no. 12, pp. 5919-5929, 2026. arXiv:2603.25522.

47. J. Dominguez, Y. Amirghofran, and M. Turilli, "FlowGentic," software, version 0.1.0, GitHub repository, 2025. <https://github.com/stride-research/flowgentic> (accessed Oct. 2026).
48. F. Yuan, Z. Ding, Y.-P. Liu, et al., "DPDispatcher: Scalable HPC task scheduling for AI-driven science," *J. Chem. Inf. Model.*, vol. 65, no. 22, pp. 12155-12160, 2025. doi:10.1021/acs.jcim.5c02081.
49. M. Besta, N. Blach, A. Kubicek, et al., "Graph of thoughts: Solving elaborate problems with large language models," arXiv:2308.09687, 2023.
50. W. Kwon, Z. Li, S. Zhuang, et al., "Efficient memory management for large language model serving with PagedAttention," in *Proc. ACM SOSP*, 2023.
51. Y. Zou, A. H. Cheng, A. Aldossary, et al., "El Agente: An autonomous agent for quantum chemistry," *Matter*, vol. 8, p. 102263, 2025.
52. R. Souza, A. Gueroudji, S. DeWitt, D. Rosendo, T. Ghosal, R. Ross, P. Balaprakash, and R. Ferreira da Silva, "PROV-AGENT: Unified provenance for tracking AI agent interactions in agentic workflows," in *Proc. IEEE 21st Int. Conf. e-Science*, 2025. arXiv:2508.02866.
53. S. Sadeghi, K. Mattsson, J. Glasheen, et al., "A self-driving fluidic lab for data-driven synthesis of lead-free perovskite nanocrystals," *Digital Discovery*, vol. 4, no. 7, pp. 1722-1733, 2025. doi:10.1039/d5dd00062a.
54. J. Xu, C. H. J. Moran, A. Ghorai, F. Bateni, J. A. Bennett, N. Mukhin, K. Latif, A. Cahn, P. Jha, F. Delgado Licon, S. Sadeghi, L. Politi, and M. Abolhasani, "Autonomous multi-robot synthesis and optimization of metal halide perovskite nanocrystals," *Nature Communications*, vol. 16, p. 7841, 2025.
55. T. Lookman, Y. Liu, and Z. Gao, "Materials informatics: Emergence to autonomous discovery in the age of AI," *Advanced Materials*, 2026. arXiv:2601.00742.
56. A. G. Kusne and A. McDannald, "Managing autonomous materials labs with multi-agent AI and its implications for the science of science," *Communications Materials*, vol. 7, art. 173, 2026.
57. C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "LLM-Planner: Few-shot grounded planning for embodied agents with large language models," in *Proc. ICCV*, 2023, pp. 2986-2997.
58. Y. Babuji, A. Woodard, Z. Li, et al., "Parsl: Pervasive parallel programming in Python," in *Proc. ACM HPDC*, 2019.
59. D. H. Ahn et al., "Flux: Overcoming scheduling challenges for exascale workflows," *Future Generation Computer Systems*, 2020.
60. N. Somasekharan, R. Pathak, M. Dhanakoti, T. Zhang, L. Yue, A. Zhu, and S. Pan, "AI CFD scientist: Toward open-ended computational fluid dynamics discovery with physics-aware AI agents," arXiv:2605.06607, 2026.
61. S. Kim, S. Moon, R. Tabrizi, et al., "An LLM compiler for parallel function calling," in *Proc. ICML*, 2024, pp. 24370-24391. arXiv:2312.04511.
62. B. Xu et al., "ReWOO: Decoupling reasoning from observations for efficient augmented language models," arXiv:2305.18323, 2023.
63. J. Leeman, Y. Liu, J. Stiles, S. B. Lee, P. Bhatt, L. M. Schoop, and R. G. Palgrave, "Challenges in high-throughput inorganic materials prediction and autonomous synthesis," *PRX Energy*, vol. 3, p. 011002, 2024. doi:10.1103/PRXEnergy.3.011002.
64. N. J. Szymanski et al., "Author Correction: An autonomous laboratory for the accelerated synthesis of inorganic materials," *Nature*, vol. 650, p. E1, 2026. doi:10.1038/s41586-025-09992-y.
65. J. Lala, O. O'Donoghue, A. Shtedritski, S. Cox, S. G. Rodrigues, and A. D. White, "PaperQA: Retrieval-augmented generative agent for scientific research," arXiv:2312.07559, 2023.
66. M. D. Skarlinski, S. Cox, J. M. Laurent, et al., "Language agents achieve superhuman synthesis of scientific knowledge," arXiv:2409.13740, 2024.
67. Y. Chiang, E. Hsieh, C.-H. Chou, and J. Riebesell, "LLaMP: Large language model made powerful for high-fidelity materials knowledge retrieval and distillation," arXiv:2401.17244, 2024.
68. S. Schmidgall and M. Moor, "AgentRxiv: Towards collaborative autonomous research," arXiv:2503.18102, 2025.
69. J. Tang, L. Xia, Z. Li, and C. Huang, "AI-Researcher: Autonomous scientific innovation," arXiv:2505.18705, 2025.
70. E. Li, J. Ren, X. Pan, C. Yan, C. Li, D. Bergemann, and Z. Yang, "Build your personalized research group: A multiagent framework for continual and interactive science automation," arXiv:2510.15624, 2025.

71. Z. Guo, Z. Chen, Z. Chen, Z. Cui, S. Shao, B. Huang, Z. Han, Y. Song, Y. Yuan, C. Zeng, X. Nie, Z. Yu, H. Zhu, J. Liao, M. Zhou, Y. Li, Y. Zhou, and W. Zhang, "Clarus: Coordinating autonomous research agents toward web-scale scientific collaboration," arXiv:2606.30246, 2026.
72. M. H. Prince, H. Chan, A. Vriza, et al., "Opportunities for retrieval and tool augmented large language models in scientific facilities," npj Computational Materials, vol. 10, p. 251, 2024.
73. S. Wang, X. Tian, P. Liu, and Y. Zhao, "Workflow closure is not scientific closure in auto-research systems," arXiv:2605.26200, 2026.
74. M. Rios-Garcia, N. Alampara, C. Gupta, et al., "AI scientists produce results without reasoning scientifically," arXiv:2604.18805, 2026.
75. B. Romera-Paredes, M. Barekatin, A. Novikov, et al., "Mathematical discoveries from program search with large language models," Nature, vol. 625, pp. 468-475, 2024.
76. M. Cemri, M. Z. Pan, S. Yang, L. A. Agrawal, B. Chopra, R. Tiwari, K. Keutzer, A. Parameswaran, D. Klein, K. Ramchandran, M. Zaharia, J. E. Gonzalez, and I. Stoica, "Why do multi-agent LLM systems fail?" in Proc. NeurIPS Datasets and Benchmarks Track, 2025. arXiv:2503.13657.
77. Y. Kim, K. Gu, C. Park, C. Park, S. Schmidgall, A. A. Heydari, Y. Yan, Z. Zhang, Y. Zhuang, Y. Liu, M. Malhotra, P. P. Liang, H. W. Park, Y. Yang, X. Xu, Y. Du, S. Patel, T. Althoff, D. McDuff, and X. Liu, "Towards a science of scaling agent systems," arXiv:2512.08296, 2025.
78. L. E. Erdogan et al., "Plan-and-act: Improving planning of agents for long-horizon tasks," arXiv:2503.09572, 2025.
79. J. Wei, X. Wang, D. Schuurmans, et al., "Chain-of-thought prompting elicits reasoning in large language models," in Proc. NeurIPS, 2022. arXiv:2201.11903.
80. S. Yao, D. Yu, J. Zhao, et al., "Tree of thoughts: Deliberate problem solving with large language models," in Proc. NeurIPS, 2023. arXiv:2305.10601.
81. X. Ning, Z. Lin, Z. Zhou, Z. Wang, H. Yang, and Y. Wang, "Skeleton-of-thought: Prompting LLMs for efficient parallel generation," in Proc. ICLR, 2024.
82. S. Agashe, Y. Fan, A. Reyna, and X. E. Wang, "LLM-coordination: Evaluating and analyzing multi-agent coordination abilities in large language models," arXiv:2310.03903, 2023.
83. R. Gong, Q. Huang, X. Ma, et al., "MindAgent: Emergent gaming interaction," arXiv:2309.09971, 2023.
84. S. Han, Q. Zhang, Y. Yao, W. Jin, Z. Xu, and C. He, "LLM multi-agent systems: Challenges and open problems," arXiv:2402.03578, 2024.
85. G. Zhang, Y. Yue, Z. Li, S. Yun, G. Wan, K. Wang, D. Cheng, J. X. Yu, and T. Chen, "Cut the crap: An economical communication pipeline for LLM-based multi-agent systems," in Proc. ICLR, 2025.
86. Z. Wang, Y. Wang, X. Liu, L. Ding, M. Zhang, J. Liu, and M. Zhang, "AgentDropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration," arXiv:2503.18891, 2025.
87. I. Gim, S. Lee, and L. Zhong, "Asynchronous LLM function calling," arXiv:2412.07017, 2024.
88. A. Amayuelas, J. Yang, S. Agashe, A. Nagarajan, A. Antoniadou, X. E. Wang, and W. Wang, "Self-resource allocation in multi-agent LLM systems," arXiv:2504.02051, 2025.
89. W. Song, J. Yue, and Z. Pang, "Abstral: Automatic design of multi-agent systems through iterative refinement and topology optimization," arXiv:2603.22791, 2026.
90. X. Chen, H. Yu, W. Ni, D. Niyato, R. Zhang, X. Wang, S. Zhang, and S. Xu, "Networking-aware energy efficiency in agentic AI inference: A survey," arXiv:2604.07857, 2026.
91. A. Ehteshami, A. Singh, G. K. Gupta, and S. Kumar, "A survey of agent interoperability protocols: Model Context Protocol (MCP), Agent Communication Protocol (ACP), Agent-to-Agent Protocol (A2A), and Agent Network Protocol (ANP)," arXiv:2505.02279, 2025.
92. W. Shin, R. Souza, D. Rosendo, F. Suter, F. Wang, P. Balaprakash, and R. Ferreira da Silva, "The (r)evolution of scientific workflows in the agentic AI era: Towards autonomous science," arXiv:2509.09915, 2025.
93. Z. S. Siegel, S. Kapoor, N. Nadgir, B. Stroebel, and A. Narayanan, "CORE-Bench: Fostering the credibility of published research through a computational reproducibility agent benchmark," Trans. Mach. Learn. Res., 2024. arXiv:2409.11363.

94. C. Hu, L. Zhang, Y. Lim, A. Wadhvani, A. Peters, and D. Kang, "Repro-Bench: Can agentic AI systems assess the reproducibility of social science research?" in Findings of ACL, 2025, pp. 23616-23626.
95. M. Alizadeh, M. Mosleh, F. Gilardi, A. Kasirzadeh, and J. A. Tucker, "AI coding agents can reproduce social science findings," arXiv:2606.11447, 2026.
96. J. Bragg, M. D'Arcy, N. Balepur, D. Bareket, B. Dalvi, S. Feldman, D. Haddad, J. D. Hwang, P. Jansen, V. Kishore, et al., "AstaBench: Rigorous benchmarking of AI agents with a scientific research suite," arXiv:2510.21652, 2025.
97. G. Starace, O. Jaffe, D. Sherburn, et al., "PaperBench: Evaluating AI's ability to replicate AI research," arXiv:2504.01848, 2025.
98. SchedMD, "sbatch," Slurm Workload Manager documentation. <https://slurm.schedmd.com/sbatch.html> (accessed Oct. 2026).
99. S. Agashe, J. Han, S. Gan, J. Yang, A. Li, and X. E. Wang, "Agent S: An open agentic framework that uses computers like a human," arXiv:2410.08164, 2024.
100. Y. Yang, H. Chai, Y. Song, S. Qi, M. Wen, N. Li, J. Liao, H. Hu, J. Lin, and G. Chang, "A survey of AI agent protocols," arXiv:2504.16736, 2025.
101. V. Ayyagari, "Model Context Protocol for agentic AI: Enabling contextual interoperability across systems," International Journal of Computational and Experimental Science and Engineering, vol. 11, no. 3, pp. 6072-6082, 2025.
102. Google, "Announcing the Agent2Agent Protocol (A2A)," Google Developers Blog, 2025.
103. Anthropic, "Introducing the Model Context Protocol," 2024.
104. L. Zhu, L. Gao, Y. Chen, D. Zhu, and J. Huang, "LAP: An agent-to-instrument protocol for autonomous science," arXiv:2606.03755, 2026.