□

Home LicensingPatentsArticles

Distributed Scientific Computing With Governed Agents

by Nick Clark | Published March 27, 2026 | PDF

Scientific computing distributes computation across clusters, clouds, and grids but does not distribute governance. A job submitted to a cluster carries no intrinsic authority over its execution parameters, data provenance, or result integrity. A cognition-native execution platform enables scientific workloads as governed autonomous agents that carry their own provenance, enforce their own reproducibility constraints, and coordinate across institutional boundaries without central schedulers.

The provenance gap in distributed scientific computing

The reproducibility crisis in science is partly a provenance crisis. When a computation produces a result, the result should carry a complete record of what inputs were used, what code executed, what parameters were set, and what environmental conditions prevailed during execution. In practice, this provenance is tracked through external metadata systems, README files, lab notebooks, and researcher memory, all of which are incomplete, inconsistent, and separable from the results they describe.

High-performance computing clusters execute jobs submitted through batch schedulers. The scheduler manages resource allocation and job sequencing but has no concept of computational provenance, result integrity, or reproducibility constraints. A job that produces incorrect results due to a numerical library version mismatch looks identical to the scheduler as a job that produces correct results. The provenance of the computation is external to the computation itself.

Multi-institutional scientific collaborations face a compounded version of this problem. Data and computation flow across institutional boundaries through ad hoc mechanisms: shared filesystems, data transfer tools, and bilateral agreements. No structural mechanism ensures that a computation performed at one institution can be reproduced at another, or that results transferred between institutions carry verifiable provenance.

Why workflow managers address process but not governance

Scientific workflow managers like Nextflow, Snakemake, and Galaxy define computational pipelines as directed acyclic graphs. These tools solve the process definition problem: they specify what steps execute in what order with what dependencies. They do not solve the governance problem: they do not carry intrinsic authority over execution parameters, data access permissions, or result validation criteria.

A Nextflow pipeline executed on different clusters with different library versions, different numerical precision settings, or different filesystem configurations can produce different results. The pipeline definition is identical. The governance of the execution environment is not captured by the pipeline definition.

How the execution platform addresses this

A cognition-native execution platform represents each scientific computation as a governed agent. The agent carries not just the computation code but also its governance policy: what execution environment is required, what input data versions are acceptable, what numerical precision constraints apply, what provenance must be recorded at each step, and what validation criteria the results must satisfy.

The agent's lineage provides structural provenance. Every state mutation, every input consumed, every intermediate result produced is recorded in the agent's append-only memory with cryptographic chaining. The provenance is not a separate metadata record. It is intrinsic to the computation itself.

When the computation migrates between substrates, whether from a local workstation to a cloud cluster or between institutions, the agent carries its governance with it. The receiving substrate validates the agent's governance requirements against its own capabilities before execution begins. A cluster that cannot provide the required library version rejects the agent rather than executing it with an incorrect version.

What implementation looks like

A research group deploying governed computation agents defines each experiment as an agent with explicit governance over execution requirements, data provenance, and result validation. The agent is submitted to the execution platform rather than to a batch scheduler. The platform validates the agent's governance against available substrates and executes on a substrate that satisfies all constraints.

For multi-institutional collaborations, computation agents cross institutional boundaries with their governance intact. An agent produced at one university and executed at another carries verifiable provenance of its complete execution history. The receiving institution can validate that the agent was executed according to its governance policy without trusting the sending institution's infrastructure.

For reproducibility, any party with access to the agent can inspect its complete lineage: every input, every parameter, every intermediate state, and every environmental condition recorded during execution. Reproducing the computation means instantiating the agent on a substrate that satisfies its governance requirements and letting it execute according to its own policy.

For funding agencies and journals requiring reproducibility, the agent's lineage provides a machine-readable, cryptographically verifiable record of computational provenance that goes beyond what any external metadata system can provide.

Execution Platform All 21 steps →

The complete runtime for governed, persistent agents.

Patent
US 19/230,933 · filed
Primary Technical Disclosure
○ A Cognition-Native Execution Platform for Distributed, Stateful, and Governable Agents
Secondary Technical
○ Six-Field Canonical Agent Schema: Structural Definition of Autonomous Semantic Agents○ Semantic Nest Instantiation: Dynamic Execution Environments From Agent Density and Entropy○ Trust Zone Overlay Governance: Logical Policy Domains Independent of Network Topology○ Scoped Quorum Mutation Validation: Independent Validators With Meta-Policy Escalation○ Meta-Policy Override Resolution: Higher-Level Governance for Local Quorum Decisions○ Semantic Router: Schema-Aware Propagation Replacing Address-Based Forwarding○ Dynamic Agent Hash Derivation: Deterministic Identity From Memory and Mutation History○ Dynamic Device Hash Derivation: Substrate Identity From Device-Local Entropy○ Content Anchor Hash Derivation: Perceptual Identity for Non-Executing Digital Content○ DAH-DDH Slope Entanglement: Binding Agent Identity to Host Device Lineage○ Trust Slope Validation Across Zone Migration: Continuity Verification With Quarantine○ Pseudonymous Propagation: Recognition by Slope Rather Than Global Identifier○ Alias Slope-Band Indexing: Symbolic Resolution Through Trust-Slope Pathfinding○ Fallback Rehydration: Recovering Partial Agents Through Contextual Policy Inference○ Structural Validator With Fallback Routing: Schema Verification Before Execution○ Execution Graph Manager: Structured Lineage of Agent Reasoning and Transformation○ Full and Partial Agent Interoperability: Cross-Boundary Semantic Exchange Under Policy○ Cross-Topology Substrate Deployment: Identical Agent Structure Across All Substrates
Applications (General)

○ Multi-Cloud Agent Orchestration Without Centralized Schedulers○ Autonomous Fleet Coordination Through Self-Governing Agents○ Enterprise Workflow Without Orchestration Servers○ Smart Contract Execution Without Blockchain Latency● Distributed Scientific Computing With Governed Agents○ Supply Chain Autonomous Agents○ Energy Grid Management Through Autonomous Agents○ Disaster Response Coordination Without Central Command
Applications (Specific)
○ Kubernetes Orchestrates Containers. It Does Not Know What They Are Doing.○ Temporal Solved Durable Workflows. The Workflows Have No Semantic Identity.○ Apache Airflow Orchestrates DAGs. The Tasks Inside Them Are Ungoverned.○ Prefect Made Data Workflows Pythonic. The Execution Model Is Still Task Scheduling.○ AWS Step Functions Made Serverless Orchestration Visual. The Steps Have No Semantic State.○ Azure Durable Functions Made Stateful Serverless Possible. The State Has No Governance.○ Nomad Schedules Any Workload. It Does Not Know What Those Workloads Are.○ Docker Swarm Simplified Container Orchestration. The Containers Are Still Opaque.○ Apache Mesos Managed Datacenter Resources. The Resources Had No Semantic Governance.○ Argo Workflows Orchestrates Kubernetes-Native Pipelines. The Pipeline Steps Have No Governance.○ Dagster Made Data Pipelines Software-Defined. The Pipeline Has No Governance Substrate.○ Luigi Defined Task Dependencies for Data Pipelines. The Tasks Execute Without Governance.○ Camunda Orchestrates Business Processes. The Process Engine Has No Semantic Agent Governance.○ Zeebe Scaled Workflow Orchestration Horizontally. Governance Did Not Scale With It.
Execution Platform overview →
AQ
deterministic
autonomy

Legal

Subject to one or more pending U.S. and international patent applications, see Patents for the current list and status. No license, express or implied, is granted. Any use requires a separate written agreement—see Licensing. Patent applications referenced on this site are pending. Claim scope, if any, is subject to examination and may issue in altered form or not at all. See Legal for terms and conditions.

Adaptive Query™ is a trademark of Nicholas Clark. U.S. federal registration is pending. federal registration. AQ™ , AQ Inside™ , Adaptive Index™ , Adaptive Network™ , Semantic Agent™ , @AQ™ , AQID™ , and Adaptive Coin™ are used as trademarks in connection with the Adaptive Query platform and brand. Other names may be trademarks of their respective owners.

Last updated: 2026-03-03

-
- Inventive Steps
- Licensing
- Patents
- Articles
- Legal
- Opportunities
- Sitemap

-
- nick@qu3ry.net
- 72 28 14 36 01

Invented by Nick Clark | Founding Investors: Devin Wilkie