

White Paper

A Federated and Distributed Scientific Compute Mesh
for Bioinformatics and Computational Biology

Centralized coordination. Distributed scientific computing.

Technical Architecture Baseline – Version 0.1, Revision A

September 9, 2026

Coordinating institution: Universidade do Estado de Santa Catarina – UDESC, Brazil

<https://github.com/BioinformaticaUDESC/LABIO.NETWORK>

Executive Summary

LABIO.NETWORK is an academic distributed-computing infrastructure designed to aggregate heterogeneous scientific resources – institutional servers, laboratory workstations, community computers and compatible mobile devices – into a unified execution layer for bioinformatics and computational biology. The initial network is coordinated by UDESC while computation is intentionally distributed across compatible participating nodes.

Modern biological research is increasingly constrained not by the availability of analytical methods, but by access to suitable compute, reproducible software environments, trusted data handling and the ability to scale computational experiments. Genome assembly, metagenomic classification, molecular docking, molecular dynamics, sequence search and scientific machine learning can require resources that are unevenly distributed across laboratories and institutions. At the same time, substantial CPU, memory and accelerator capacity remains intermittently underused in universities, research workstations, servers and personal devices.

LABIO.NETWORK addresses this mismatch through a federated architecture that separates *control* from *execution*. A central Control Plane handles identity, policy, queueing, application metadata, scheduling, accounting, validation and provenance. Scientific work is executed only by nodes that satisfy explicit compatibility, security and data-governance requirements. Nodes request assignments through authenticated outbound connections, which avoids requiring personal or institutional machines to expose public inbound services.

LABIO Credits are conceived as an internal scientific resource-accounting mechanism representing validated useful computation. They support reciprocal access to shared compute while distinguishing productive scientific work from simple device uptime.

The problem we address

The infrastructure problem has five interacting dimensions:

- **access asymmetry:** many researchers lack continuous access to HPC, high-memory servers or accelerators;
- **resource fragmentation:** useful capacity is distributed across institutions and device classes without a common scheduling layer;
- **heterogeneity:** x86-64, ARM64, GPUs, operating systems and runtimes differ substantially in capability;
- **scientific trust:** results from remote or volunteer nodes require integrity checks, provenance and application-specific validation;
- **data governance:** biological datasets range from fully public sequences to restricted or regulated human data and therefore cannot share one execution policy.

The LABIO vision

The long-term objective is a federated scientific compute layer where institutions retain governance over their own resources while interoperating through common contracts. Community participation expands available capacity for suitable non-sensitive workloads, and

mobile devices contribute only bounded kernels compatible with their architecture, battery, network and thermal constraints.

What is being built

The public architecture defines five major user-facing components:

1. **LABIO Workspace:** research interface for project preparation, job submission, status tracking, results, devices and credits;
2. **LABIO Mobile:** Android ARM64 edge node for explicitly authorized, bounded scientific workloads;
3. **LABIO Desktop:** workstation node for Windows, Linux and macOS systems;
4. **LABIO Server:** Linux-first node for dedicated, GPU and institutional compute;
5. **LABIO AI:** evidence-grounded scientific assistance for method selection, reproducible workflow preparation and interpretation support.

Public maturity baseline at White Paper version 0.1.

Component	Public status
LABIO Workspace	Operational prototype
LABIO Mobile	Closed testing
Control Plane and job API	Operational prototype
LABIO Desktop	Architecture and prototype planning
LABIO Server for Linux	Architecture and prototype planning
LABIO AI	Research and development
Three NVIDIA A100 accelerators	Expected; pending installation and validation
NVIDIA H100 accelerators	Proposed for a future funding request
100,000 smartphones / 100 servers	Analytical capacity scenarios, not deployed inventory

Problem and Scientific Motivation

Computational biology is increasingly infrastructure-limited

Biological data generation has become progressively more accessible, while analysis frequently remains dependent on large-memory hosts, accelerators, parallel execution or long-running workflows. This creates a structural gap between laboratories able to maintain dedicated compute clusters and those relying on individual workstations or temporary cloud allocations. The result is not merely slower computation: limited compute can constrain the number of replicates, parameter combinations, candidate molecules, reference databases or alternative hypotheses that can be evaluated.

Distributed scientific computing offers a complementary model. Volunteer-computing platforms demonstrated that geographically dispersed devices can deliver meaningful aggregate capacity when workloads are partitionable and failure is expected rather than exceptional [16, 17]. LABIO extends this idea toward bioinformatics-specific scheduling, scientific validation, provenance, institutional federation and explicit data classification.

Why a bioinformatics-specific network is necessary

A generic batch scheduler is insufficient because biological workloads have distinct requirements. A docking campaign may consist of millions of independent ligand evaluations and therefore scale efficiently across heterogeneous nodes. Molecular dynamics, in contrast, may depend on sustained GPU performance, memory bandwidth, interconnect behavior and checkpoint continuity. Metagenomics can combine high-throughput preprocessing with large reference databases and memory-intensive assembly. Sensitive human genomics introduces governance constraints that override pure performance considerations.

Accordingly, LABIO treats **application identity**, **data class**, **runtime compatibility**, **resource capability** and **validation profile** as first-class scheduling inputs.

Design objectives

The network is designed around the following technical objectives:

- maximize useful scientific throughput rather than generic theoretical FLOPS;
- preserve reproducibility through versioned applications, input hashes, parameters and provenance records;
- tolerate node churn, disconnects and heterogeneous performance;
- prevent arbitrary remote code execution on contributor devices;
- retain institutional control over quotas, priorities and data policies;
- provide transparent resource accounting linked to validated work;
- support gradual evolution from prototype to federated production infrastructure.

Scientific principle. More available compute does not replace experimental design, statistical validity or biological interpretation. It expands the set of scientifically justified computations that can be evaluated within practical time and resource constraints.

Foundational Technologies and Prior Art

Volunteer and distributed computing

Volunteer computing provides the core precedent for scheduling independent tasks on contributed devices under intermittent availability. BOINC formalized a platform model in which centralized project services coordinate distributed clients, while later work investigated globally optimized scheduling and heterogeneous host performance [16, 17]. LABIO adopts the principle of pull-based distributed execution but adds domain-specific controls for scientific software, biological data classification, provenance and institutional federation.

Identity and Zero Trust

The security architecture follows the Zero Trust principle that network location, device ownership or institutional membership does not automatically confer trust [1]. User sessions and device identities are separate security domains. Federated user authentication is aligned

with OpenID Connect [2], while native application authorization follows the external user-agent pattern described by OAuth 2.0 for Native Apps [6].

Portable scientific runtimes

Linux server and institutional workloads are designed around content-addressed, versioned execution artifacts compatible with the OCI Image Specification [4]. Supply-chain assurance is informed by SLSA provenance concepts [5]. For workflow portability, systems such as Nextflow demonstrate how process dependencies and computational environments can be described independently from the underlying execution site [9].

Scientific provenance and FAIR principles

Scientific results are useful only when their computational history is reconstructable. LABIO provenance is conceptually aligned with W3C PROV-O entities, activities and agents [3]. Metadata and data-management decisions are further informed by FAIR principles – findability, accessibility, interoperability and reusability [8].

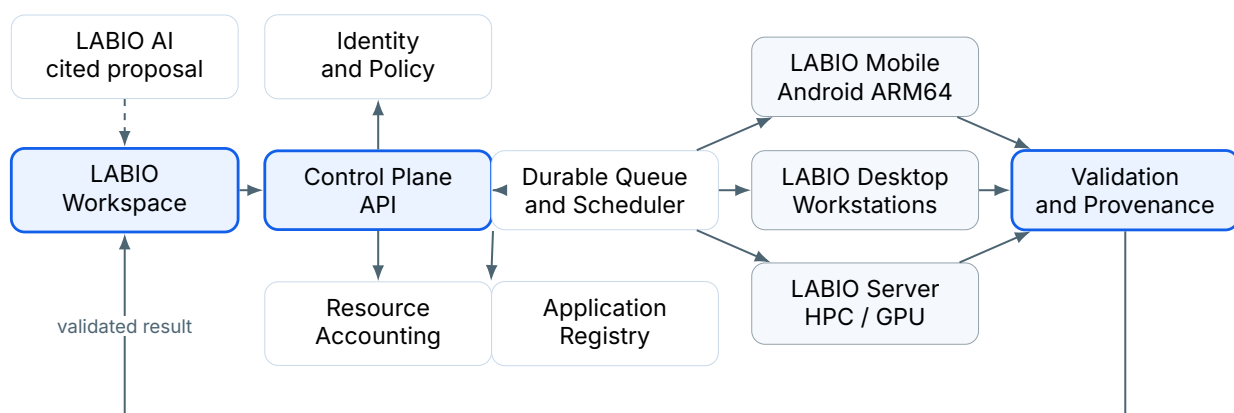
Observability and institutional scheduling

OpenTelemetry and Prometheus provide reference models for vendor-neutral traces, metrics and time-series observability [11, 12]. Institutional queues may draw on Kubernetes-native admission, quota and fair-sharing concepts represented by Kueue [10] without requiring every LABIO deployment to use the same internal scheduler.

LABIO.NETWORK System Architecture

Control plane and execution plane

LABIO separates global coordination from local execution. The Control Plane maintains the authoritative logical state of accounts, devices, applications, jobs, assignments, results and accounting entries. Compute nodes never become authoritative for global state; they receive time-bounded assignments and return result manifests with execution metadata.



Reference architecture: centralized coordination with distributed scientific execution.

Canonical entities

Core logical entities and their stable identifiers.

Entity	Stable identity	Selected relationships
Account	Opaque account ID	Projects, devices, accounting entries
Device	Random device ID	Owner, capabilities, policy, heartbeats
Application	Name + semantic version	Runtime artifact, schemas, validation profile
Job	Immutable job ID	Requester, application, input manifest, state
Assignment	Lease ID + attempt	Job, device, expiry, resource limits
Result	Result ID + content hashes	Job, assignment, validation state
Accounting entry	Immutable accounting event ID	Account, job/contribution event, amount
Provenance record	Provenance ID	Inputs, application, parameters, execution, outputs

Pull-based node communication

Personal and institutional compute nodes request work through authenticated outbound connections. This topology is compatible with NAT, CGNAT, residential routers, mobile networks and restrictive institutional firewalls because no unsolicited inbound connection to the worker is required. Short-lived scoped credentials reduce the impact of leaked enrollment material.

Product boundaries

Each LABIO product has a bounded role. Workspace is the researcher-facing web environment; Mobile, Desktop and Server are execution classes; LABIO AI is an advisory layer that may propose scientifically grounded actions but should not bypass typed application schemas or explicit authorization for actions that spend credits, access restricted data or launch computation.

Capability-Aware Scheduling

Hard eligibility constraints

For a job j and node n , eligibility is determined before ranking. Let the binary predicates represent architecture (A), operating system (O), runtime (R), memory (M), accelerator (G), application allow-list (L), data policy (D) and participant policy (P). Then

$$E(n, j) = A_{nj}O_{nj}R_{nj}M_{nj}G_{nj}L_{nj}D_{nj}P_{nj},$$

with $E(n, j) = 1$ required for assignment. A node that is fast but incompatible or unauthorized is never a candidate.

Soft ranking

Among eligible nodes, a normalized scheduling score may be expressed as

$$S(n, j) = w_c C_{nj} + w_r R_n + w_a A_n + w_p P_{nj} + w_q Q_n + w_l L_{nj} - w_e E_{nj},$$

where C_{nj} is compatibility quality, R_n reliability, A_n current availability, P_{nj} measured scientific performance, Q_n network quality, L_{nj} data locality and E_{nj} estimated resource or energy penalty. The weights w are policy parameters, not scientific constants; changes require versioning and evaluation.

The selected node is therefore

$$n^*(j) = \arg \max_{n \in \mathcal{N}: E(n,j)=1} S(n, j).$$

Reliability as a dynamic signal

A node's reliability estimate can be updated using an exponentially weighted moving average (EWMA):

$$R_n^{(t)} = \lambda R_n^{(t-1)} + (1 - \lambda)x_t,$$

where $x_t \in [0, 1]$ represents the quality of the latest attempt and λ controls memory of historical performance. Application-specific reliability may be maintained separately because a node can be reliable for short CPU tasks while unsuitable for long GPU workloads.

Institutional fairness

Federated resource pools may apply local quotas before global sharing. If institution k exposes capacity $C_k(t)$ and reserves $Q_k(t)$ for local projects, the externally schedulable capacity is bounded by

$$C_k^{\text{shared}}(t) \leq \max\{0, C_k(t) - Q_k(t)\}.$$

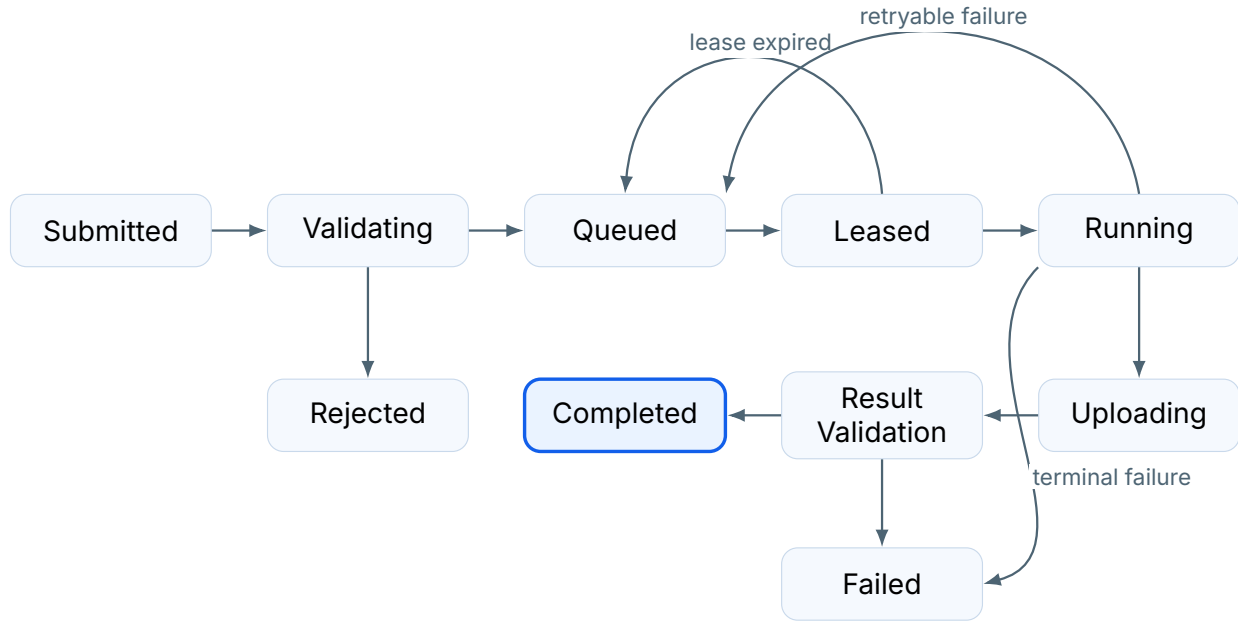
This preserves institutional autonomy while making idle capacity available to authorized network projects.

The scheduler should optimize for **measured application throughput**, not marketing specifications. A phone CPU core, a server CPU core and a GPU TFLOP are not interchangeable units of scientific capacity.

Job Lifecycle, Leases and Fault Tolerance

State machine

Assignments are time-bounded leases with an attempt number. This prevents a late result from an expired worker from settling the same job twice and makes disconnects normal recoverable events.



Conceptual LABIO job lifecycle with explicit validation, requeue and terminal states.

Failure probability and redundancy

If a task is replicated to r statistically independent nodes and the probability that one attempt finishes successfully within the lease is p , the probability that at least one attempt succeeds is

$$P_{\text{success}}(r) = 1 - (1 - p)^r.$$

Replication improves resilience but consumes additional compute. Therefore redundancy should be application- and trust-dependent rather than universal.

For majority validation over r odd independent replicas with per-replica probability p of a correct result, the probability that a strict majority is correct is

$$P_{\text{maj}} = \sum_{k=(r+1)/2}^r \binom{r}{k} p^k (1 - p)^{r-k}.$$

This idealized model is useful for reasoning but must not be interpreted as a substitute for application-specific validators or correlated-failure analysis.

Checkpoint and requeue

Long-running workloads should use application-supported checkpointing where available. Short independent tasks can simply be requeued. The scheduler can select the recovery strategy from a validation profile containing maximum retry count, checkpoint semantics, deadline and estimated recomputation cost.

Distributed Compute Capacity Model

Peak versus effective capacity

Theoretical aggregate performance is

$$P_{\text{peak}} = \sum_{i=1}^N P_i,$$

where P_i is the theoretical performance of node i . This quantity is insufficient for a heterogeneous scientific network. LABIO therefore defines a simplified effective capacity model:

$$P_{\text{effective}} = \sum_{i=1}^N P_i A_i \eta_i C_i,$$

where A_i is availability, η_i is measured execution efficiency and C_i is workload compatibility. Depending on the application, P_i may be replaced entirely by an empirical benchmark such as reads/s, dockings/hour or ns/day.

Task-throughput model

For an independently partitionable task with benchmarked execution time t_i on node i , estimated daily throughput is

$$T_{\text{day}} = \sum_{i=1}^N \frac{86,400 A_i \eta_i}{t_i}.$$

For virtual screening, this becomes an estimate of validated docking evaluations per day, provided t_i is measured using a fixed receptor, ligand set, software version and parameterization.

Smartphone capacity scenario

For N registered smartphones, each contributing c logical cores with measured eligibility/availability factor A ,

$$C_{\text{mobile}} = NcA.$$

The public planning scenario of 100,000 registered smartphones with two contributed logical cores and $A = 0.25$ yields 50,000 concurrently effective logical cores *in the accounting model*. This is explicitly not equivalent to 50,000 server cores because sustained performance, instruction sets, memory, thermal constraints and runtime efficiency differ.

Hybrid capacity

The strongest architecture combines device classes rather than forcing all workloads onto one class:

- smartphones: short, independent, non-sensitive portable kernels;
- desktops/workstations: medium-duration jobs and opportunistic CPU/GPU acceleration;
- Linux servers/HPC: high-memory, long-running, accelerator-dependent and restricted workloads;
- dedicated accelerator pools: compatible AI, structural biology and simulation workloads after installation and validation.



Analytical capacity scenarios; not deployed inventory.

Dimension	100,000 smartphones	100 Linux servers
Availability	Intermittent; battery, heat, user and network dependent	More predictable; maintenance-window dependent
Workload	Many short independent tasks	Long, memory-heavy or accelerator-dependent jobs
Runtime	Approved native/portable kernels	OCI containers and optimized native tools
Failure model	High churn; frequent requeue	Lower churn; planned maintenance
Data policy	Public/explicitly eligible	Institutional and policy-controlled
Validation	Stronger redundancy for untrusted nodes	Trust tier plus application validation

Scientific Validation and Provenance

Result validation

A distributed node is not trusted merely because it completed a process without error. Validation is part of the job contract. Depending on the application, LABIO may use deterministic output checks, checksums, expected-value ranges, cross-node replication, reference test cases or post-processing consistency rules.

For an application a , define a validation function

$$V_a(I, \theta, O, M) \rightarrow \{\text{accept, reject, review}\},$$

where I denotes input hashes, θ the execution parameters, O output artifacts and M execution metadata. The validator is versioned together with the application profile.

Scientific provenance record

A reproducible job should record at minimum:

- immutable job and assignment identifiers;
- application name, semantic version and artifact digest;
- hashes of input objects;
- complete execution parameters;
- runtime class and processor architecture;
- sanitized node class and benchmark profile;
- timestamps for submission, assignment, start and completion;
- output hashes and validation state.

Conceptual LABIO provenance manifest.

```
job_id: LAB-XXXXXXX
application:
  name: autodock-vina
  version: 1.x
```

```
artifact_digest: sha256:...
input:
  receptor_sha256: ...
  ligand_sha256: ...
parameters:
  exhaustiveness: 8
execution:
  node_class: institutional | community | mobile
  architecture: amd64 | arm64
  runtime: OCI | native | portable
result:
  output_sha256: ...
  validated: true
```

These records are conceptually aligned with W3C PROV-O [3]. Public statistics must use aggregate data and must not expose the direct mapping between a public account and a physical device.

Reproducibility across heterogeneous devices

Bitwise identity is not always scientifically required or even achievable across CPU/GPU implementations. The validation profile therefore distinguishes *bitwise deterministic*, *numerically tolerant* and *scientifically equivalent* outputs. Any tolerance must be application-specific, justified and recorded.

LABIO Credits and Scientific Resource Accounting

LABIO Credits are an internal accounting unit for validated useful scientific contribution and consumption of distributed resources.

Earning credits

A conceptual credit function for a completed job can be written as

$$C_{\text{earned}} = W \cdot R \cdot V \cdot Q,$$

where W is normalized computational work, R a resource factor, V the validation factor and Q a quality/reliability factor. A node that remains online without producing validated work does not receive the same credit as a node completing useful scientific computation.

A practical normalization may be derived from benchmark-normalized resource seconds:

$$W = \alpha_c t_c b_c + \alpha_g t_g b_g + \alpha_m tm + \alpha_s s + \alpha_n d,$$

where CPU time t_c , GPU time t_g , memory-time tm , storage s and data transfer d are weighted by transparent coefficients α , while b_c and b_g capture benchmark class. The coefficients require empirical calibration and governance review.

Using credits

The cost of a job may be decomposed as

$$C_{\text{job}} = C_{\text{CPU}} + C_{\text{GPU}} + C_{\text{memory}} + C_{\text{storage}} + C_{\text{transfer}} + C_{\text{priority}}.$$

Credits allow resource reciprocity: contributors can receive access to capacity in return for validated work. Institutional pools may choose separate quota or accounting rules.

Accounting properties

The resource-accounting log should be append-only and idempotent. Every mutation includes a reason, linked job/contribution event, immutable accounting event ID and reconciliation state. A duplicate result upload must never issue duplicate credits.

Security and Privacy Architecture

Zero Trust boundaries

LABIO assumes that a distributed environment may contain compromised accounts, malicious workers, tampered applications or fraudulent outputs. The security model therefore treats the following domains independently:

USER != DEVICE != APPLICATION != DATA != HOST FILESYSTEM

No authorization in one domain automatically grants unrestricted access to another. This direction is consistent with Zero Trust architecture principles [1].

Application security and supply chain

Researchers should not directly push arbitrary executables to volunteer nodes. The intended lifecycle is source review, dependency validation, compatibility testing, scientific benchmarking, artifact signing and registration. Nodes execute only approved artifacts whose digest matches the assignment. OCI content-addressable manifests [4] and SLSA provenance concepts [5] provide relevant external standards.

Workload isolation

A job should see only its input objects, temporary working directory and explicitly assigned CPU, memory, accelerator and network permissions. It must not gain access to personal documents, browser profiles, SSH keys, unrelated datasets or other jobs. Server nodes should prefer rootless containerization or an equivalent isolation boundary; mobile and desktop agents require platform-specific sandboxing and explicit resource limits.

Data classification

Proposed scientific data classes and scheduling policy.

Class	Description	Execution policy
T0	Public scientific data	Eligible for distributed execution
T1	Internal non-sensitive research data	Policy-controlled execution
T2	Restricted research data	Trusted institutional nodes only
T3	Sensitive or regulated human data	Controlled infrastructure only

Sensitive genetic, clinical or directly identifiable human data should not be distributed to arbitrary community devices. Data classification is a hard scheduler constraint, not a ranking preference.

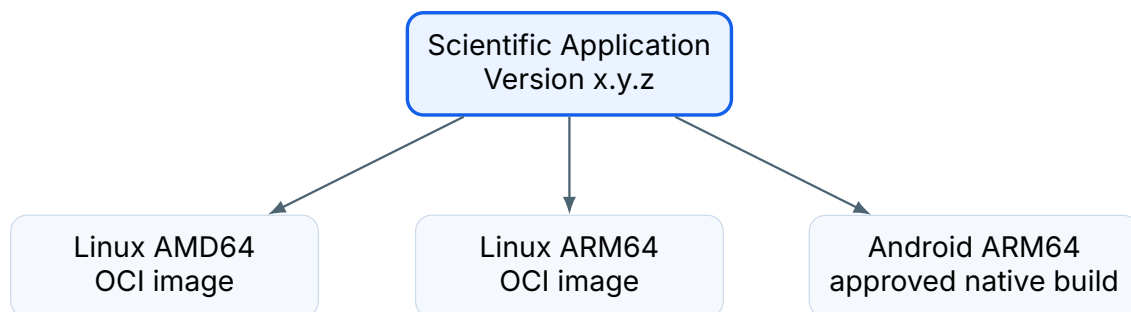
Threat model

The architecture explicitly considers Sybil enrollment, replay of assignments, result forgery, credential theft, denial of service, malicious workloads, data exfiltration and software supply-chain compromise. Mitigations include authenticated enrollment, short-lived credentials, nonce/attempt binding, digest verification, allow-listed runtimes, rate limiting, result validation, reputation, audit events and restricted-data placement.

Scientific Application Ecosystem

Application Registry

The LABIO Application Registry represents each supported scientific application as a versioned contract with input schema, output schema, runtime artifacts, capability requirements, validation profile and data-class restrictions. Different platform builds can implement the same scientific application identity.



A single scientific application can expose architecture-specific execution artifacts.

Representative workload classes

Target scientific domains and representative workloads.

Scientific area	Representative workloads
Structural bioinformatics	Molecular docking, scoring, structural analysis
Molecular dynamics	GROMACS and related simulation workflows [19]
Genomics	Alignment, assembly, annotation and variant analysis
Metagenomics	Classification, assembly and functional analysis
Phylogenetics	Tree inference, evolutionary models and sequence analysis
Sequence analysis	BLAST-like searches, k-mer operations, transformations [20]
Scientific AI	Inference and selected machine-learning workloads
Scientific workflows	Multi-stage pipelines coordinated by workflow engines [9]

Virtual screening as a naturally distributed workload

AutoDock Vina is representative of docking tools that can support independent ligand evaluations [18]. If a campaign contains N ligand-target tasks and D equivalent continuously available devices, the idealized completion time is

$$T_{\text{ideal}} \approx \frac{N\bar{t}}{D},$$

where $\bar{t}$ is average task time. Real execution includes scheduler, transfer, validation and heterogeneity overhead. This makes docking an appropriate benchmark family for assessing LABIO's network effect.

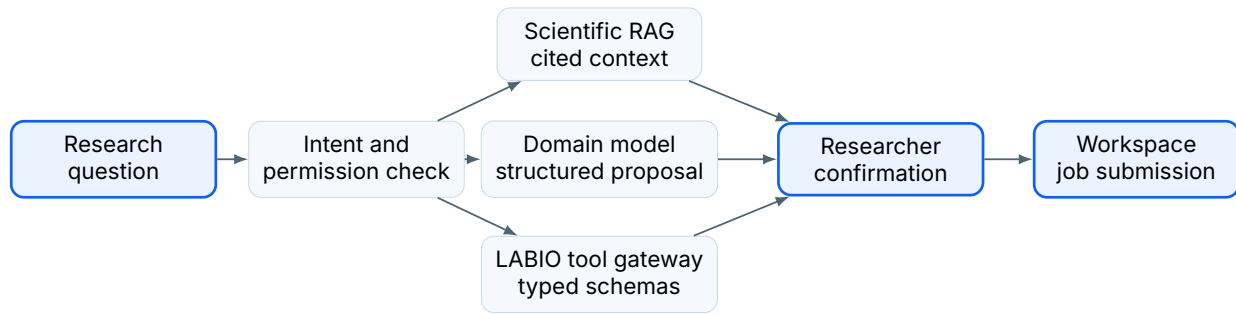
Workflow orchestration

Bioinformatics analyses commonly contain dependent stages. A workflow engine can manage process dependencies while LABIO determines where each compatible task executes. The separation avoids embedding LABIO-specific scheduling logic into every scientific pipeline.

LABIO AI: Evidence-Grounded Assistance

Role and boundaries

LABIO AI is planned as a research assistant that helps users understand methods, prepare reproducible analyses, select approved tools and interpret outputs while preserving evidence, uncertainty and user control. Retrieval-augmented generation provides a conceptual basis for conditioning model responses on attributable source collections [7].



LABIO AI separates evidence retrieval, model generation, typed tool access and user authorization.

Scientific safeguards

The retrieval layer must respect project and institutional access boundaries. Generated text is not treated as scientific evidence by default. The tool gateway validates parameters against approved schemas and does not execute arbitrary commands generated by the model. Actions that spend credits, access restricted data or launch computation require explicit authorization and a visible summary of parameters, cost and destination.

Evaluation should include citation correctness, source coverage, task accuracy, parameter validity, hallucination behavior, prompt-injection resistance, data-boundary compliance and reproducibility across Portuguese, English and Spanish.

Implementation Direction

The public architecture deliberately distinguishes current prototypes from planned production components. Final technology selections remain subject to benchmark, security and operational validation.

Current implementation direction, not a frozen production stack.

Layer	Technology direction
Frontend / Workspace	React / Next.js
Control API	Go and/or FastAPI-based services
Metadata	PostgreSQL
Institutional orchestration	Kubernetes
Batch queueing / quota concepts	Kueue [10]
Scientific workflows	Nextflow [9]
Containers	OCI-compatible runtimes [4]
Community computing	LABIO Compute Agent / volunteer-computing concepts
Mobile computing	LABIO Mobile Agent, Android ARM64
Observability	OpenTelemetry and Prometheus-compatible telemetry [11, 12]
Object storage	S3-compatible infrastructure

LABIO Mobile

The Mobile node is in closed Android testing. It registers an opaque device identifier, reports sanitized capabilities and current availability, requests work through outbound authenticated

connections, respects charging/network/thermal/user policies and receives credits only for validated useful computation. Sustained execution must remain user-visible and consistent with Android foreground-service restrictions [15].

LABIO Server

The Server node is Linux-first and targets dedicated servers, GPU systems, laboratory clusters and institutional infrastructure. It is designed for unattended operation, multiple execution slots, explicit CPU/memory/GPU quotas, maintenance windows and institution-first policies. Accelerator capacity is published as operational only after delivery, installation, burn-in, security review, driver/runtime validation and scientific benchmarking.

Accelerator roadmap

Three NVIDIA A100 accelerators are documented as expected rather than operational; NVIDIA H100 accelerators are described only as a proposed future acquisition direction. Exact throughput must not be inferred from vendor peak specifications before model, memory, form factor, interconnect and application benchmarks are known [13, 14].

Scientific Benchmarking Methodology

Workload-oriented metrics

LABIO should report scientific productivity in application-level units. Examples include:

- validated dockings/hour and ligands/day;
- GROMACS ns/day;
- reads processed/s and sequences aligned/s;
- assemblies/day or time-to-assembly;
- validated inference throughput;
- workflow completion time and failure-recovery overhead.

Every benchmark should report software version, dataset, parameters, node class, sample size, variability and relevant energy/thermal policy. This avoids misleading comparisons based on peak FLOPS alone.

Parallel efficiency

For a workload with single-node time T_1 and distributed time T_D using D comparable workers, speedup and parallel efficiency are

$$S_D = \frac{T_1}{T_D}, \quad \eta_D = \frac{S_D}{D}.$$

Heterogeneity and communication overhead make $\eta_D < 1$ in practical systems. For independent task bags, the key bottlenecks are often assignment latency, artifact transfer, result upload, node churn and long-tail task times.

Heterogeneity index

For benchmark throughput values x_i within a node class, the coefficient of variation

$$CV = \frac{\sigma_x}{\mu_x}$$

provides a simple measure of dispersion. High CV suggests that a coarse hardware label is insufficient and that the scheduler should retain finer benchmark profiles.

Energy-aware reporting

Where power telemetry is available and privacy policy permits, useful-work energy intensity can be expressed as

$$I_E = \frac{\int_0^T P(t) dt}{N_{\text{validated}}},$$

where $N_{\text{validated}}$ is the number of accepted scientific tasks. The metric is workload-specific and must not be compared across unrelated algorithms without context.

Governance, Federation and Institutional Autonomy

LABIO.NETWORK is designed as academic and scientific infrastructure. UDESC coordinates the initial network architecture and Control Plane, while the target federation model allows participating institutions to retain authority over their own machines, quotas, maintenance windows, trusted data classes and local priorities.

Federated resource pools

An institution may expose only an approved fraction of idle capacity. Resource sharing can be project-specific, time-bounded or reciprocal. Local policies are evaluated before global scheduling; LABIO therefore does not require an institution to surrender operational control in order to participate.

Governance principles

Federation should preserve:

- scientific governance and reproducibility;
- transparent resource accounting;
- institutional autonomy;
- security and auditability;
- explicit data-governance boundaries;
- documented scheduler and credit policies;
- versioned technical contracts and migration paths.

Public reporting

Public network statistics should be derived aggregates. They may include broad node counts, capacity classes, benchmark methodology and validated workload throughput, but should omit serial numbers, private hostnames, management addresses, credentials, exact device locations and mappings from public users to physical hardware.

Roadmap and Future Research

Development roadmap expressed as technical stages rather than guaranteed dates.

Stage	Primary milestones
Architecture and proto-type	Control Plane contracts; device/job models; Application Registry; scheduler prototype; scientific benchmark framework
Distributed execution	Linux and Windows agents; validation; resource accounting; LABIO Credits prototype; institutional runtime adapters
Mobile computing	Android ARM64 agent; battery/thermal-aware scheduling; mobile benchmark suite; approved ARM64 application registry
Federation	Multi-institution resource pools; quotas; high-availability Control Plane; federated execution policies
Scientific platform	Jupyter-oriented workspace; application catalog; workflows; LABIO AI; distributed genomics and structural-biology workflows

Research questions

Important research questions remain open:

- How should useful scientific work be normalized across CPU, GPU and mobile architectures?
- How can fraudulent or subtly incorrect computation be detected efficiently without universal replication?
- What redundancy level minimizes expected cost while achieving application-specific validation confidence?
- How should the scheduler balance fairness, performance, energy, locality and institutional policy?
- Which bioinformatics kernels achieve meaningful efficiency on Android ARM64 without unacceptable battery or thermal impact?
- How should sensitive-data policies interoperate across institutions with different governance requirements?
- How does network scale affect real virtual-screening, sequence-analysis and workflow throughput under node churn?
- Can benchmark-derived scheduling outperform hardware-description-only scheduling across heterogeneous node classes?

Open engineering decisions

Before production-scale operation, the project must freeze and review job/result schema versioning, node transport and heartbeat protocols, object-storage retention, trusted runtime signing, scheduler fairness, credit normalization, institutional federation rules and measurable service objectives including recovery point and recovery time targets.

Conclusion

LABIO.NETWORK proposes a scientific-computing architecture in which heterogeneous resources can be converted into a shared bioinformatics execution layer without collapsing identity, device trust, application trust, data governance and resource accounting into a single undifferentiated system. Central coordination provides a consistent control and audit plane; distributed execution makes it possible to use institutional, community and mobile resources according to their actual capability and policy.

The design is deliberately conservative about capacity claims. Expected accelerators are not described as installed, and scenarios involving 100,000 smartphones or 100 servers are treated as analytical models rather than deployed inventory. Scientific throughput must be established by reproducible application benchmarks.

The central technical contribution of LABIO is therefore not simply aggregation of processors. It is the combination of capability-aware scheduling, approved versioned scientific runtimes, explicit data classification, result validation, provenance, fault tolerance and reciprocal resource accounting in a domain-specific network for computational biology.

As the architecture matures, its success should be evaluated by reproducibility, useful scientific throughput, security, accessibility and the quality of research enabled – not by node counts alone.

Scope of Version 0.1

This document is a public technical architecture baseline, not a certification report or a claim of production-scale deployment. Components explicitly marked as prototype, closed testing, planned, expected or proposed retain those maturity labels. Numerical capacity examples are planning models until representative hardware, datasets, software versions and benchmark protocols are measured.

Document governance and recommended citation

The public GitHub repository is the technical source of truth for evolving component contracts, security boundaries and maturity labels. Material architecture changes should result in a new white-paper version so that scientific and institutional users can distinguish historical design assumptions from the current implementation.

Recommended citation: LABIO.NETWORK Team. *LABIO.NETWORK: A Federated and Distributed Scientific Compute Mesh for Bioinformatics and Computational Biology*. Technical White Paper, Version 0.1. Universidade do Estado de Santa Catarina – UDESC, 2026.



Distributed computing for bioinformatics and computational biology.

Coordinated by Universidade do Estado de Santa Catarina – UDESC, Brazil.

References

- [1] National Institute of Standards and Technology. *Zero Trust Architecture*. NIST Special Publication 800-207, 2020. <https://doi.org/10.6028/NIST.SP.800-207>.
- [2] OpenID Foundation. *OpenID Connect Core 1.0 incorporating errata set 2*. 2014. https://openid.net/specs/openid-connect-core-1_0-18.html.
- [3] W3C. *PROV-O: The PROV Ontology*. W3C Recommendation, 2013. <https://www.w3.org/TR/prov-o/>.
- [4] Open Container Initiative. *OCI Image Format Specification*. <https://specs.opencontainers.org/image-spec/>.
- [5] OpenSSF. *Supply-chain Levels for Software Artifacts (SLSA) Specification*. <https://slsa.dev/spec/>.
- [6] Denniss, W.; Bradley, J. *OAuth 2.0 for Native Apps*. RFC 8252, IETF, 2017. <https://doi.org/10.17487/RFC8252>.
- [7] Lewis, P. et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Advances in Neural Information Processing Systems*, 33, 2020.
- [8] Wilkinson, M. D. et al. The FAIR Guiding Principles for scientific data management and stewardship. *Scientific Data* 3, 160018, 2016. <https://doi.org/10.1038/sdata.2016.18>.
- [9] Di Tommaso, P. et al. Nextflow enables reproducible computational workflows. *Nature Biotechnology* 35, 316–319, 2017. <https://doi.org/10.1038/nbt.3820>.
- [10] Kubernetes SIG Scheduling. *Kueue Documentation*. <https://kueue.sigs.k8s.io/docs/overview/>.
- [11] Cloud Native Computing Foundation. *OpenTelemetry Documentation*. <https://opentelemetry.io/docs/>.
- [12] Prometheus Authors and The Linux Foundation. *Prometheus Documentation*. <https://prometheus.io/docs/introduction/overview/>.
- [13] NVIDIA. *NVIDIA A100 Tensor Core GPU Datasheet*. <https://www.nvidia.com/en-us/data-center/a100/>.
- [14] NVIDIA. *NVIDIA H100 Tensor Core GPU*. <https://www.nvidia.com/en-us/data-center/h100/>.
- [15] Google Android Developers. *Foreground services overview*. <https://developer.android.com/develop/background-work/services/fgs>.
- [16] Anderson, D. P. BOINC: A Platform for Volunteer Computing. *Journal of Grid Computing* 18, 99–122, 2020. <https://doi.org/10.1007/s10723-019-09497-9>.
- [17] Anderson, D. P. Globally Scheduling Volunteer Computing. *Future Internet* 13(9), 229, 2021. <https://doi.org/10.3390/fi13090229>.
- [18] Trott, O.; Olson, A. J. AutoDock Vina: improving the speed and accuracy of docking with a new scoring function, efficient optimization, and multithreading. *Journal of Computational Chemistry* 31(2), 455–461, 2010. <https://doi.org/10.1002/jcc.21334>.
- [19] Abraham, M. J. et al. GROMACS: High performance molecular simulations through multi-level parallelism from laptops to supercomputers. *SoftwareX* 1–2, 19–25, 2015. <https://doi.org/10.1016/j.softx.2015.06.001>.
- [20] Altschul, S. F. et al. Basic local alignment search tool. *Journal of Molecular Biology* 215(3), 403–410, 1990. [https://doi.org/10.1016/S0022-2836\(05\)80360-2](https://doi.org/10.1016/S0022-2836(05)80360-2).
- [21] LABIO.NETWORK Team. *LABIO.NETWORK public architecture and technical documentation*. BioinformaticaUDESC GitHub repository, accessed September 9, 2026. <https://github.com/BioinformaticaUDESC/LABIO.NETWORK>.