

QHA (Quantum Hyper-Aether) Research History to Date

From Concept Formation to the LLM_SEM Integrated Architecture

September 24, 2026

This document reconstructs the history of QHA-related discussions, design decisions, and experiments so that the evolution of the research direction can be followed.

Abstract

Quantum Hyper-Aether (QHA) is a research framework aimed at constructing an AI-native computing architecture in which semantics, purpose, context, and knowledge become primary units of computation and resource management. The research began with the idea of moving beyond conventional transaction- and process-centered computing toward a semantic-oriented execution model capable of dynamically organizing computation according to the meaning of tasks. Early work focused on virtual-machine-based realization, including VM creation, reuse, dormancy, deletion, inter-VM communication, semantic data formats, Semantic Tasks, and semantic distance. The architecture was subsequently extended using Semantic Event-based Asynchronous Graph Neural Networks (SE-AEGNN), Unknown/Discovery VMs, autonomous knowledge reacquisition, and semantic lifecycle control. Research then expanded into business continuity and disaster recovery, including adaptive and geo-distributed configurations in which semantic units can be restored or relocated according to importance, dependencies, failures, workload, and knowledge availability. These developments motivated the broader Semantic OS abstraction, where Semantic State, Semantic Scheduling, executor selection, Semantic Memory, and event-driven processing correspond to operating-system-level functions. A Semantic CPU was then proposed to accelerate candidate search, semantic-distance calculation, scoring, routing, and scheduling. Most recently, QHA has incorporated LLM_SEM as a practical semantic representation and routing mechanism. LLM_SEM provides semantic vectors and projection-based separation for identifying and routing Semantic Tasks, while QHA supplies the higher-level execution, lifecycle, scheduling, and distributed resource-management framework. The current research direction is to implement and experimentally evaluate a minimal QHA + LLM_SEM integrated prototype, initially using dummy VMs to isolate and measure the effectiveness of semantic-level routing, scheduling, and resource management.

1. Overview

All architecture diagrams and timeline figures in this document are embedded as PNG images to preserve layout reproducibility.

QHA (Quantum Hyper-Aether) is a research concept that has evolved toward an AI-native distributed computing platform in which meaning, purpose, context, and knowledge are treated as primary units of processing. In contrast, conventional clouds and operating systems manage resources mainly in physical or implementation-level units such as CPU time, memory, processes, and virtual machines.

In the early phase, the research focused on the Hyper-Aether / Quantum Hyper-Aether conceptual model, in which VMs are created, deleted, and reused according to semantic requirements and communicate through neural-network-like interfaces. The work was then refined through Semantic Tasks, semantic distance, event-driven scheduling, SE-AEGNN, Discovery VMs, knowledge reacquisition, BC/DR, and geo-distributed configurations. Later, the central abstraction shifted toward the Semantic OS.

Since August 2026, a Semantic CPU has been defined as an acceleration mechanism for the Semantic OS, with clearer separation of the roles of the Semantic ISA, Scheduler, Executor, and Semantic Memory. In September 2026, LLM_SEM, derived from the custom LLM development line, was adopted as a practical implementation candidate for semantic representation and semantic routing. The research has now moved to a minimal QHA + LLM_SEM integrated prototype. The current focus is to make the QHA abstraction experimentally testable

using semantic vectorization and routing by LLM_SEM, dummy VMs that represent processing time and load, and an event-driven scheduler.

2. Overall Research Timeline

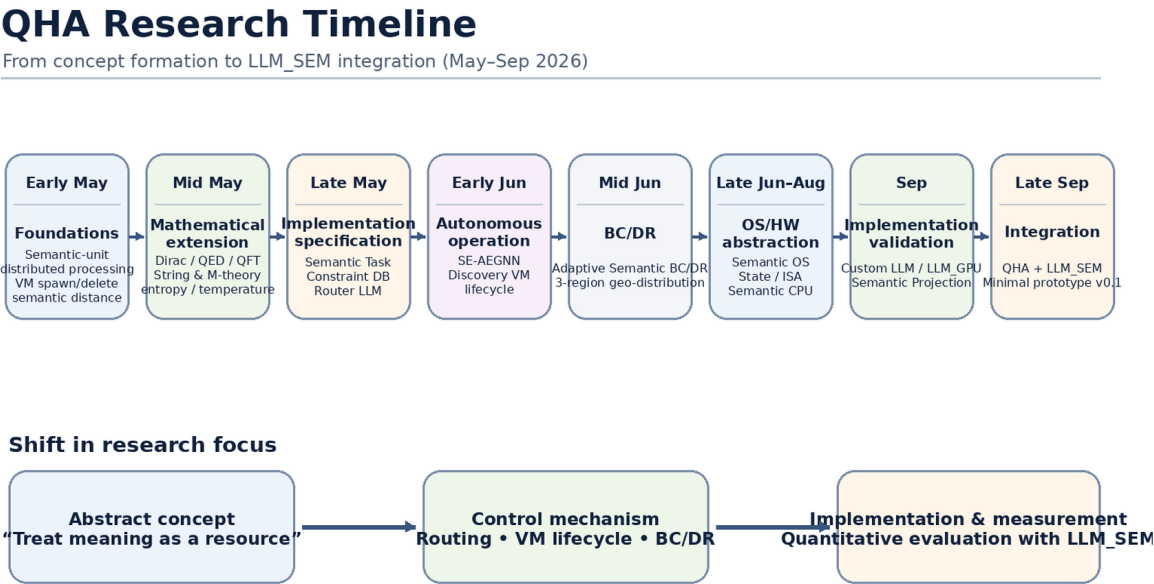


Figure 1. Overall QHA research timeline

Period	Central Theme	Main Outcome
Early May 2026	Foundations of Hyper-Aether / QHA	Semantic-unit distributed processing, VM spawn/delete, inter-VM communication, and semantic distance
Mid-May 2026	Extension using mathematical and physical models	Dirac/QED/QFT, string theory and M-theory, action S, particles/modes, entropy, and temperature correspondences
Late May 2026	Implementation specification	Semantic Task, LLM roles, VM internals, Constraint DB, Router LLM, and multiple-LLM use
Early June 2026	Autonomous operation and SE-AEGNN	VM lifecycle, Unknown/Discovery VM, Promote/Merge/Reject/Delete, and local updates
Mid-June 2026	BC/DR and geo-distribution	Comparison of four methods and failure scenarios, Adaptive

		Semantic BC/DR, and a three-region geo-distributed configuration
Late June 2026	Abstraction from QHA to Semantic OS	Systematization of Semantic State, Semantic Scheduler, executor dispatch, and Semantic Memory
August 2026	Semantic ISA / Semantic CPU	Dedicated hardware concept for accelerating semantic distance, candidate search, and scoring; separation of Sequencer/UI/ISA
September 2026	Implementation feasibility validation	Custom LLM, LLM_GPU, virtual GPU, Semantic Projection, dataset experiments, and multi-seed evaluation
Late September 2026	Adoption of LLM_SEM and QHA integration	Minimal QHA + LLM_SEM integrated prototype; VMs used as dummies representing processing time and load

3. Starting Point: Using Meaning as a Unit of Resource Management

The starting point of QHA was the question: in an AI-native cloud platform, should computing resources be managed not only by CPUs and transactions, but also by semantic units? Conventional load balancing primarily uses request counts, CPU load, queue length, and similar metrics. In generative AI and knowledge processing, however, semantic similarity, purpose, context, and required knowledge can be crucial information for deciding where a task should be processed.

From this idea, QHA defined the Semantic Task as the basic processing unit and explored a configuration in which each VM corresponds to a semantic domain, knowledge domain, or purpose. From the resource administrator's perspective, a potential benefit is that system status can be understood not only as CPU utilization but also in terms such as which semantic domains are congested and which knowledge domains are insufficient.

- Load-balancing granularity: from transaction-level units to Semantic Task units.
- Placement decisions: introduce Keyword and Semantic routing in addition to simple load-based routing.
- Semantic interpretation of resources: treat a VM not merely as a compute container but as an execution domain carrying meaning, purpose, and knowledge.

4. Early VM-Centric QHA Architecture

Evolution of the QHA Architecture

From a VM-centric model to Semantic OS / Semantic CPU / LLM_SEM

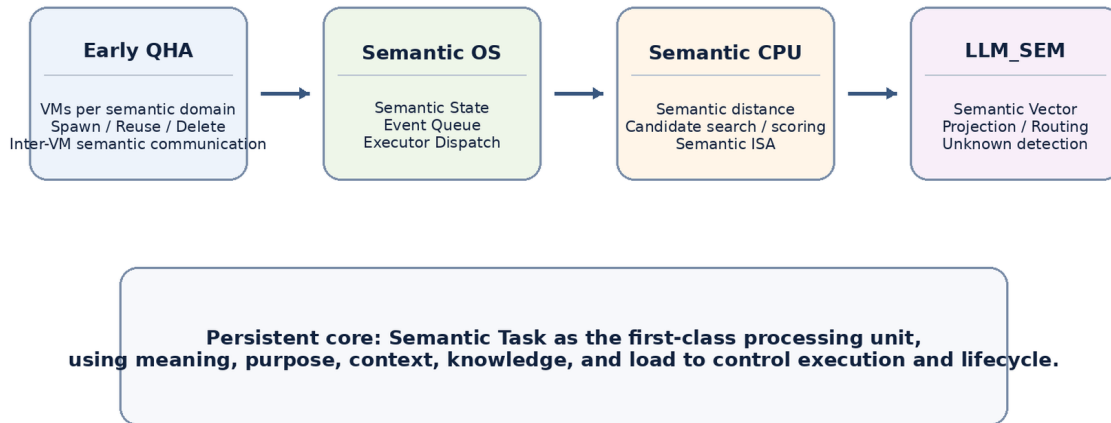


Figure 2. Evolution of the QHA architecture

In early QHA, the VM was the main system building block. A VM was created when a new semantic domain or processing purpose became necessary and was made Dormant or deleted when no longer needed. The design started with homogeneous VMs and allowed their roles to differentiate as required.

The proposed VM internals included a Semantic Filter, local knowledge, an LLM or other inference engine, and communication interfaces to other VMs. Constraint information was moved outside individual VMs into an external Constraint DB that could be shared by multiple VMs. The purpose of a downstream VM could also be inherited from an upstream VM, shifting the design from fixing everything at VM creation time toward inheriting and updating purpose at runtime.

4.1 VM Spawn, Deletion, and Reuse

- **Spawn:** create a VM when an unhandled semantic domain, increased load, a new purpose, or an unknown category appears.
- **Reuse:** if the semantic distance to an existing VM is small, reuse it instead of creating a new VM.
- **Dormant:** keep a VM in a sleeping state instead of deleting it immediately, preserving the possibility of reuse.
- **Delete:** delete a VM when it has been unused for a long time, has low semantic novelty, and has a low probability of reuse.

Because deleting a VM may remove knowledge needed for future queries, an autonomous learning mechanism was considered to detect changes in user requirements and reacquire the necessary knowledge. Under this view, VM deletion becomes closer to cache management with recoverable knowledge than to irreversible knowledge loss.

5. Inter-VM Communication and the Semantic Data Format

Inter-VM communication was initially considered in a hardware-oriented form: the N output signals of one neural network would be serialized and reconstructed as parallel inputs at another VM. The discussion later expanded to what should be used as the payload, including natural language, tokens, sentences, embeddings, and context.

The resulting direction was not to fix communication to a single representation. Instead, semantic information should be handled through a canonical abstract form and translated into executor-dependent internal representations such as embeddings, graphs, or symbolic forms. Natural language remains highly readable but is ambiguous and redundant, so the control plane should primarily use semantic vectors and structured metadata while still allowing natural-language payloads when appropriate.

- Semantic ID / Task ID
- Purpose / Intent
- Context
- Semantic representation (e.g., embedding)
- Payload (natural language, token sequence, structured data, model state, etc.)
- Metadata such as Constraint / Priority / TTL / provenance

6. Defining the Semantic Task and Semantic Distance

QHA routing required a definition of semantic distance. In addition to direct embedding-space distance, the research examined a measure based on the commonality of keyword sets associated with a semantic concept. As a result, semantic distance was positioned not as one fixed algorithm but as an abstract interface for evaluating semantic closeness.

A Semantic Task was organized as a processing unit that combines processing purpose, input meaning, context, constraints, and required knowledge. This allows the scheduler to decide not only which VM is available, but which VM is semantically most appropriate for the task.

7. Role of LLMs and the Use of Multiple LLMs

An LLM is not QHA itself; it is one type of executor responsible for semantic understanding, purpose estimation, context processing, and generation. When multiple LLMs are used, a Router LLM can classify input by semantic domain and dispatch it to an appropriate LLM or VM. Experimental configurations combining an M5Stack LLM, Python on a PC, and OpenAI services were also considered, as was the possibility of logically partitioning LLM roles.

An important design principle is that the QHA control plane should not depend entirely on an LLM. In low Semantic Temperature regions, where semantic uncertainty and change are small, rules and conventional automation can be sufficient. LLMs and exploratory processing are reserved for higher-temperature, higher-uncertainty regions.

8. Conceptual Extension Using Mathematical and Physical Models

In May 2026, correspondences with the Dirac equation, QED, quantum field theory, string theory, and M-theory were explored to describe QHA state transitions and interactions more generally. This was not a

claim that QHA is itself a physical theory. Rather, these ideas were investigated as mathematical metaphors or formalization tools for representing multiple VM types, mode changes, interactions, fields, an action S , and state transitions in a unified manner.

Entropy and Semantic Temperature were introduced in parallel. Entropy represented uncertainty or diversity in semantic states, while temperature represented the intensity of state change or uncertainty. This led to a resource-selection rule in which deterministic automation is favored at low temperature and LLM-based or exploratory processing is used more aggressively at high temperature.

9. Event-Driven Operation and SE-AEGNN

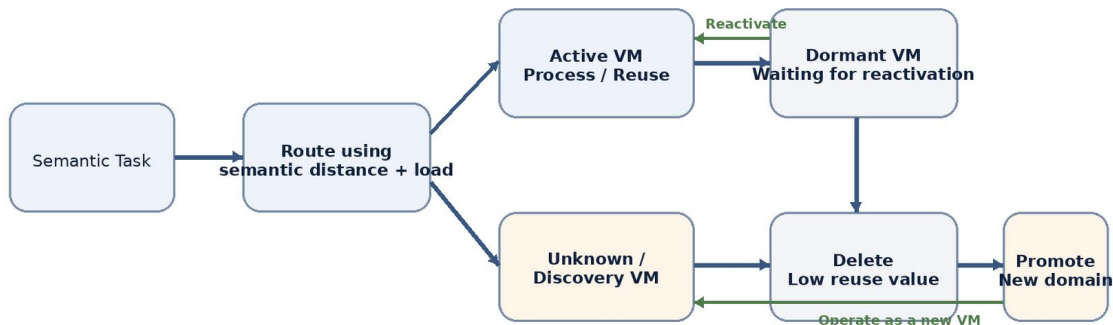
For time evolution in QHA, the research moved away from Global Update, in which every VM is continuously updated, toward local updates triggered only where events occur. Semantic Event-based Asynchronous Graph Neural Network (SE-AEGNN) was introduced as a model in which VM spawn, deletion, reuse, and knowledge reacquisition are treated as local updates on a semantic event graph.

Time itself was also treated as an event. Periodic time-tick events could evaluate Dormant VMs and prevent inactive components from disappearing indefinitely. This clarified the scheduler as an event-driven system with a queue and led naturally from QHA toward the Semantic OS abstraction.

10. Unknown / Discovery VMs and the Autonomous Lifecycle

VM Lifecycle and Unknown / Discovery Handling

Spawn, reuse, dormancy, and deletion driven by semantic events



Main decision variables: semantic distance / novelty / reuse probability / load / TTL / constraints

Figure 3. VM lifecycle and Unknown / Discovery processing

To handle Semantic Tasks that are not sufficiently close to any existing VM, the concepts of Unknown VM and Discovery VM were introduced. A Discovery VM temporarily receives unknown input and creates a provisional semantic cluster or temporary VM. Based on subsequent observations, the system selects Promote, Merge, Reject, or Delete.

- Promote: elevate the temporary cluster to an independent VM representing a new semantic domain.
- Merge: integrate it with an existing VM if it is sufficiently close.
- Reject: classify it as noise, a transient event, or a misclassification.
- Delete: remove it if its retention value and expected reuse are low.

Dormant state, semantic novelty, and reuse probability were then incorporated into deletion decisions, improving lifecycle control beyond a simple TTL-based policy.

11. Expansion into BC/DR Research

The design in which VMs are independent semantic units was also considered useful for business continuity and disaster recovery. Instead of restoring the entire system in the same shape after a failure, Adaptive Semantic BC/DR was proposed, restoring Semantic Services in an order determined by semantic importance and dependency relationships.

Minimal Python simulations compared multiple recovery methods across multiple failure scenarios and were later expanded to a three-region Geo-Distributed Autonomous Semantic BC/DR model for wide-area disasters. This raised additional issues such as a Knowledge Sync Middleware, semantic-state and knowledge synchronization across regions, and limiting fault propagation through a VM-plus-container layered design.

- Recovery priority based on semantic importance
- Semantic Task relocation to another region during regional failure
- Knowledge/state synchronization and partial recovery
- Separation of failure domains using a two-layer VM and container structure

12. From QHA to the Semantic OS

By mid-June 2026, the first major phase of QHA exploration was concluded and the research moved to a higher-level abstraction: the Semantic OS. The QHA core principle—treating meaning, purpose, context, and knowledge as processing units—was redefined as operating-system components.

In the Semantic OS, Semantic State was defined as the counterpart of machine state in a conventional CPU. The Scheduler evaluates Semantic State and selects an executor such as an LLM, logical inference engine, Python process, or external tool. Semantic Memory retains past semantic states and knowledge and can update neural models or knowledge bases when required.

12.1 Major Components of the Semantic OS

- Semantic State: meaning / context / intent / constraints / provenance, etc.
- Semantic Scheduler: selects the next operation based on semantic distance, priority, load, temperature, and constraints.
- Executor Dispatch: routes work to LLMs, symbolic logic, Python, external tools, and other executors.
- Semantic Memory: retains history, knowledge, embeddings, graph state, and related information.
- Event Queue: manages external input, semantic changes, time ticks, failures, and knowledge updates as events.

With this abstraction, a QHA VM no longer has to correspond strictly to a physical or conventional virtual machine; it can instead be interpreted as one implementation of a Semantic Execution Context.

13. Development of the Semantic CPU / Semantic ISA

In August 2026, the Semantic CPU was defined as a dedicated hardware concept for accelerating Semantic OS bottlenecks. Its purpose is not to replace the full computation of an LLM, but to parallelize and accelerate the operations repeatedly used by the Semantic Scheduler, including candidate-domain search, semantic-distance calculation, scoring, routing, and event selection.

The Semantic CPU concept separates the UI, Sequencer, Semantic ISA, and Execution Units so that semantic operations can be issued by the OS as instructions. By separating canonical Semantic State from executor-dependent representations, the upper-level API can remain stable even if embeddings, graphs, or symbolic representations change.

14. From the Custom LLM to LLM_SEM

In September 2026, a minimal Transformer/LLM was implemented in Python to examine the feasibility of QHA and the Semantic OS. A custom virtual GPU runtime, GPU ISA, and GPU scheduler were also developed. This provided an experimental base in which the model, execution system, memory management, training, and inference could all be controlled directly.

The work then evolved into LLM_SEM, focused on semantic processing. Experiments included class separation using Semantic Projection, unknown-input detection, known holdout evaluation, multi-seed evaluation, within-class and between-class similarity, semantic margin, and nearest-centroid distance.

The role of LLM_SEM is to make the 'semantics' of QHA measurable in an actual vector space instead of leaving them as a purely abstract concept. It converts a Semantic Task into an embedding, measures distance to known semantic domains, and connects the result to routing and Unknown detection. This provides an experimentally testable answer to a long-standing QHA question: how should Semantic Distance be implemented?

15. QHA + LLM_SEM Architecture as of September 2026

QHA + LLM_SEM Minimal Integrated Prototype v0.1

Current architecture as of September 2026

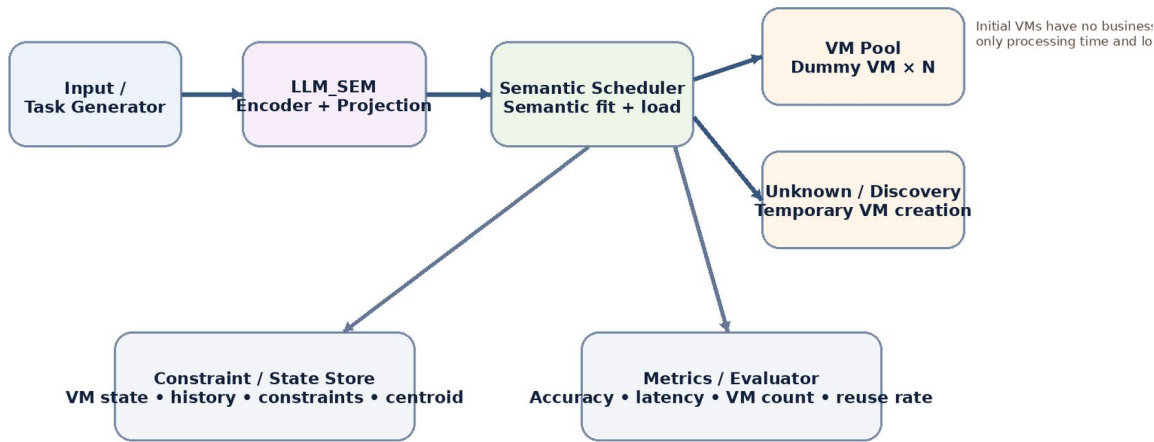


Figure 4. QHA + LLM_SEM minimal integrated prototype v0.1

In the currently adopted direction, QHA is the meaning-based control architecture, while LLM_SEM implements semantic representation and routing. VMs do not initially need to provide complete business services; in the first prototype they may simply be dummy VMs that consume assigned processing time and load.

This is because the primary purpose of the first integrated experiment is not to test application-specific processing inside each VM, but to observe how Semantic Tasks are routed to VMs and how VM spawn, reuse, deletion, and relocation respond to changing semantics and load. Dummy VMs allow QHA control policies to be measured independently of application-specific execution.

15.1 Current Division of Roles

Component	Current Role	Implementation Image
Input / Task Generator	Generate Semantic Tasks	Text input and experimental datasets
LLM_SEM	Semantic vectorization, classification, distance, and Unknown detection	Encoder + Semantic Projection
Semantic Scheduler	Routing, spawn, reuse, and deletion decisions	Python event-driven scheduler
VM Pool	Execution targets for semantic domains	Initial version uses dummy VMs based on sleep/computation load

Constraint / State Store	Share constraints, VM state, and history	External DB or Python state
Metrics / Evaluator	Measure accuracy, load, waiting time, VM count, etc.	CSV + visualization

16. Design Decisions That Have Become Relatively Stable

- The basic QHA processing unit is the Semantic Task, and meaning, purpose, and context are used in routing decisions.
- A VM is an execution context responsible for a semantic domain and has a lifecycle of spawn, reuse, Dormant, and deletion.
- Control is event-driven, and only the required parts of the system are updated locally.
- An Unknown / Discovery mechanism is provided so that unfamiliar inputs are not forced into existing categories.
- Constraint DB and shared state are external to individual VMs and can be referenced by multiple VMs.
- Semantic representation is handled canonically at the upper layer, while embeddings and similar structures remain implementation-dependent.
- An LLM is not the only executor; logical processing, Python, and external tools are selected as appropriate.
- BC/DR uses semantic importance and dependency relationships to determine recovery order and relocation.
- LLM_SEM is used as the semantic-representation and routing implementation for QHA.
- In the minimal integrated prototype, VM-internal processing is simplified as a dummy workload so that the control mechanism itself can be evaluated.

17. Major Remaining Issues

The center of the research has shifted from conceptual definition to integrated experimentation. The following issues are particularly important at present.

- How should the LLM_SEM semantic vector be stored in the QHA Semantic State?
- How should routing thresholds, Unknown-detection radius, and VM Spawn / Merge / Delete thresholds be unified?
- How should semantic fitness and actual system load be weighted against each other?
- How many VMs should be allowed, and how should over-fragmentation be prevented?
- When the semantic distribution changes over time, when should centroids or projection parameters be updated?
- What conditions should trigger reactivation of Dormant VMs and knowledge reacquisition?
- At what stage of the integrated prototype should failures and inter-region relocation be introduced?
- When dummy VMs are replaced by real application executors, how much of the observed behavior will remain valid?
- Which Semantic CPU instructions should be implemented in hardware based on measured execution profiles?

18. Next Implementation Stage

The next natural step is to implement the QHA + LLM_SEM Minimal Integrated Prototype v0.1 and observe semantic routing and the VM lifecycle at the same time. The initial experiment can use approximately six known semantic classes plus Unknown inputs, with each VM implemented as a dummy component that only consumes processing time.

1. Convert each input into a semantic vector using LLM_SEM.
2. Route using a combination of distance to existing VM centroids, the post-projection score, and current load.
3. If no suitable VM exists, send the task to Unknown / Discovery and create a temporary VM when conditions are satisfied.
4. A VM remains busy for the processing time associated with its assigned task and updates its load when processing finishes.
5. Move VMs to Dormant after a specified period of inactivity, then decide whether to reuse or delete them.
6. Record routing accuracy, Unknown precision/recall, average waiting time, p95 latency, number of VMs created, reuse rate, and load-balance metrics.
7. Compare Random, Load-based, Keyword, and Semantic routing to evaluate the QHA control mechanism.

19. Position of the Research as a Whole

QHA research began with the abstract idea of a virtual computing space in which semantically specialized VMs interact. It has gradually become more concrete through operating-system abstractions, schedulers, events, semantic state, distributed recovery, dedicated hardware concepts, and actual semantic-vector learning.

The largest conceptual change has been to stop treating QHA as a single specialized virtualization mechanism and instead separate it into three layers: the Semantic OS as the software abstraction layer, LLM_SEM as the semantic representation engine, and the Semantic CPU as a future acceleration mechanism. This separation makes staged implementation and evaluation much easier.

The current QHA is not a completed system tied to a particular VM product or a particular LLM. It remains a prototype-stage computer-architecture and operating-system research framework in which semantics are treated as first-class resource-management targets. However, because LLM_SEM now enables quantitative evaluation of Semantic Routing, the next phase can update design decisions using measured results rather than conceptual plausibility alone.

20. Terminology

Term	Meaning in This Research
QHA	Quantum Hyper-Aether: a distributed-computing concept that organizes resources in units of meaning, purpose, context, and knowledge.
Semantic Task	A processing unit containing meaning, purpose, context, and constraints.
Semantic State	The semantic counterpart of machine state

	managed by the Semantic OS.
Semantic Distance	A measure representing the closeness between a Semantic Task and a VM or knowledge domain.
Discovery VM	An exploratory VM that temporarily handles unknown or newly emerging semantic domains.
SE-AEGNN	An asynchronous graph-processing model that performs local updates in response to semantic events.
Semantic OS	An operating-system abstraction derived from the core ideas of QHA.
Semantic CPU	A hardware concept that accelerates semantic search, distance calculation, scoring, and related Semantic Scheduler operations.
LLM_SEM	An experimental semantic vectorization, Semantic Projection, and routing-evaluation system used with QHA.
Semantic Temperature	A concept representing variability, uncertainty, and the need for exploration in semantic states.

Appendix A. Timeline of Major Research Topics

Early May 2026: QHA objectives, VM types, spawn/delete conditions, Semantic Data Format, neural-network-style inter-VM communication, semantic distance, LLM adoption, action S, Dirac/QED/QFT, string/M-theory, entropy, and temperature.

Late May 2026: multiple LLMs, Router LLM, M5Stack LLM, Semantic Task, load-balancing granularity, Constraint DB, and multi-physical-server configurations.

Early June 2026: SE-AEGNN, comparison with Global Update, four methods and three scenarios, Discovery VM, Dormant state, Promote/Merge/Reject/Delete, and knowledge reacquisition.

Mid-June 2026: BC/DR, Adaptive Semantic BC/DR, wide-area disasters, three regions, Geo-Distributed Autonomous Semantic BC/DR, and a temporary conclusion of the QHA exploration phase.

Late June–August 2026: Semantic OS, Semantic State, Scheduler, executor dispatch, Semantic Memory, Semantic ISA, and Semantic CPU.

Early September 2026: custom LLM, virtual GPU, GPU ISA, training/inference, and LLM_GPU.

Late September 2026: LLM_SEM, Semantic Projection, dataset expansion, holdout evaluation, multi-seed evaluation, and the decision to integrate LLM_SEM with QHA.

Note: This document reconstructs QHA-related discussions, design work, and experimental directions through September 24, 2026. Correspondences with mathematical physics are currently used primarily as research metaphors and formalization tools and do not imply experimental validation as a physical theory.