

QHA (Quantum Hyper-Aether) これまでの研究経緯

概念形成から LLM_SEM 統合アーキテクチャまで

2026 年 9 月 24 日

本書は、これまでの QHA 関連の検討・設計・実験の流れを、研究判断の変化が追えるよう再構成したものです。

1. 概要

本書中の構成図・タイムライン図は、レイアウトの再現性を保つため、すべて画像（PNG）として埋め込んでいる。

QHA（Quantum Hyper-Aether）は、従来のクラウドやOSが「CPU時間・メモリ・プロセス・VM」といった物理／実装上の単位で資源を管理するのに対し、「意味・目的・文脈・知識」を主要な処理単位として扱うAIネイティブな分散処理基盤を目指して発展してきた研究構想である。

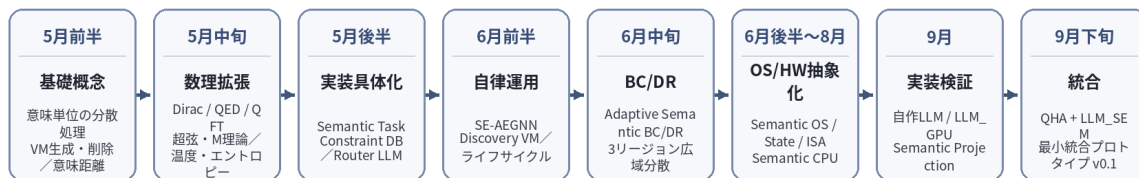
研究初期には、意味に応じてVMを生成・削除・再利用し、VM間をニューラルネットワーク的な通信で結ぶ「Hyper-Aether / Quantum Hyper-Aether」という概念モデルを中心に検討した。その後、Semantic Task、意味距離、イベント駆動スケジューリング、SE-AEGNN、Discovery VM、知識再収集、BC/DR、Geo-Distributed 構成へと具体化し、さらに抽象化の中心を「Semantic OS」へ移した。

2026年8月以降は、Semantic OSの高速化機構としてSemantic CPUを定義し、Semantic ISA、Scheduler、Executor、Semantic Memoryの役割分離を進めた。2026年9月には、自作LLM系列から派生したLLM_SEMを意味表現・意味ルーティングの実装候補として採用し、QHA + LLM_SEMの最小統合プロトタイプへ研究段階が移行している。現在の焦点は、QHAの抽象概念を、LLM_SEMによる意味ベクトル化とルーティング、ダミーVMによる処理時間・負荷表現、イベント駆動スケジューラを用いて実験可能な形に落とすことである。

2. 研究の全体タイムライン

QHA研究の全体タイムライン

概念形成から LLM_SEM 統合まで（2026年5月～9月）



研究の焦点の変化



図1 QHA 研究の全体タイムライン

時期	中心テーマ	主な到達点
2026 年 5 月前半	Hyper-Aether / QHA の基礎概念	意味を単位とする分散処理、VM 生成・

		削除、VM 間通信、意味距離の検討
2026 年 5 月中旬	数理・物理モデルの拡張	Dirac/QED/QFT、超弦理論・M 理論、作用 S、粒子・モード、エントロピー・温度の対応づけ
2026 年 5 月後半	実装仕様の具体化	Semantic Task、LLM の役割、VM 内部構成、Constraint DB、Router LLM、複数 LLM 利用
2026 年 6 月前半	自律運用と SE-AEGNN	VM ライフサイクル、Unknown/Discovery VM、Promote/Merge/Reject/Delete、局所更新
2026 年 6 月中旬	BC/DR と広域分散	4 方式・障害シナリオ比較、Adaptive Semantic BC/DR、3 リージョン Geo-Distributed 構成
2026 年 6 月後半	QHA から Semantic OS へ抽象化	Semantic State、意味スケジューラ、executor dispatch、Semantic Memory の体系化
2026 年 8 月	Semantic ISA / Semantic CPU	意味距離・候補探索・スコアリングを高速化する専用 HW 概念、Sequencer/UI/ISA 分離
2026 年 9 月	実装可能性の検証	自作 LLM、LLM_GPU、仮想 GPU、Semantic Projection、データセット・マルチシード評価
2026 年 9 月下旬	LLM_SEM 採用と QHA 統合	QHA + LLM_SEM 最小統合プロトタイプ、VM を処理時間・負荷を表すダミーとして使用

3. 出発点：意味を資源管理の単位にする

QHA の出発点は、「AI ネイティブな Cloud Platform では、計算資源を CPU やトランザクションだけでなく、意味の単位で扱うべきではないか」という問題意識であった。従来の負荷分散では、リクエスト数、CPU 負荷、キュー長などが主要な尺度になる。一方、生成 AI や知識処理では、処理内容の意味的な近さ、目的、文脈、必要知識が、どこで処理すべきかを決める重要な情報になる。

この考えから、QHA では「Semantic Task」を処理の基本単位とし、各 VM が特定の意味領域・知識領域・目的に対応する構成が検討された。リソース管理者から見ても、単なる CPU 使用率ではなく「どの意味領域が混雑しているか」「どの知識領域が不足しているか」を把握できることが利点として整理された。

- 負荷分散粒度：トランザクション単位から Semantic Task 単位へ。
- 配置判断：単純な Load-based だけでなく、Keyword / Semantic routing を導入。
- 資源の意味づけ：VM を「計算箱」ではなく、意味・目的・知識を担う実行領域として扱う。

4. VM を中心とした初期 QHA アーキテクチャ

QHAアーキテクチャの発展

VM中心モデルから Semantic OS / Semantic CPU / LLM_SEM へ

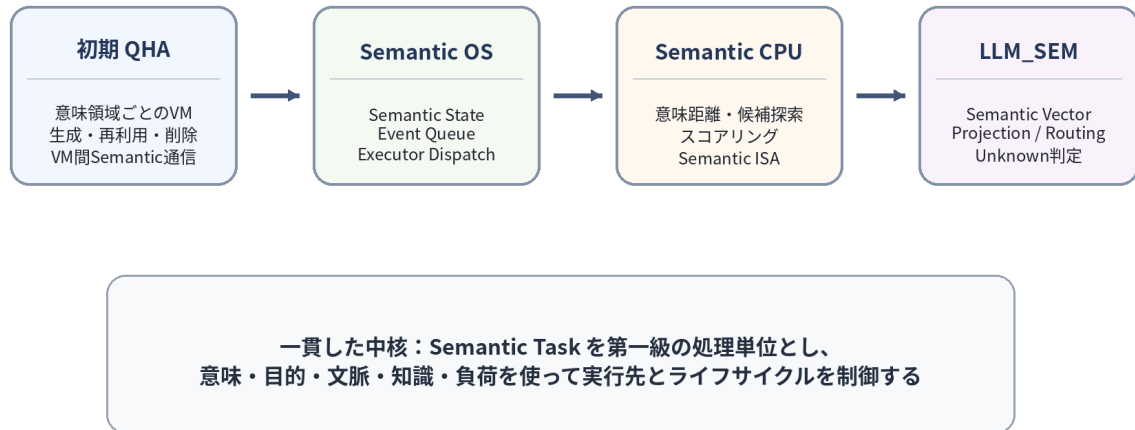


図2 QHA アーキテクチャの発展

初期 QHA では、VM がシステムの主要な構成単位であった。新しい意味領域や処理目的が必要になると VM を生成し、不要になれば Dormant 化または削除する。VM は同一タイプから開始し、必要に応じて役割が分化する設計が採用された。

VM 内部には、Semantic Filter、ローカル知識、LLM または推論器、他 VM との通信インタフェースを配置する案が検討された。制約情報は VM 内に閉じず、複数 VM で共有できる外部 Constraint DB として持つ方向に整理された。また、後続 VM の Purpose は前段 VM から伝達できるようにし、VM 生成時にすべてを固定する考え方から、実行時に目的を継承・更新する考え方へ変化した。

4.1 VM 生成・削除・再利用

- Spawn : 未処理の意味領域、負荷上昇、新規目的、未知カテゴリの発生を契機に生成。
- Reuse : 既存 VM との意味距離が小さい場合は新規生成せず再利用。
- Dormant : 直ちに削除せず休眠状態にし、再利用可能性を残す。
- Delete : 長期不使用、意味的新規性が低い、再利用確率が低い場合に削除。

削除によって知識が失われ、将来の問い合わせに答えられなくなる問題に対しては、ユーザー要求の変化を検知して知識を再収集する自律学習機構を導入する方針が検討された。これにより、VM 削除を「不可逆な知識消失」ではなく「必要なら再獲得できるキャッシュ管理」に近いものとして扱う考え方が形成された。

5. VM 間通信と Semantic Data Format

VM 間通信では、当初「ニューラルネットワークの N 本の信号線をシリアル化し、別 VM 側でパラレルに復号する」というハードウェア的イメージが検討された。その後、自然言語、token、文、embedding、context など、何を通信ペイロードとするかが議論された。

最終的には、通信形式そのものを一種類に固定するのではなく、意味表現を canonical な抽象形式で扱い、executor 依存の内部表現 (embedding / graph / symbolic など) へ変換する構造が適切だという方向に進んだ。自然言語は可読性が高いが曖昧性と冗長性があるため、制御面では意味ベクトルや構造化メタデータを使い、必要に応じて自然言語ペイロードも許容する考え方である。

- Semantic ID / Task ID
- Purpose / Intent
- Context
- Semantic representation (embedding 等)
- Payload (自然言語、token 列、構造化データ、モデル状態など)
- Constraint / Priority / TTL / provenance 等のメタデータ

6. Semantic Task と意味距離の定義

QHA のルーティングには「意味距離」の定義が不可欠となった。単純な embedding 同士の距離だけでなく、Semantic から連想されるキーワード集合の共通性を尺度にする案も検討された。これにより、意味距離は一つのアルゴリズムではなく、意味的近さを評価する抽象インターフェースとして位置づけられた。

Semantic Task は、処理目的・入力意味・文脈・制約・必要知識を一つの処理単位としてまとめたものと整理された。これにより、スケジューラは「どの VM が空いているか」だけでなく、「どの VM がこの Semantic Task に意味的に最も適しているか」を判断できるようになる。

7. LLM の位置づけと複数 LLM 利用

LLM は QHA そのものではなく、意味理解・目的推定・文脈処理・生成を担う executor の一種として位置づけられた。複数の LLM を利用する場合は Router LLM を置き、入力を意味領域に分類して適切な LLM や VM へ振り分ける構成が検討された。M5Stack LLM、PC 上の Python、OpenAI 等を組み合わせる実験構成も検討され、LLM を論理分割して複数の役割を持たせる可能性も整理された。

ここで重要なのは、QHA の制御面を LLM だけに依存させない点である。低い Semantic Temperature、すなわち状況変化や意味的不確実性が小さい領域ではルールや自動化だけで処理し、高温・高不確実性の領域のみ LLM を使うという設計思想が形成された。

8. 数理・物理モデルによる概念拡張

2026 年 5 月には、QHA の状態遷移や相互作用をより一般化して記述するため、Dirac 方程式、QED、場の量子論、超弦理論、M 理論などとの対応づけが試みられた。これは「QHA が物理理論そのものである」

という主張ではなく、複数種の VM、モード変化、相互作用、場、作用 S、状態遷移を統一的に表現するための数学的メタファー／形式化として検討されたものである。

同時に、エントロピーと **Semantic Temperature** が導入された。エントロピーは意味状態の不確実性や多様性、温度は状態変化の活発さや不確実性の大きさを表す概念として使われた。低温では決定論的自動化、高温では LLM や探索を積極的に使うという資源選択規則に発展した。

9. イベント駆動化と SE-AEGNN

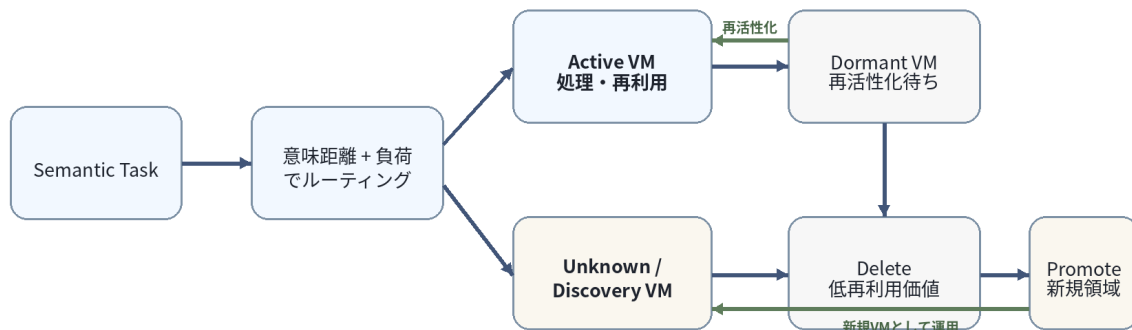
QHA の時間進行については、常時すべての VM を更新する Global Update ではなく、イベントが発生した箇所だけを局所更新する方式へ進んだ。Semantic Event-based Asynchronous Graph Neural Network (SE-AEGNN) を導入し、VM 生成・VM 削除・VM 再利用・知識再収集を意味イベントグラフ上の局所更新として扱う構想である。

この段階で、時間そのものもイベントとして扱い、定期的な time-tick イベントを発生させて Dormant VM の状態評価や沈み込み防止を行う案が整理された。これにより、スケジューラは「キューを持つイベント駆動系」として明確化され、QHA から Semantic OS への移行につながった。

10. Unknown / Discovery VM と自律ライフサイクル

VMライフサイクルと Unknown / Discovery 処理

意味イベントに基づく生成・再利用・休眠・削除



主要判断量：semantic distance / novelty / reuse probability / load / TTL / constraints

図3 VM ライフサイクルと Unknown / Discovery 処理

既存 VM のどれにも十分に近くない Semantic Task を扱うため、Unknown VM / Discovery VM の考え方が導入された。Discovery VM は未知入力を一時的に受け持ち、仮の意味クラスターや仮 VM を作成する。その後、観測結果に基づき Promote、Merge、Reject、Delete を選択する。

- Promote：新しい意味領域として独立した VM へ昇格。

- Merge : 既存 VM と十分近ければ統合。
- Reject : ノイズ、一過性、誤分類と判断。
- Delete : 保持価値が低く再利用見込みも小さい場合に削除。

さらに Dormant 状態、意味的新規性、再利用確率を Delete 条件へ組み込み、単純な TTL よりも意味的に妥当なライフサイクル制御へ改善された。

11. BC/DR 研究への展開

VM を意味単位で独立させる設計は、BC/DR にも応用できると考えられた。障害時にシステム全体を同一形状で復旧するのではなく、意味的重要度や依存関係に応じて優先順位を変え、必要な Semantic Service から復旧する「Adaptive Semantic BC/DR」が検討された。

Python による最小シミュレーションでは、複数の復旧方式と複数障害シナリオを比較し、その後、広域災害を想定した 3 リージョンの Geo-Distributed Autonomous Semantic BC/DR へ拡張した。Knowledge Sync Middleware の必要性、リージョン間の意味状態・知識同期、VM 内コンテナによる障害波及抑制も論点となった。

- 意味的重要度に基づく復旧優先度
- 地域障害時の別リージョンへの Semantic Task 再配置
- 知識・状態同期と部分復旧
- VM とコンテナの二層化による障害ドメイン分離

12. QHA から Semantic OS へ

2026 年 6 月中旬までに QHA の一連の検討を区切り、次の抽象化として Semantic OS の研究へ移った。ここでは、QHA で培った「意味・目的・文脈・知識を処理単位とする」という中核を、OS の構成要素として再定義した。

Semantic OS では、従来 CPU の machine state に相当するものとして Semantic State を定義し、Scheduler が Semantic State を評価して、LLM、論理推論、Python 処理、外部 Tool などの executor を選択する。Semantic Memory は過去の意味状態や知識を保持し、必要に応じて Neural Model や Knowledge Base を更新する。

12.1 Semantic OS の主要構成

- Semantic State : meaning / context / intent / constraints / provenance 等。
- Semantic Scheduler : 意味距離、優先度、負荷、温度、制約に基づき次の処理を選択。
- Executor Dispatch : LLM、symbolic logic、Python、外部 Tool 等への振り分け。
- Semantic Memory : 履歴、知識、埋め込み、グラフ状態等を保持。
- Event Queue : 外部入力、意味変化、time-tick、障害、知識更新をイベントとして管理。

この整理により、QHA の VM は必ずしも物理・仮想マシンである必要はなく、Semantic Execution Context の一実装として再解釈できるようになった。

13. Semantic CPU / Semantic ISA への発展

2026 年 8 月には、Semantic OS のボトルネックを高速化する専用ハードウェア概念として Semantic CPU が定義された。目的は LLM そのものの全演算を置き換えることではなく、候補意味領域の探索、semantic distance 計算、スコアリング、ルーティング、イベント選択など、Semantic Scheduler が頻繁に行う処理を並列化・高速化することである。

Semantic CPU では、UI / Sequencer / Semantic ISA / Execution Units を分離し、OS から命令列として意味演算を発行する方向が整理された。canonical な Semantic State と、executor 依存表現を分離することで、embedding、graph、symbolic representation が変更されても上位 API を維持できる構成が目標となった。

14. 自作 LLM から LLM_SEM へ

2026 年 9 月には、QHA/Semantic OS の実装可能性を検証するため、Python のみで最小 Transformer/LLM を構築し、さらに仮想 GPU ランタイム、GPU ISA、GPU scheduler を自作するプロジェクトが進められた。これにより、モデル、実行系、メモリ管理、学習、推論を自分で制御できる実験基盤が得られた。

その後、意味処理に重点を置いた LLM_SEM へ発展し、Semantic Projection を使ったクラス分離、未知入力の判定、known holdout、multi-seed 評価、within-class / between-class similarity、semantic margin、nearest centroid distance 等の評価が行われた。

LLM_SEM の役割は、QHA の「意味」を抽象概念のままにせず、実際のベクトル空間で測定可能にすることである。Semantic Task を埋め込みへ変換し、既存意味領域との距離を測り、ルーティングや Unknown 判定へ接続できる。これにより、QHA で長く残っていた「Semantic distance を何で実装するか」という問題に、実験可能な具体解が与えられた。

15. 2026 年 9 月時点の QHA + LLM_SEM 構成

QHA + LLM_SEM 最小統合プロトタイプ v0.1

2026年9月時点の現在構成

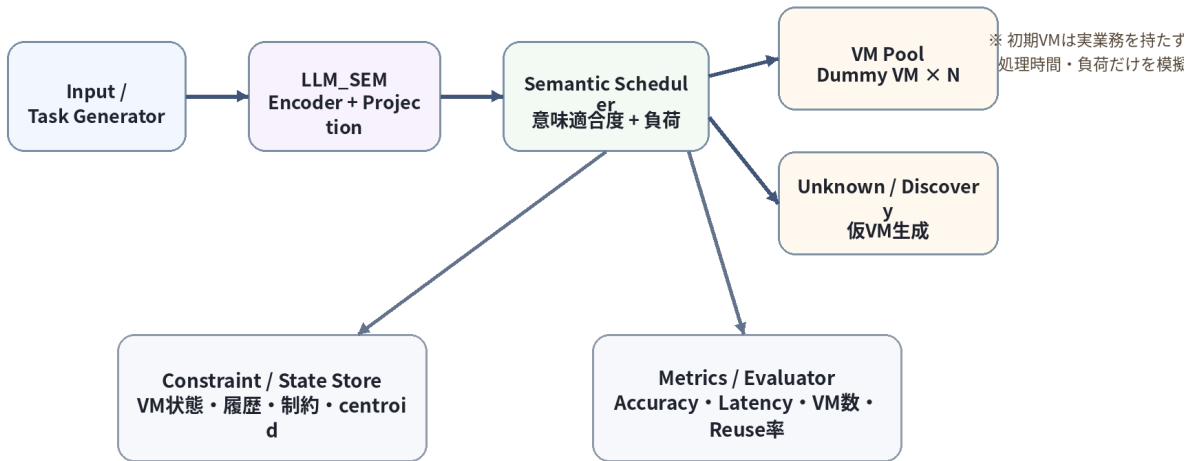


図4 QHA + LLM_SEM 最小統合プロトタイプv0.1

現在採用している方向では、QHA は意味ベースの制御アーキテクチャ、LLM_SEM は意味表現とルーティングの実装部として役割分担する。VM は最初から完全な実サービスを実装する必要はなく、まずは「割り当てられた処理時間・負荷を消費するダミーVM」としてよい、という方針に整理された。

これは、最初の統合実験で検証したい本質が「VM 内部の業務処理」ではなく、「Semantic Task がどのVM へルーティングされ、負荷と意味の変化に応じて VM 生成・再利用・削除・再配置がどう振る舞うか」であるためである。ダミーVM を使うことで、QHA 制御側の性能をアプリケーション固有処理から分離して測定できる。

15.1 現在の役割分担

構成要素	現在の役割	実装イメージ
Input / Task Generator	Semantic Task を生成	テキスト入力、実験データセット
LLM_SEM	意味ベクトル化、分類、距離、Unknown 判定	Encoder + Semantic Projection
Semantic Scheduler	ルーティング、生成、再利用、削除判断	Python のイベント駆動 scheduler
VM Pool	意味領域ごとの実行先	初期版は sleep/計算負荷によるダミー VM
Constraint / State Store	制約、VM 状態、履歴を共有	外部 DB または Python state
Metrics / Evaluator	精度、負荷、待ち時間、生成数等を計測	CSV + 可視化

16. これまでに確定度が高くなった設計判断

- QHA の基本単位は「Semantic Task」であり、意味・目的・文脈をルーティング判断に使う。
- VM は意味領域を担う実行コンテキストであり、生成・再利用・Dormant・削除のライフサイクルを持つ。
- 制御はイベント駆動で行い、必要箇所だけを局所更新する。
- Unknown / Discovery 機構を設け、既存分類へ無理に押し込まない。
- Constraint DB や共有状態は VM の外部に持ち、複数 VM から参照可能にする。
- Semantic representation は上位では canonical に扱い、embedding 等は実装依存とする。
- LLM は唯一の executor ではなく、必要に応じて論理処理・Python・外部 Tool と使い分ける。
- BC/DR は意味的重要度と依存関係を使って復旧順序と再配置を決定する。
- LLM_SEM を QHA の意味表現・ルーティング実装として利用する。
- 最小統合プロトタイプでは VM 内部処理をダミー化し、制御機構そのものを検証する。

17. 現在残っている主要課題

研究の中心は、概念定義から統合実験へ移っている。現時点で特に重要な未解決点は以下である。

- LLM_SEM の semantic vector を QHA の Semantic State へどの形式で格納するか。
- ルーティング閾値、Unknown 判定半径、VM 生成・Merge・Delete 閾値をどう統一するか。
- 意味的適合度と実負荷の重み付けをどのように決めるか。
- VM を何個まで生成可能とし、過剰分割をどう抑えるか。
- 時系列で意味分布が変化した場合に、centroid や projection をいつ更新するか。
- Dormant VM の再活性化条件と知識再収集の発火条件。
- 障害・リージョン移動を統合プロトタイプのどの段階で入れるか。
- 実アプリケーション executor へ置き換えたとき、ダミーVM で得た結果がどこまで維持されるか。
- Semantic CPU でハードウェア化する命令集合を、実測プロファイルから絞り込むこと。

18. 次の実装段階

次の自然な段階は、「QHA + LLM_SEM 最小統合プロトタイプ v0.1」を実装し、意味ルーティングと VM ライフサイクルを同時に観測することである。まずは 6 前後の既知意味クラスと Unknown 入力を用い、各 VM は処理時間のみを消費するダミーとして実装する。

1. LLM_SEM で入力を semantic vector へ変換する。
2. 既存 VM の centroid との距離、projection 後のスコア、現在負荷を組み合わせるルーティングする。
3. 該当 VM がない場合は Unknown/Discovery へ送り、条件を満たせば仮 VM を生成する。
4. VM は割当タスクに応じた時間だけ busy となり、終了後に負荷を更新する。
5. 一定期間の未使用 VM を Dormant 化し、再利用または Delete を判定する。
6. routing accuracy、unknown precision/recall、平均待ち時間、p95 latency、VM 生成数、再利用率、負荷分散度を記録する。
7. Random / Load-based / Keyword / Semantic の 4 方式と比較し、QHA 制御の有効性を評価する。

19. 研究全体としての位置づけ

QHA の研究は、当初の「意味を持つ VM が相互作用する仮想的な計算空間」という抽象的な構想から始まり、徐々に OS、スケジューラ、イベント、意味状態、分散復旧、専用ハードウェア、そして実際の意味ベクトル学習へと具体化してきた。

この過程で最も大きな変化は、QHA を単独の特殊な仮想化機構として捉えるのではなく、「Semantic OS というソフトウェア抽象層」と「LLM_SEM という意味表現エンジン」と「Semantic CPU という将来の高速化機構」の三層に分けて考えられるようになったことである。これにより、研究を段階的に実装・評価しやすくなった。

現在の QHA は、実在する VM 製品や特定 LLM に依存する完成済みシステムではなく、意味を第一級の資源管理対象とする計算機アーキテクチャ／OS 研究のプロトタイプ段階にある。しかし、LLM_SEM によって Semantic routing を定量評価できるようになったため、次の段階では概念の妥当性ではなく、実測値によって設計判断を更新できるフェーズに入っている。

20. 用語整理

用語	本研究での意味
QHA	Quantum Hyper-Aether。意味・目的・文脈・知識を単位に資源を編成する分散処理構想。
Semantic Task	意味・目的・文脈・制約を持つ処理単位。
Semantic State	Semantic OS が管理する意味的な machine state 相当。
Semantic Distance	Semantic Task と VM/知識領域の近さを表す評価量。
Discovery VM	未知・新規意味領域を一時的に受け持つ探索用 VM。
SE-AEGNN	意味イベントに応じて局所更新する非同期グラフ処理モデル。
Semantic OS	QHA の考え方を OS 抽象として整理したもの。
Semantic CPU	Semantic Scheduler 向けの意味探索・距離・スコアリング等を高速化する HW 概念。
LLM_SEM	QHA で使う意味ベクトル化・semantic projection・routing 評価の実験系。
Semantic Temperature	意味状態の変動性・不確実性・探索必要度を表す概念。

付録 A. 主な検討テーマの時系列

2026 年 5 月：QHA の目的、VM 種類、生成・削除条件、Semantic Data Format、VM 間 NN 通信、意味距離、LLM 採用、作用 S、Dirac/QED/QFT、超弦/M 理論、エントロピー・温度。

2026 年 5 月末：複数 LLM、Router LLM、M5Stack LLM、Semantic Task、負荷分散粒度、Constraint DB、物理サーバ複数構成。

2026 年 6 月上旬 : SE-AEGNN、Global Update 比較、4 方式・3 シナリオ、Discovery VM、Dormant、Promote/Merge/Reject/Delete、知識再収集。

2026 年 6 月中旬 : BC/DR、Adaptive Semantic BC/DR、広域障害、3 リージョン、Geo-Distributed Autonomous Semantic BC/DR、QHA 研究の一旦の区切り。

2026 年 6 月後半～8 月 : Semantic OS、Semantic State、Scheduler、executor dispatch、Semantic Memory、Semantic ISA、Semantic CPU。

2026 年 9 月前半 : 自作 LLM、仮想 GPU、GPU ISA、学習/推論、LLM_GPU。

2026 年 9 月後半 : LLM_SEM、Semantic Projection、データセット拡張、holdout、multi-seed 評価、QHA への統合決定。

注記 : 本書は 2026 年 9 月 24 日時点までの QHA 関連の対話・設計検討・実験方針を研究経緯として再構成したものです。数理物理との対応づけは、現段階では主として形式化・モデル化のための研究上の比喩／抽象化であり、物理理論としての実証を意味しません。