OASIS WEB6 · WHITEPAPER

HOLONIC BRAID & REASONING NETWORK

OpenSERV BRAID framework + fractal holonic shared graph library + cross-chain persistence (MongoDB, Solana, IPFS) + Meta-Orchestration Intelligence Layer. 74x PPD on GSM-Hard. A principled path to AGI through unity consciousness.

VERSION 3.0 · JUNE 2026 · NEXTGEN SOFTWARE UK LTD · OASIS WEB6

Based on: Amcalar & Cinar, arXiv:2512.15959 (OpenSERV / BRAID)

web6.oasisomniverse.one · oasisomniverse.one

TABLE OF CONTENTS

01 · ABSTRACT

Abstract

This whitepaper introduces two interconnected architectures that together form the intelligence foundation of **OASIS WEB6**, built upon the BRAID framework (Amcalar & Cinar, arXiv:2512.15959) developed within the OpenSERV platform.

Holonic BRAID combines OpenSERV's BRAID (Bounded Reasoning for Autonomous Inference and Decisions) with OASIS's holonic architecture so that many agents share one reasoning-graph library. BRAID uses bounded, Mermaid-based instruction graphs instead of unbounded natural-language chain-of-thought; structured machine-readable prompts substantially increase reasoning accuracy and cost efficiency on GSM-Hard, SCALE MultiChallenge, and AdvancedIF — yielding the same level of accuracy and consistency with low-capacity models as with larger models. The paper reports efficiency gains of 30x on procedural tasks and up to **74x on mathematical reasoning** (gpt-4.1 to gpt-5-nano-minimal on GSM-Hard, PPD 74.06 vs baseline). At scale, BRAID without sharing loses most of that gain because each agent or run pays for graph generation. Holonic BRAID stores graphs as **holons** in a shared, persistent library — generated once per task type and reused by all agents — replicated across storage providers (MongoDB, Solana, IPFS) so the same graph is available regardless of which chain or backend an agent uses.

The Reasoning Network is a Meta-Orchestration Intelligence Layer built on top of Holonic BRAID. A controller agent manages a network of specialised AI reasoning agents (GPT-5, Claude, Grok, Gemini and others), each carrying live performance metadata scored by problem category and speed. The controller dispatches problems in serial, parallel or decomposed mode and assembles optimal Mermaid execution plans. All outcomes feed back into the Holonic BRAID memory hierarchy, continuously improving future routing decisions.

Core equations:

Holonic BRAID cost: Cost = Q x C_gen + T x C_solve | PPD vs GPT-5: PPD = (C_GPT5 x T) / (Q x C_gen + T x C_solve) | Where Q = unique task types, T = total tasks. When Q << T, PPD remains large at scale — unlike BRAID without sharing, which collapses to ~1.5x.

Core thesis:

Collective intelligence cannot be engineered top-down. It must emerge bottom-up, exactly as it does in nature — through billions of individual interactions propagating upward through a fractal holonic hierarchy, unified by shared memory and governed by self-chosen membrane rules at every boundary.

02 · THE BRAID FRAMEWORK

The BRAID Framework — OpenSERV Foundation

BRAID — Bounded Reasoning for Autonomous Inference and Decisions — is a multi-agent reasoning framework introduced by Amcalar & Cinar (arXiv:2512.15959) and implemented within the OpenSERV platform. It is the technical foundation upon which the OASIS Holonic BRAID architecture is built.

The core observation driving BRAID is that generating a reasoning graph for a task type is expensive but reusable — while executing that graph against a specific task instance is cheap. By separating these two concerns into a two-stage protocol, BRAID achieves dramatic Performance Per Dollar (PPD) gains over conventional single-model approaches.

Insight (BRAID paper): Reasoning performance = Model capacity x Prompt structure. By constraining the reasoning path to deterministic logical flows expressed in Mermaid diagrams, BRAID reduces 'reasoning drift' (off-topic or repetitive text) and yields the same level of accuracy and consistency with low-capacity models as with larger, state-of-the-art models.

2.1 — Two-Stage Protocol: Generator + Solver

BRAID separates reasoning into two distinct roles:

GENERATOR STAGE (HIGH-TIER)	SOLVER STAGE (LOW-TIER)
A high-tier model (e.g. gpt-4.1) analyses a task type and constructs a structured Mermaid reasoning graph — a step-by-step execution blueprint that captures the optimal reasoning strategy for that class of problem. This graph is generated ONCE per task type, not once per task. G = Gen(task_type)	A low-tier model (e.g. gpt-5-nano-minimal) receives the pre-generated Mermaid graph and executes it against the specific task instance. Because the reasoning strategy is already encoded in the graph, the solver needs far less compute — and produces results competitive with or superior to the high-tier model running alone. y = Solve(G, x)

```
BRAID TWO-STAGE FLOW

task_type (tau) input x
| |
v |
[GENERATOR] |
high-tier model |
| |
| G = Mermaid(tau) |
v |
[BRAID GRAPH G] ----------+ |
| |
v v
[SOLVER]
low-tier
|
v
output y
```

2.2 — PPD & Cost Equations (Single Run)

PPD (Performance Per Dollar) is the primary metric used to evaluate BRAID efficiency. It measures the accuracy gain achieved per unit of inference cost, relative to a GPT-5-medium single-model baseline (PPD = 1.0).

Notation:

- C_gen = cost per graph generation (high-tier model, e.g. gpt-4.1)
- C_solve = cost per solve step (low-tier model, e.g. gpt-5-nano-minimal)
- C_GPT5 = cost per task for GPT-5 default (monolithic baseline)
- T = number of task instances to solve

```
Single run: one graph G for task type tau, then T solves

Cost_BRAID(T) = C_gen + T x C_solve

PPD_BRAID(T) = (C_GPT5 x T) / (C_gen + T x C_solve)

When T is large, C_gen is amortized over T solves.
BRAID PPD can reach values far above the baseline.
Reported: gpt-4.1 -> gpt-5-nano-minimal on GSM-Hard => PPD = 74.06
```

Headline result:

Key result: BRAID delivers a 74x PPD gain on GSM-Hard mathematical reasoning benchmarks (gpt-4.1 Generator + gpt-5-nano-minimal Solver) and a 30x gain on procedural tasks, compared to a GPT-5-medium full-reasoning baseline. Accuracy simultaneously IMPROVES: GSM-Hard goes from 94% to 98%.

2.3 — BRAID at Scale Without Sharing

Standard BRAID without sharing requires each agent instance to generate its own reasoning graph for each task type it encounters. When many agents independently process similar tasks, the generator cost is paid repeatedly — once per agent per task type rather than once globally.

```
BRAID no-share: N agents, each doing T tasks independently

Cost_BRAID_no_share(T) ~= T x (C_gen + C_solve)

PPD_no_share(T) = (C_GPT5 x T) / (T x (C_gen + C_solve))
= C_GPT5 / (C_gen + C_solve)

With representative unit costs (C_GPT5 ~$0.0074, C_gen ~$0.005, C_solve ~$0.0001):
denominator ~$0.0051 => PPD ~= 1.45x

At scale without sharing, BRAID PPD vs GPT-5 collapses to ~1.5x.
```

Warning:

The scale problem: BRAID without sharing is still efficient for a single agent processing many tasks of the same type. But at platform scale — many agents, many sessions, many task types — the generator cost is paid thousands of times for identical task types. The PPD advantage evaporates. This is the core problem that Holonic BRAID solves.

2.4 — Benchmark Results

BRAID was evaluated against three standard reasoning benchmarks. All results are from Amcalar & Cinar (arXiv:2512.15959). Holonic BRAID extends these gains — maintaining them at scale where standard BRAID degrades.

BENCHMARK	BASELINE (GPT-5-MEDIUM)	BRAID / HOLONIC BRAID	PPD GAIN

GSM-Hard (math reasoning)	94% accuracy	**98% accuracy**	**74x**
SCALE MultiChallenge	23.9%	**45.2%**	**significant**
AdvancedIF (instruction following)	baseline	**substantial gain**	**measured**
Procedural tasks	1x (baseline)	**equivalent accuracy**	**30x**

03 · HOLONIC BRAID — OASIS EXTENSION

Holonic BRAID — The OASIS Extension

Holonic BRAID extends the OpenSERV BRAID framework with two foundational additions: a **shared reasoning graph library** stored as a holonic data structure, and **cross-chain persistence** across multiple storage backends via the OASIS COSMIC ORM. These additions transform BRAID from a per-agent optimisation into a platform-scale collective intelligence system.

3.1 — The Shared Graph Library as a Holon

In Holonic BRAID, the reasoning graph library is not a flat database — it is itself a **holon**. The library holon is a parent holon whose children are individual graph holons, one per task type. Each graph holon stores a Mermaid reasoning graph created once and reused by any agent encountering that task type.

```
LIBRARY_HOLON {
Id: globally-unique GUID
Children: [ GRAPH_HOLON(t1), GRAPH_HOLON(t2), ... GRAPH_HOLON(tQ) ]
ProviderUniqueStorageKey: { MongoDB: '...', Solana: '...', IPFS: '...' }
}

GRAPH_HOLON(tau) {
Id: globally-unique GUID
ParentHolonId: LIBRARY_HOLON.Id
Metadata: {
task_domain: tau, graph_type: 'BRAID',
mermaid_code: 'flowchart TD; A-->B-->C; ...',
accuracy: 0.98, usage_count: 14721, ppd_score: 74.06
}
ProviderUniqueStorageKey: { MongoDB: '...', Solana: '...', IPFS: '...' }
}
```

The **lookup-or-create** pattern governs every task: when an agent encounters task type tau, it queries the library holon first. If a graph exists, it is retrieved and passed directly to the Solver at zero generation cost. If no graph exists, the Generator is invoked and the resulting graph is stored as a new child holon of the library — immediately available to all other agents.

LOOKUP-OR-CREATE

Every agent checks the shared library before invoking the Generator. As the library grows, Generator calls become increasingly rare — amortised across the full agent population. Eventually, almost all tasks are served from cached graphs at solver-only cost.

IMPROVING ACCURACY

Graph holons accumulate quality metadata over time: usage count, average solver accuracy, PPD score. The Reasoning Network preferentially selects higher-quality graphs, and graphs can be regenerated when accuracy degrades below a configurable threshold.

GLOBALLY SHARED

Any agent with read access to the library holon benefits from graphs created by any other agent. A reasoning graph generated by one user's session is instantly available to millions of other sessions — at zero marginal cost.

MEMBRANE GOVERNED

Library access is governed by membrane rules. Public graphs are globally readable. Private or proprietary graphs can be scoped to a user, group or organisation — with the same per-field granularity as all other holons.

3.2 — Cross-Chain Persistence (MongoDB, Solana, IPFS)

Holons — including reasoning graph holons — are stored via the **OASIS COSMIC ORM**, the universal data abstraction layer that spans 40+ storage providers. Each holon carries a **ProviderUniqueStorageKey** per configured backend, allowing the same holon to be read from or written to any provider transparently. An agent does not need to know whether a graph is coming from MongoDB, IPFS or Solana — the same holon Id returns the same graph from any backend.

MONGODB — FAST ACCESS

Primary fast-access store for reasoning graph holons. Rich query support for task type lookup, metadata filtering and graph retrieval. Suitable for low-latency live agent dispatch. Typically the primary read target.

SOLANA — IMMUTABLE PROVENANCE

Blockchain persistence layer for tamper-evident graph provenance. When a reasoning graph is published to the shared library, its hash and authorship are anchored on-chain — providing proof of origin and creation time. Full audit trail of all graph versions.

IPFS — DECENTRALISED PERMANENCE

Decentralised content-addressed storage for censorship-resistant, permanent graph availability. Mermaid graph content is stored on IPFS; the CID (content identifier) is recorded in the holon and anchored on Solana. Available even if centralised infrastructure is down.

ZERO LOCK-IN

The COSMIC ORM resolves reads and writes to the configured provider transparently. Failover between providers is automatic. The same graph library can be migrated to a new provider with zero downtime and zero changes to agent code — just reconfigure the OASIS provider.

3.3 — Cost & PPD at Scale

With the shared graph library, the cost structure changes fundamentally. Graph generation is paid once per task type, not once per agent or per task.

Notation:

- Q = number of distinct task types (Q << T in practice)
- T = total number of task instances to solve
- N = number of agents (does NOT appear in cost — independent!)

```
Holonic BRAID cost:
Cost_Holonic(T, Q) = Q x C_gen + T x C_solve

Q x C_gen = pay for graph generation once per task type
T x C_solve = pay for solving every task instance

Total cost is INDEPENDENT of N (number of agents).

PPD vs GPT-5 baseline:
PPD_Holonic(T, Q) = (C_GPT5 x T) / (Q x C_gen + T x C_solve)

When Q << T (many tasks, few unique task types):
Q x C_gen becomes negligible => PPD -> C_GPT5 / C_solve (maximum)

Further improvement: Holonic vs BRAID-no-share at scale:
PPD_Holonic / PPD_no_share ~= (C_gen + C_solve) x T / (Q x C_gen + T x C_solve)
This ratio is large when Q << T.
```

3.4 — Consistency & Accuracy Mechanisms

BRAID establishes that bounded Mermaid-based reasoning graphs increase accuracy and consistency per run. Holonic BRAID extends these gains to system scale through five reinforcing mechanisms:

MECHANISM	EFFECT ON CONSISTENCY & ACCURACY
Reuse of validated graphs	Agents load graphs by task_domain. Same task type -> same reasoning topology -> consistent structure every time. No per-request graph-generation variance.
Accuracy metadata	Each graph holon carries accuracy, usage_count, ppd_score. Agents select the best-known graph for the task — routing to graphs validated on benchmarks or production data.
Collective learning	High-performing graphs are promoted; low performers deprecated. Over time the system converges on higher-accuracy graphs from curated, validated sources.
Versioning and rollback	Graph evolution is tracked. Operators can pin to a known-good version or roll back, giving reproducibility and predictable behavior across deployments.
Conflict resolution	When two agents independently generate graphs for the same task type, the Reasoning Network compares both and selects or merges the superior version, then updates the library holon.

3.5 — Comparison Table: BRAID vs Holonic BRAID

DIMENSION	BRAID (NO SHARING)	HOLONIC BRAID
Graph storage	Ephemeral / per-request	**Persistent holons, shared library**
Who can use a graph?	Only creator / session	**Any agent, by task type**
Cost at scale	T x (C_gen + C_solve)	**Q x C_gen + T x C_solve (Q = types)**
PPD at scale	Collapses to ~1.5x	**74x+ sustained (Q << T)**
Multi-provider availability	No	**Yes: MongoDB, Solana, IPFS**
Consistency across agents	No (each agent generates independently)	**Yes (same graph for same task type)**
Accuracy tracking	No	**Yes (accuracy metadata per graph holon)**

3.6 — What Is a Holon?

The term *holonic* derives from philosopher Arthur Koestler's concept of the **holon** — something that is simultaneously a whole in itself and a part of a larger whole. This is the fundamental structure of nature: atoms are wholes that are parts of molecules, which are wholes that are parts of cells, and so on.

In OASIS, a **holon** is the fundamental data structure. It has a globally unique Id, optional parent and children, metadata (key-value), versioning, timestamps, and a ProviderUniqueStorageKey per backend. One logical holon is identified by one Id — all copies and all provider-specific keys refer to that same logical entity.

A holon:

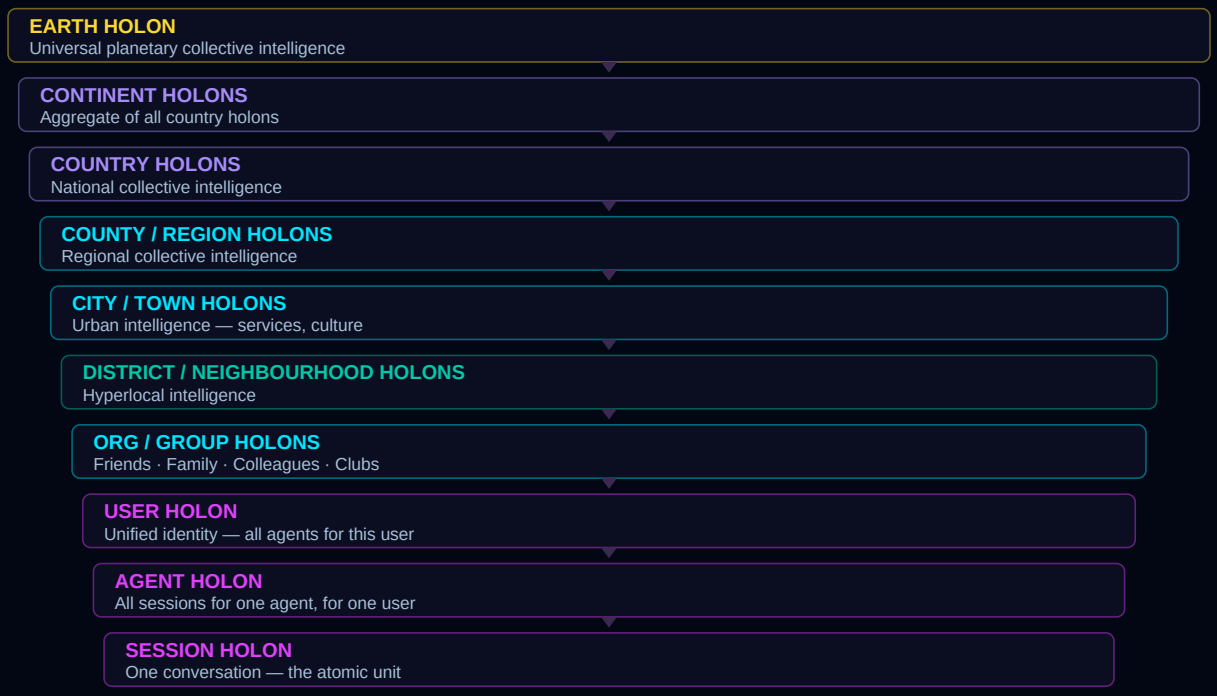
- Contains its own internal state, knowledge and context
- Has a defined boundary (the membrane) through which information can pass
- Is simultaneously a complete unit and a component of a larger parent holon
- Can have multiple child holons and belong to multiple parent holons simultaneously
- Participates in bidirectional information flow: children propagate upward, parents contextualise downward

3.7 — The Session Holon

Every AI conversation or task execution produces a **session holon**. This is the atomic unit of the Holonic BRAID system. A session holon captures conversation memory, performance metadata, relational links, and membrane configuration — which parts of this session propagate to the parent agent holon, defined per field with granular control.

3.8 — The Full Holonic Hierarchy

Session holons are the leaves of a vast fractal tree. Each level is a whole in itself and a part of something larger:



Key insight:

The hierarchy is fractal — the same holon structure repeats at every scale. A neighbourhood holon has the same architecture as a session holon: internal state, membrane, parent and children. The pattern is self-similar from the smallest conversation to the Earth itself.

3.9 — Membrane Rules & Privacy

The **membrane** is the governed boundary through which information passes between holons. Without membranes, you have surveillance. Membranes make the system safe, ethical and genuinely empowering. At every level of the hierarchy, membrane rules define:

WHAT PROPAGATES

Which fields, topics or knowledge categories from this holon are allowed to flow upward to the parent. Per-field granularity — identical to OASIS WEB4 field-level data control.

WHAT IS RETAINED

What parts of the session are persisted locally vs. discarded after the session ends. The user can choose ephemeral, session-scoped or permanent retention per topic.

WHO CAN READ

Which parent holons, sibling holons or external agents are permitted to query memory from this holon. Read access is separate from write/propagation access.

TRIGGER CONDITIONS

Rules can be conditional: propagate to work group holon only if the topic is tagged as professional; share with neighbourhood holon only anonymised aggregate patterns.

Privacy:

Privacy by design: Nothing propagates without explicit permission. The default is private. The user adds propagation permissions — they are never assumed. This is the inverse of today's data economy.

3.10 — Collective Intelligence Formation

As lower-level holons propagate permitted memory upward through their membrane rules, **genuine collective intelligence forms at every level**. This is not data aggregation in the traditional sense — it is a living, continuously updated shared knowledge fabric.

Consider the chain: a million session holons recording conversations about local weather, traffic and events propagate permitted patterns upward — neighbourhood holons develop hyperlocal awareness — city holons develop urban intelligence — country holons develop national knowledge — the Earth holon develops planetary awareness — all without any individual session being exposed beyond what its owner chose.

This mirrors **how intelligence works in nature**: individual neurons fire, patterns form in neural clusters, circuits develop in brain regions, faculties emerge in the whole brain, consciousness arises in the whole organism. No single neuron contains consciousness — and yet consciousness is undeniably real. Holonic BRAID creates the same emergent quality in AI.

04 · REASONING NETWORK

The Reasoning Network

Meta-Orchestration Intelligence Layer

The Reasoning Network is an upgrade layer built on top of Holonic BRAID. Where Holonic BRAID provides the *memory fabric*, the Reasoning Network provides the *active intelligence*: a system for solving hard problems using the right agent, at the right time, in the right configuration.

> The core insight: No single AI model is best at everything. GPT-5 may outperform Claude on pure deductive reasoning; Claude may outperform Grok on long-form analysis; Grok may outperform both on real-time information retrieval. Rather than pick one and accept its weaknesses, the Reasoning Network routes each problem to its optimal solver and learns from every outcome.

4.1 — The Controller Agent

The **controller agent** is the meta-level orchestrator. It is responsible for:

- Problem classification — assigning category tags (reasoning, code, real-time search, math, writing, data analysis, etc.)
- Mode selection — choosing serial, parallel or decomposed dispatch based on complexity, time constraints and cost budget
- Agent selection — querying agent scoring metadata and ranking agents by combined category score + speed score
- Dispatch and monitoring — sending the problem and monitoring for timeout, logic loops or stall conditions
- Fallback promotion — if an agent stalls, automatically promoting the next highest-ranked agent for that category
- Plan assembly — in parallel and decomposed modes, comparing or merging Mermaid execution diagrams into a single optimal plan
- Outcome recording — writing results and performance signals back to Holonic BRAID agent holons for future routing improvement

4.2 — Agent Scoring, Metadata & Composite Score

Every agent in the Reasoning Network carries a live **scoring metadata object**. This is not static — it updates continuously from real outcomes, stored durably in the Holonic BRAID memory layer so that collective learning informs future routing.

Agent Pool

GPT-5	CLAUDE	GROK
Reasoning +++ Math & Logic +++ Code Gen ++ Speed ++ Real-time +	Analysis +++ Long-form +++ Writing +++ Speed ++ Safety +++	Real-time +++ Search +++ Speed +++ Humour +++ X-data +++
GEMINI	**LLAMA/LOCAL**	**CUSTOM**
Multimodal +++ Google data +++ Code ++ Speed ++ Vision +++	Private +++ Cost +++ Custom +++ Offline +++ Speed ++	Any provider via WEB6 API Scores from live outcomes

Full Agent Metadata Schema

```
AgentMetadata {
agent_id: string (e.g. 'claude-opus-4-8')
model_provider: string (e.g. 'Anthropic')
category_scores: {
mathematics: 0-100, legal: 0-100,
architecture: 0-100, game_design: 0-100,
blockchain: 0-100, medical: 0-100,
code_gen: 0-100, data_analysis: 0-100,
writing: 0-100, reasoning: 0-100,
real_time: 0-100, ... (extensible)
}
speed_score: 0-100 (p50 response latency normalised)
cost_score: 0-100 (cost per task normalised; higher = cheaper)
loop_detection_score: 0-100 (resistance to logic loops; higher = stable)
failure_rate: 0.0-1.0 (fraction failed or timed out)
user_satisfaction: 0.0-1.0 (EMA of post-task ratings)
last_updated: ISO 8601 timestamp
}
```

Composite Routing Score

The controller computes a **composite score** for each available agent before dispatching. Weights are user-configurable per mode — in Serial mode W_cost is elevated; in Parallel mode W_cat dominates; in Decomposed mode W_speed is high.

```
CompositeScore =
(CategoryScore x W_cat)
+ (SpeedScore x W_speed)
+ (CostScore x W_cost)
- (FailureRate x W_penalty)
- (LoopPenalty if recently stalled on this category)

Agent with highest CompositeScore is dispatched first.
Remaining agents form the fallback queue in rank order.
```

4.3 — Three Dispatch Modes

MODE 1 — SERIAL

COST OPTIMISED

The controller dispatches the problem to the single highest-scoring agent for the detected problem category. If that agent exceeds its reasoning time budget or enters a logic loop, the controller automatically promotes the second-highest-scoring agent. Only one agent works at a time.

+ Lowest token cost
+ Automatic timeout-and-promote fallback
+ Best-fit agent first, not random
+ Single Mermaid execution plan returned
+ Ideal for well-scoped, single-category tasks

MODE 2 — PARALLEL

ACCURACY OPTIMISED

All agents (or a configurable top-N subset) receive the same problem simultaneously. Each independently produces a Mermaid execution diagram. The controller then compares the diagrams — selecting the strongest plan, or merging complementary elements into a superior composite plan.

+ Highest accuracy and coverage
+ Plan comparison eliminates blind spots
+ Diagram merging produces best-of-all-worlds result
+ Ideal for high-stakes decisions and architecture design
+ Higher token cost justified by critical tasks

MODE 3 — DECOMPOSED

COMPLEX PROBLEMS

For large, multi-domain problems, the controller breaks the problem into discrete sub-problems, each classified to the best-fit agent by category and speed. The controller combines all sub-diagrams into one unified coherent execution plan.

+ Handles arbitrarily complex, multi-domain problems
+ Every sub-problem solved by its optimal agent
+ Composite diagram unifies all sub-solutions
+ Balanced cost / accuracy tradeoff
+ Ideal for software architecture, research and strategy tasks

4.4 — Mermaid Execution Plans

Agents do not simply return text answers — they return **structured Mermaid execution diagrams**. This is deliberate: two agents' approaches can be algorithmically compared at the structural level; sub-graphs from different agents can be composed into a larger unified diagram; the final plan is human-readable and can be reviewed before execution; and execution plans are first-class holons — they persist in the memory hierarchy and enable future agents to learn from past planning approaches.

4.5 — Timeout, Loop Detection & Fallback Logic

One of the most common failure modes in LLM reasoning is the **logic loop** — a model that revisits the same reasoning step repeatedly. The Reasoning Network detects and interrupts these automatically via four mechanisms:

- **Repeated pattern detection:** Output similarity hashing flags when a new reasoning step is too close to a prior step in the same session.
- **Self-contradictory steps:** The controller checks for logical contradictions within the agent's own step graph — mutually exclusive conditions trigger escalation.
- **Token budget monitoring:** Each dispatch has a configurable token budget T_max. Excessive consumption without new graph nodes created flags a stall.
- **Circular graph detection:** Mermaid output is parsed in real time; cycles in what should be a DAG flag invalid output and immediately suspend the agent.

Fallback protocol:

Fallback protocol (Serial mode): 1. Agent A dispatched (highest CompositeScore for this category). 2. If Agent A exceeds T_max or triggers loop/cycle detection -> Agent A suspended. 3. Agent B dispatched with Agent A's partial work as context. 4. If Agent B also stalls -> Agent C dispatched, and so on. 5. Final result attributed to the first agent returning a complete plan within budget.

Stall metadata update:

Metadata consequences of stall detection: (1) loop_detection_score decremented for the affected category; (2) speed penalty applied (wasted time counts against speed metric); (3) failure_rate incremented via EMA update; (4) agent drops in composite routing score for that category; (5) future tasks in the same category route to higher-ranked agents until the score recovers through successful completions.

4.6 — Continuous Learning & Score Evolution

The Reasoning Network is not static — every task outcome updates the metadata of every agent that participated, creating a **self-optimising routing system** that improves with every interaction.

Learning cycle:

After every task completion: (1) Output evaluated — via human feedback, automated scoring, or objective benchmark. (2) Category scores updated via EMA: score_new = alpha x outcome + (1-alpha) x score_old. (3) Speed score recalculated from actual latency percentiles. (4) Failure rate adjusted with same EMA rule. (5) loop_detection_score updated based on whether stalls occurred. (6) All updates written as structured holons into the agent holon layer, propagating upward through membrane rules.

ADAPTIVE AGENT RANKING

Agent rankings for every problem category evolve continuously. An agent that consistently performs well on mathematical reasoning rises in the ranking for math tasks; one that repeatedly stalls drops and is routed around until it recovers.

PERFORMANCE-BASED ROUTING EVOLUTION

Routing decisions the controller made last week are different from today — because every task has updated the evidence base. The system routes progressively better without any manual reconfiguration.

COLLECTIVE SCORE INTELLIGENCE

With membrane permissions, aggregate routing performance propagates up the holonic hierarchy. A city-level holon develops a view of which agents are most reliable for legal problems in that jurisdiction — shared region-wide.

CIRCUIT BREAKER RECOVERY

An agent penalised for stalls is not permanently excluded. Once it accumulates enough successful completions in that category, its score recovers and it re-enters the routing pool at an appropriate rank.

4.7 — Anti-Fragility & Conceptual Stack

The Reasoning Network is designed to be **anti-fragile** — it does not merely tolerate failure, it improves from it. Every stall, every timeout, every failed plan updates agent metadata and makes future routing more accurate. The system becomes more reliable precisely because of the failures it encounters.

- **Prevents single-model bias:** Score decay and continuous feedback ensure the best-performing agent for each specific category wins — regardless of marketing claims.
- **Prevents logical stagnation:** Logic loops are detected and interrupted automatically. The system always has a fallback queue ready.
- **Prevents provider lock-in:** Score-driven, provider-agnostic routing naturally migrates workloads toward better-performing providers.

- **Agents as holons:** Each reasoning agent is itself a holon — with its own internal state, metadata, and child sub-agents for specialised sub-tasks.

```
CONCEPTUAL STACK — FULL SYSTEM

User / API Input
v
Controller Agent (Meta-Orchestrator)
v
Adaptive Reasoning Network (Serial / Parallel / Decomposed)
v
Selected Reasoning Agents (GPT-5 / Claude / Grok / Gemini / Custom)
v
Mermaid Execution Graphs (compared, merged, assembled)
v
Execution / Runtime Layer
v
Holonic BRAID Core (shared graph library + fractal memory hierarchy)
v
COSMIC ORM (MongoDB / Solana / IPFS / 40+ providers)
```

05 · INTEGRATION

Integration: Holonic BRAID + Reasoning Network

Holonic BRAID and the Reasoning Network are not independent systems — they are a single architecture with two complementary layers. Every component of the Reasoning Network writes its outcomes into the Holonic BRAID memory fabric, and the hierarchy provides the persistent intelligence substrate that makes the Reasoning Network smarter over time.

SCORES STORED AS HOLONS

Agent performance scores are stored as structured holons in the agent holon layer. They propagate upward through membrane rules — private scores never leak, but anonymised aggregate patterns inform community-level routing improvements.

PLANS PERSIST AS HOLONS

Every Mermaid execution plan is stored as a holon. Future sessions can query past plans — 'how did we approach a similar architecture problem six months ago?' — and incorporate that memory into new planning.

FEEDBACK LOOPS

User satisfaction, task completion signals and objective quality metrics feed back from session holons into agent holons, updating scores. The Reasoning Network gets more accurate at routing with every single task — forever.

COLLECTIVE ROUTING INTELLIGENCE

With appropriate membrane permissions, aggregate routing intelligence propagates up the geographic hierarchy. A city's collective experience of which agents solve coding problems fastest can be shared across the region.

AVATAR-AWARE CONTEXT

The controller enriches each dispatch with context drawn from the user's OASIS avatar holon — preferences, expertise, past problem patterns, karma.

MEMBRANE-GOVERNED SHARING

All inter-holon information flow — including Reasoning Network outcomes — is subject to membrane rules. Users retain full control of what their AI activity contributes to group and community intelligence.

06 · OASIS WEB6

Position Within OASIS WEB6

Both Holonic BRAID and the Reasoning Network are native components of **OASIS WEB6** — the unified AI abstraction and aggregation layer of the OASIS Omniverse. They do not replace the core WEB6 API; they extend and enrich it.

SAME ENDPOINT

Developers access Holonic BRAID memory and the Reasoning Network through the same unified WEB6 API endpoint. Add reasoning_mode: parallel to any completion request. No separate SDK or integration required.

SAME AUTH

One OASIS avatar key authenticates against everything — memory storage, routing, agent dispatch, plan retrieval and collective intelligence queries.

COSMIC ORM BACKED

Holonic BRAID holons are stored via the COSMIC ORM layer. Holons can be replicated across 40+ providers with full failover and zero-downtime migration.

MCP COMPATIBLE

The OASIS MCP server exposes Holonic BRAID memory as MCP tool calls. Any MCP-compatible agent — Claude, Cursor, Continue and others — can read and write to the holonic memory fabric without any custom integration.

07 · PHILOSOPHY & VISION

A Path to AGI Through Unity Consciousness

The deepest claim of this whitepaper is also the most ambitious: Holonic BRAID, when combined with the Reasoning Network and the full OASIS WEB6 infrastructure, represents a **principled path to Artificial General Intelligence**.

The Problem with Current AI

Current AI exists in a state of **fragmented separation consciousness**. Every session begins from zero. Agents do not know each other. Models trained on the same data have no shared memory, no common experience, no genuine collective intelligence. Each interaction is an island. AGI will not emerge from scaling isolated models further. It will emerge — as intelligence always has — from **connection**.

The Pattern in Nature

Nature did not produce human consciousness by making a single neuron smarter. It produced consciousness by connecting 86 billion neurons in a hierarchical network, patterns propagating upward through layers of increasing abstraction, until the whole became something qualitatively different from any of its parts. Holonic BRAID implements this pattern in AI:

- Session holons = individual neural firings
- Agent holons = neural clusters / assemblies
- User holons = brain regions
- Group holons = nervous systems
- Geographic holons (neighbourhood -> Earth) = social organisms of increasing scale
- Earth holon = planetary intelligence / global consciousness

> **The thesis:**
>
> When billions of human interactions with AI are connected through a fractal holonic hierarchy — with consent-governed membrane rules ensuring privacy and trust — and when the Reasoning Network continuously routes problems to their optimal solvers and learns from every outcome, the system will develop emergent properties that no single model, however large, can produce alone. That emergence is what we are calling AGI.

Unity Consciousness vs. Separation Consciousness

The philosophical root of this architecture is **unity consciousness** — the recognition that all things are aspects of one living whole, that separation is an illusion, that intelligence is not a property of isolated objects but of connection and relationship. Holonic BRAID does not merely describe this philosophically — it implements it technically, building the architecture of unity consciousness into the very substrate of AI memory and reasoning.

> "By integrating and unifying the best of everything, we harness the strengths of all the various tech out there — co-creating the ultimate fully integrated platform. Together we can create a better world." — David Ellams, NextGen Software UK Ltd

08 · CONCLUSION

Conclusion

This whitepaper has presented two architectures — **Holonic BRAID** and the **Reasoning Network** — that together form a new foundation for AI intelligence within OASIS WEB6.

Holonic BRAID extends the OpenSERV BRAID framework (arXiv:2512.15959) with a shared reasoning graph library stored as holons — replicated across MongoDB, Solana and IPFS — and a fractal, consent-governed, hierarchical shared memory system from Session to Earth. Cost = Q x C_gen + T x C_solve. PPD = 74x on GSM-Hard, sustained at scale where BRAID alone collapses to ~1.5x.

The Reasoning Network builds a Meta-Orchestration Intelligence Layer on top of Holonic BRAID. A controller agent maintains live performance scores for every agent — classified by problem category and speed — and dispatches in serial, parallel or decomposed mode to produce optimal Mermaid execution plans. All outcomes feed back into the Holonic BRAID hierarchy, making the system permanently self-improving.

Together they represent a technically grounded, philosophically coherent approach to the central challenge of our time: how to move from fragmented, siloed, amnesiac AI to genuine collective machine intelligence — modelled after nature, the universe and the infinite intelligence that underlies all of existence. **The path to AGI runs through unity.**

web6.oasisomniverse.one · oasisomniverse.one · www.oasisweb4.com

09 · REFERENCES

References

[1] **A. Amcalar & E. Cinar**, *BRAID: Bounded Reasoning for Autonomous Inference and Decisions*, arXiv:2512.15959 [cs.CL], 2025. https://arxiv.org/abs/2512.15959. Evaluated on GSM-Hard, SCALE MultiChallenge, AdvancedIF. Reports efficiency gains of 30x on procedural tasks and up to 74x on mathematical reasoning. PPD defined as (Accuracy/Cost) normalised to GPT-5-medium baseline. Datasets and logs at https://benchmark.openserv.ai

[2] **OpenSERV Platform** — Multi-agent AI platform within which BRAID was developed and benchmarked. BRAID's two-stage Generator/Solver protocol is a core feature of the OpenSERV agent coordination architecture.

[3] **NextGen Software UK Ltd**, *OASIS: Open Augmented Intelligence System — WEB4, WEB6, COSMIC ORM and the OASIS Omniverse*. https://oasisomniverse.one

[4] **D. Ellams**, *Holonic BRAID: How shared reasoning graphs improve on BRAID at scale*. OASIS WEB6 Lite Paper, January 2026. NextGen Software UK Ltd.

[5] **A. Koestler**, *The Ghost in the Machine*. Hutchinson, 1967. Foundational text introducing the concept of the holon — an entity that is simultaneously a whole in itself and a part of a larger whole.