Contents

Intelligence Aggregator Architecture

Why Radiant Outperforms Any Single Model

Version: 4.18.3 | December 2024

The Core Principle: A System > A Model

Radiant (Think Tank) produces significantly better results than any single state-of-the-art model (GPT-4o, Claude 3.5 Opus, Gemini Ultra, etc.)—not because it is "smarter" in a raw IQ sense, but because of a fundamental architectural principle:

A well-designed system will always outperform any single component within it.

A SOTA model like Gemini Ultra is a single engine. Radiant is the Formula 1 team that uses that engine. By wrapping models in layers of verification, diverse reasoning, and deterministic tools, Radiant raises the **floor** of reliability and the **ceiling** of complexity.

---

1. Mixture of Agents (MoA) Advantage

Recent research proves that an ensemble of models often outperforms a single superior model. Radiant doesn't just ask one model—it **triangulates the truth**.

Single Model Approach

```
User → "Complex physics question" → Gemini Ultra → Response

Problem: If Gemini has a blind spot or bias in that area,
         it hallucinates confidently with no correction.
```

Radiant Ensemble Approach

```
User → "Complex physics question"
         ↓

    ↓         ↓            ↓
 Gemini    Claude      DeepSeek
 (reasoning) (nuance)   (math)
    ↓         ↓            ↓

         ↓
    Synthesizer Model
    "Find consensus, discard outliers"
         ↓
    Verified Response
```

Result: Statistically filters out "hallucination noise" that any single model inevitably produces.

---

2. Adversarial Verification (The Critic Loop)

A single model struggles to "check its own work" because the same neural pathways that made the mistake will likely validate it.

Single Model Self-Check

```
User: "Draft a secure legal contract"
Model: [Generates contract with subtle error]
User: "Is this correct?"
Model: "Yes, this is correct" ← Doubles down on error
```

Radiant Cross-Provider Check

```
Step 1 - Draft:
  Gemini generates the contract

Step 2 - Audit:
  Claude (different provider, different training data)
  receives the draft with hostile persona:
  "You are a Senior Security Auditor. Find loopholes."

Step 3 - Refine:
```

```
  If Claude finds issues → Gemini rewrites
  Loop until PASS or max iterations

Step 4 - Deliver:
  User receives vetted, peer-reviewed output
```

Result: A peer-reviewed output versus a first draft.

---

3. Execution vs. Simulation (The Sandbox Advantage)

SOTA models are **probabilistic text generators**. They simulate logic. Radiant can be a **deterministic execution engine**.

Single Model Code Generation

```
User: "Write a Python script to visualize data"
Model: [Writes code that looks perfect]

Reality: Uses a function deprecated in 2024
         → Code crashes when user runs it
```

Radiant Draft-Verify-Patch Loop

```
Step 1: Gemini writes the code

Step 2: Radiant executes code in isolated sandbox (Micro-VM)

Step 3: Radiant catches DeprecationWarning from stderr

Step 4: Radiant feeds error back to Gemini: "Fix this error"

Step 5: Repeat until code passes execution

Result: User receives code GUARANTEED to run
```

Result: Working code, not code that "looks like it would run."

---

4. Avoiding the Safety Tax

Generalist models are heavily tuned for general safety, which often degrades performance in niche or technical domains (the "alignment tax").

Single Model Safety Limitation

```
User: "Create a penetration testing strategy"
Model: "I can't help with that" or [generic watered-down answer]
```

Radiant Specialized Routing

```
Brain Router detects: Domain = "Cybersecurity"

Routes to: Self-hosted uncensored model
           (Running on Radiant's SageMaker layer)
           Fine-tuned specifically for security auditing

Result: Professional, actionable penetration test plan
```

Result: Professional output instead of a safety lecture.

---

5. The Radiant Multiplier

Radiant wins because **Radiant includes the SOTA model**.

```
If:  Radiant = SOTA Model + Verification + Tools + Memory

Then: Radiant > SOTA Model (mathematically certain)
```

Comparison Matrix

Feature	Single SOTA Model	Radiant (Orchestrator)
Reliability	Single point of failure (hallucination)	Consensus-based (MoA) verification
Code Output	Probabilistic (might run)	Deterministic (verified in sandbox)
Bias	Provider-specific training bias	Bias cancellation (Google + Anthropic + OpenAI)
Long Context	"Lost in the Middle" syndrome	Map-Reduce processing of massive datasets
Domain Expertise	Safety-filtered generalist	Specialized model routing
Self-Correction	Validates own errors	Cross-provider adversarial checking

---

Trade-offs

Radiant's superiority comes with costs:

Trade-off	Impact	Mitigation
Latency	Higher (multiple steps)	Parallel execution, caching
Cost	2-4x more tokens	Use selectively for high-value tasks
Complexity	More moving parts	Robust fallback chains

Recommendation: Enable MoA and Verification for high-stakes tasks (legal, medical, financial, security). Use single-model routing for casual conversations.

Configuration

See RADIANT Admin Guide - Intelligence Aggregator for configuration options.

Default Settings

Feature	Default	Recommended For
Uncertainty Detection	**On**	All tasks
Success Memory	**On**	All tasks
MoA Synthesis	Off	Research, analysis, high-stakes
Cross-Provider Verification	Off	Legal, code, security
Code Execution	Off	Coding mode only

Related Documentation

- RADIANT Admin Guide - Full configuration
- Think Tank Admin Guide - User-facing features
- Provider Rejection Handling - Fallback system

Document Version: 4.18.3 Last Updated: December 2024