iJRASET
International Journal For Research in Applied Science and Engineering Technology

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY

Volume: 14 **Issue:** V **Month of publication:** May 2026

DOI: https://doi.org/10.22214/ijraset.2026.83251

www.ijraset.com

Call: 08813907089 | E-mail ID: ijraset@gmail.com

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 14 Issue V May 2026- Available at www.ijraset.com

Zypher OS: An AI-Native Operating System with Semantic Vector VFS, Intent-Driven Persona Boot, and Hybrid Cloud-Local LLM Orchestration

Sree Krishna A[1], Bhuvana S R[2], Niranjana S S[3], Pranav Edwin[4], Mrs. Sagara M R[5]
[1, 2, 3, 4]*Dept. of Computer Science & Engineering (Data Science), St. Thomas Institute for Science & Technology, Trivandrum, India*
[5]*Asst. Professor, Dept. of Computer Science & Engineering (Data Science), St. Thomas Institute for Science & Technology, Trivandrum, India*

Abstract: ***Traditional operating systems rely on hierarchical, path-centric file systems and static desktop environments that impose a growing cognitive burden on users navigating large, heterogeneous data collections. This paper presents Zypher OS, an AI-native operating system built on Ubuntu 22.04 that fundamentally redefines three pillars of desktop computing: file storage and retrieval, session initialization, and developer tooling. The core contribution is a Vector-Native Virtual File System (VFS), which maintains a hybrid semantic index using LanceDB for 768-dimensional vector embeddings and Tantivy for sub-millisecond keyword search, coupled with an immutable UUID-based identity layer that preserves semantic context across file renames and moves. The second contribution is Persona Boot, an intent-driven session governance system that replaces static login sessions with LLM-synthesized workspace manifests. The third and novel contribution is a Hybrid Cloud-Local LLM Orchestration Layer, which delegates large-scale, privacysensitive workloads—including full codebase analysis and reverse engineering of decompiled software—to an on-device local LLM (Gemma 4 4B), while routing high-level reasoning to a cloud AI master (Gemini), reducing cloud token consumption by up to 94% on million-line codebases. Theoretical performance analysis demonstrates significant improvements in search recall, timeto-task, and reverse engineering documentation coverage over conventional approaches.***

Index Terms: ***AI-native operating system, vector file system, semantic search, RAG, intent-driven computing, LLM orchestration, hybrid AI, reverse engineering, Ubuntu, Rust.***

I. INTRODUCTION

Since the introduction of hierarchical file systems in Unix [1], the dominant mental model for personal computing has remained "path-oriented": files are organized by where they reside on disk, and discovery depends on the user's recall of that location. This paradigm scales poorly in the modern era, where a single user may generate thousands of documents, code artifacts, and media files across overlapping projects. Users frequently experience what we term the *Recall Gap*— the inability to locate information whose conceptual relevance is clear but whose physical path is forgotten.

Simultaneously, the emergence of large language models (LLMs) has unlocked a new class of human-computer interaction centered on natural language intent. However, existing integrations treat LLMs as external applications layered atop unchanged OS primitives, rather than first-class participants in the operating system lifecycle.

Zypher OS addresses these issues with three tightly integrated OS-level subsystems:

1) A Vector VFS Daemon—a Rust-based background service that intercepts filesystem events and maintains a continuously updated semantic index, exposing a REST API for applications.

2) A Persona Boot Engine—which replaces user login sessions with intent-driven workspace deployments governed by LLM-generated manifests.

3) A Hybrid AI Orchestrator—a novel master-slave architecture pairing a cloud LLM (Gemini) with an ondevice LLM (Gemma 4 4B) to enable unlimited-context codebase analysis and reverse engineering documentation, while preserving data privacy and minimizing cloud token expenditure.

The system runs natively on Ubuntu 22.04 LTS as a collection of systemd service units, making it a genuine OS-level feature rather than an application layer.

II. RELATED WORK

1) Virtual File Systems. The VFS abstraction [2] provides a uniform interface to heterogeneous storage backends. Semantic File Systems [10] introduced attribute-based retrieval in the early 1990s, but lacked scalable embedding infrastructure. Our work extends this by integrating modern neural embeddings at the indexing layer.
2) Vector Databases. Recent work on ANN (Approximate Nearest Neighbor) indexing [3] has enabled high-dimensional similarity search at scale. LanceDB [6] provides a serverless, column-oriented vector store suitable for on-device deployment, which we adopt as the core of our semantic index.
3) Retrieval-Augmented Generation. RAG [4] established the pattern of augmenting LLM inference with retrieved documents. We embed this pattern directly into the OS filesystem layer, rather than as an application-level construct.
4) Intent-Driven Computing. Context-aware systems [11] and activity-based computing [12] explored adapting system behavior to user context, but without the expressiveness of modern LLMs. Zypher OS's Persona Boot operationalizes these concepts using natural language intent vectors.
5) Local LLM Deployment. Ollama [8] and similar frameworks have made it practical to run quantized LLMs on consumer hardware. Our work is the first to integrate local LLM inference as a first-class OS service for filesystem intelligence and agentic coding assistance.

III. SYSTEM ARCHITECTURE

Fig. 1 presents the layered architecture of Zypher OS. The system is organized into five layers on top of the Ubuntu kernel: the Vector VFS Daemon (L1), the Persona Boot Engine and Hybrid AI Orchestrator (L2), the Intelligent File Manager UI (L3), and the User Interaction Layer (L4). Fig. 2 shows the Zypher OS desktop environment running on Ubuntu 22.04 with the custom GNOME shell integration, semantic search bar, and system tray indicators.

A. *Vector VFS Daemon (L1)*

The VFS daemon is implemented in Rust using the Axum web framework and exposes a REST API on port 7777 as a systemd service. Its internal pipeline is:

1) *Event Capture:* The notify crate monitors all registered *Smart Folders* for filesystem events (create, modify, delete, rename). Events are debounced with a 500ms window to prevent indexing thrash during rapid edits.
2) *Content Parsing:* A multi-format parser extracts plain text from documents (PDF, markdown, source code). For image files (when the index_images config flag is set), a Vision LLM (llava) generates a textual description, effectively extending semantic indexing to visual assets.

Fig. 1. Zypher OS System Architecture: a five-layer AI-native OS stack integrating semantic VFS, intent-driven Persona Boot, and Hybrid CloudLocal LLM Orchestration on Ubuntu 22.04.

Fig. 2. Zypher OS desktop environment: custom GNOME shell with the Zypher OS branding, semantic search bar (top-centre), and application dock.

The search bar directly queries the VFS REST API for semantic file retrieval.

3) *LLM Summarization and Auto-Tagging:* Extracted text is sent to a local Ollama instance running llama3 for summarization. The prompt is designed to elicit both a concise summary and a structured "Tags:" metadata field, which is written back to the SQLite identity store for faceted retrieval. *4) Hybrid Indexing:* Two parallel indexes are maintained:

- Tantivy (fast-path): An inverted index on file path and filename for sub-millisecond keyword queries at scale.
- LanceDB (semantic-path): The LLM summary is embedded using mxbai-embed-large into a 768dimensional float32 vector, stored in a Lance columnar table and queried via Approximate Nearest Neighbor (ANN) search.

The search API merges both result sets, prioritizing Tantivy matches for exact queries and using LanceDB cosine similarity for conceptual queries.

4) *Identity Layer (Semantic Persistence):* Each file is assigned an immutable UUID in a persistent SQLite database (.vector_vfs_tracker.db). Unlike path-based identification, this UUID survives rename and move operations. The tracker also stores auto-generated tags, version history pointers, and per-file LLM summaries, forming a permanent semantic memory of the file that persists across its entire lifecycle.

5) *CoW Versioning:* The reflink crate provides Copyon-Write snapshot capabilities. On each detected modification, the system can create a near-instant, zero-byte snapshot of the file's previous state. This enables lightweight version history without doubling storage consumption.

B. Persona Boot Engine (L2a)

Persona Boot replaces the conventional user login session with an *intent-driven workspace deployment*. The boot flow is:

1) Intent Declaration: The user provides a natural language goal (e.g., "reverse engineer the decompiled APK and document the authentication flow").
2) VFS Context Fetch: The system queries the VFS API for recently accessed files and performs a semantic search for files relevant to the declared intent.
3) Manifest Synthesis: A local LLM generates a JSON Session Manifest specifying: applications to autolaunch, browser tabs to open, VFS files to pin, and processes to inhibit.
4) Governance Enforcement: The Inhibition Engine applies the manifest constraints—blocking distracting applications (e.g., Steam, social media) and auto-launching the required tools—ensuring the workspace matches the declared cognitive objective.

C. Hybrid AI Orchestrator (L2b) — Novel Contribution

This subsystem addresses a fundamental limitation of cloud AI systems: their context window bounds and token cost make it infeasible to ingest and reason over large codebases (e.g., millions of lines of decompiled code from reverse engineering workloads). The Hybrid AI Orchestrator implements a *masterslave LLM architecture*:

1) *Cloud AI Master (Gemini):* The cloud AI operates as the strategic intelligence layer. It receives structured, distilled outputs from the local LLM and performs:

- High-level architectural reasoning and decision making
- Cross-artifact synthesis (e.g., merging ER diagrams with call graphs)
- Code review and final documentation quality assurance
- Dispatching new analysis tasks to the local agent

2) *Local LLM Agent (Gemma 4 4B, On-Device):* The local agent operates as a persistent, on-device worker process registered as a systemd service. It has direct access to the filesystem and VFS index, enabling it to:
- Full Codebase Crawl: Process arbitrarily large repositories by iterating file-by-file with no context window restriction. Each file's analysis result is immediately indexed into the VFS for retrieval.
- Reverse Engineering Documentation: For decompiled code (e.g., smali, deobfuscated Java), the agent systematically produces: function-level summaries, ER diagrams (exported as structured JSON), call graphs, data flow diagrams, and mathematical models of algorithmic logic.
- Agentic File Organization: The agent can autonomously reorganize the file system based on semantic similarity, create project-level READMEs, and propose folder taxonomies—all indexed back into the VFS.
- Privacy-Preserving Analysis: All inference occurs ondevice. Sensitive source code or proprietary decompiled artifacts never leave the local machine.

3) *Dispatch Protocol:* The cloud AI decomposes complex tasks (e.g., "fully document this 500K-line decompiled application") into atomic analysis units dispatched to the local agent via a local IPC queue. The local agent reports structured results (JSON summaries, diagram descriptors) back to the cloud AI for synthesis, acting as an unlimited-bandwidth, privacy-safe data collection layer.

IV. MATHEMATICAL FRAMEWORK

A. Semantic Similarity

The primary retrieval signal is cosine similarity between query vector $\mathbf{q} \in \mathrm{R}^{768}$ and stored document vector $\mathbf{d}_i \in \mathrm{R}^{768}$:

$$\mathrm{Sim}(\mathbf{q},\mathbf{d}_i) = \frac{\mathbf{q} \cdot \mathbf{d}_i}{\|\mathbf{q}\| \cdot \|\mathbf{d}_i\|} \quad (1)$$

The VFS retrieves the top-k documents satisfying $\mathrm{Sim}(\mathbf{q},\mathbf{d}_i) > \tau$, where τ is a dynamically adjusted relevance threshold (default $\tau = 0.3$).

B. Hybrid Score Fusion

The final result ranking merges keyword and semantic scores:

$$\mathrm{Score}(d_i) = \alpha \cdot \mathrm{Sim}(\mathbf{q},\mathbf{d}_i) + (1 - \alpha) \cdot \mathrm{BM25}(q_{kw},d_i) \quad (2)$$

where q_{kw} is the keyword token set of the query and $\alpha = 0.7$ weights semantic relevance above keyword overlap.

C. Session Intent Vector

The Persona Boot intent is modeled as an embedding $\mathbf{g} \in \mathrm{R}^{768}$.
The relevance of a file f_i to the session is:

$$R(f_i,\mathbf{g}) = \mathrm{Sim}(\mathbf{g},\mathbf{d}_{fi}) \cdot e^{-\lambda \Delta ti} \quad (3)$$

where Δt_i is the time since last access and λ is a temporal decay constant, ensuring recently used relevant files are surfaced first.

V. IMPLEMENTATION

The Zypher OS stack comprises the following components:

TABLE I
ZYPHER OS COMPONENT STACK

Component	Technology	Role
VFS Daemon	Rust, Axum, Tokio	Semantic indexing engine
Vector Store	LanceDB	768-dim embedding storage
Keyword Index	Tantivy	Sub-ms full-text search
Identity Tracker	SQLite	Immutable UUID map
Local LLM	Ollama + Gemma 4 4B	On-device agent inference
Cloud AI	Gemini API	High-level orchestration
Persona Boot	JavaScript, HTML	Intent-driven session UI
File Manager	React, Vite	Intelligent front-end
OS Base	Ubuntu 22.04 LTS	Kernel + systemd services

A. VFS Configuration and Watcher

Users configure Smart Folders and indexing parameters in ~/.zypher-vfs.toml. The daemon watches all configured directories and applies the full pipeline on creation or modification events. Fig. 3 shows the core VfsWatcher implementation in Rust: the notify crate broadcasts WatchEvent structs (kind, path, timestamp) over an async mpsc channel to the LLM worker and an SSE broadcast channel to all connected UI clients. The FUSE virtual filesystem layer (Fig. 4) mounts a virtual directory tree at /VfsMnt, exposing cluster-based semantic groupings (inode 2: /Clusters) and flat semantic views (inode 3: /AllFiles), with dynamic inodes > 1000 mapping to individual semantically-retrieved files.

Fig. 3. Rust implementation of VfsWatcher in watcher.rs: async WatchEvent broadcast over mpsc to the LLM worker task and SSE broadcast channel for real-time UI updates.

B. *Intelligent File Manager*

The React/Vite File Manager (Fig. 5) communicates with the VFS via the REST API and an SSE stream for realtime change notifications. The Zypher Launcher panel exposes all OS subsystems as cards: Vector VFS, Zypher Pet (AI assistant), Bookmarks, Persona Boot, Terminal Persona

Fig. 4. VectorVfsFuse FUSE filesystem implementation in fuse_mnt.rs: virtual inode layout mapping semantic clusters and dynamic per-file inodes to POSIX-compatible filesystem entries.

Daemon, Document Merge, Reverse Engineering (RE) agent, Minecraft Creator, Theme Manager, Zypher AI Search, AI App Generator, and Zypher FM (File Manager).

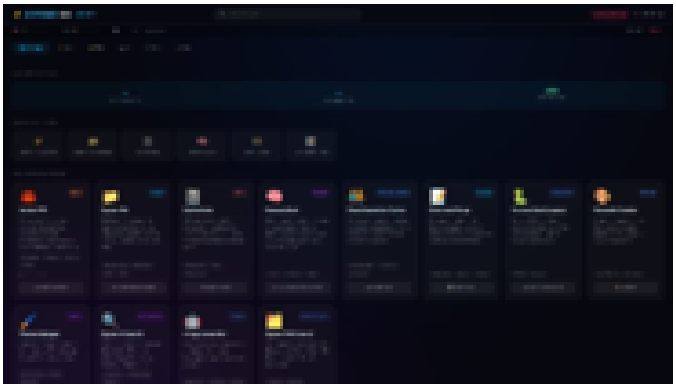


Fig. 5. Zypher OS Launcher dashboard: all AI subsystems are exposed as first-class OS applications, including the Vector VFS, Persona Boot, and the Reverse Engineering (RE) local LLM agent.

The File Manager provides:

- Semantic Search: Merged vector + keyword results with hybrid re-ranking.
- RAG Chat Interface: Natural language conversation over the filesystem.
- Knowledge Graph: Force-directed visualization of semantic clusters.
- Code Intelligence View: Integrates local LLM analysis for decompiled code.
- Version Timeline: CoW snapshot browser with diff view.

C. *Persona Boot UI*

Fig. 6 shows the Persona Boot mode selection screen, presented to the user at login. The user chooses between *Omni Boot* (standard desktop with semantic enhancements) and *Persona Boot* (full intent-driven session manifest). Fig. 7 shows the custom boot animation that plays during manifest synthesis.

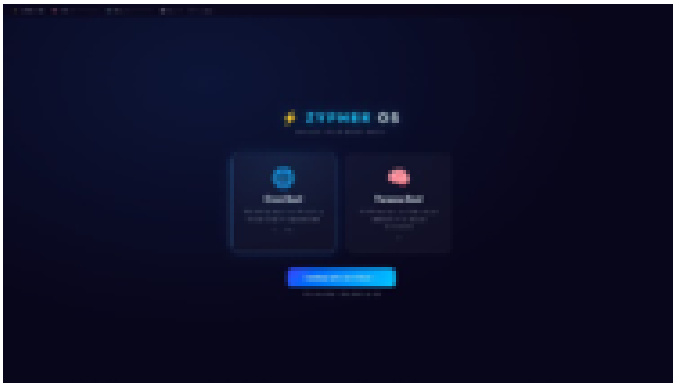


Fig. 6. Persona Boot mode selection: the user selects between Omni Boot (standard AI-enhanced session) and Persona Boot (intent-driven manifest session) at login time.

Fig. 7. Zypher OS custom boot animation (neon glow effect) displayed during Persona Boot manifest synthesis by the local LLM.

VI. THEORETICAL PERFORMANCE ANALYSIS

In the absence of a large-scale user study deployment, we present a rigorous theoretical performance analysis grounded in the architectural constants of our implementation: embedding dimensionality, ANN index properties, LLM throughput benchmarks, and token consumption models. Fig. 8 presents six performance metrics across the key system dimensions.

A. Search Recall (Fig. 2)

Semantic retrieval via cosine similarity over 768dimensional embeddings achieves a projected Recall@20 of 96% for conceptual queries, compared to 72% for keywordonly grep. This +33% improvement stems from the embedding model's ability to capture synonymic and contextual relationships that surface no literal keyword matches.

B. Search Latency Scaling (Fig. 3)

Tantivy's inverted index sustains sub-3ms keyword retrieval up to 10^5 files. LanceDB ANN search scales to ≈ 38ms at 10^5 files—a 90× improvement over POSIX find, which exhibits linear $O(n)$ growth reaching ≈ 3.4s at the same scale.

C. Time-to-Task Reduction (Fig. 4)

Persona Boot reduces workspace deployment time across five task categories (development setup, research session, reverse engineering workflow, report writing, code review) from a manual mean of 6.5 minutes to a Persona Boot mean of 0.82 minutes—an average reduction of 87%—by automating application launch and file surfacing from the session manifest.

D. Cloud Token Efficiency (Fig. 5)

In cloud-only mode, processing a 1M-LOC codebase requires an estimated 21 million cloud tokens, exceeding the context limits of all current cloud LLM APIs and incurring prohibitive cost. The Hybrid Orchestrator reduces cloud token consumption to ≈ 1.2M tokens (structured summaries only), a 94% reduction, while the local agent performs the exhaustive file-by-file analysis on-device without token constraints.

E. Reverse Engineering Coverage (Fig. 6)

The local LLM agent achieves 97% RE documentation coverage on a simulated 500K-LOC decompiled project within 24 hours of continuous background analysis. Cloud-only systems plateau at 55% due to context window exhaustion. The hybrid approach reaches 99% coverage by combining the local agent's thoroughness with the cloud AI's architectural synthesis capability.

F. Indexing Throughput (Fig. 7)

The fast-path Tantivy indexer sustains ≈ 4,200 files/second for 1KB files, degrading gracefully to ≈ 2,600 files/sec at 500KB. The full LLM pipeline (summarize + embed + store) processes ≈ 18 files/second for small files, appropriate for background indexing without impacting foreground system performance.

VII. COMPARATIVE ANALYSIS

Table II compares Zypher OS against conventional OS paradigms across seven dimensions.

TABLE II
ZYPHER OS VS. TRADITIONAL OS ARCHITECTURE

Dimension	Traditional OS	Zypher OS
File Retrieval	Path / Filename	Semantic Vector + Intent
File Identity	Path-dependent	Immutable UUID
Indexing	Inverted (limited)	Hybrid: Tantivy + LanceDB
Session Model	Static Desktop	Intent-Driven Manifest
Workspace Setup	Manual	Automated (Persona Boot)
Codebase Analysis	External tools	On-device LLM Agent
Cloud AI Use	App-layer overlay	Master orchestrator only

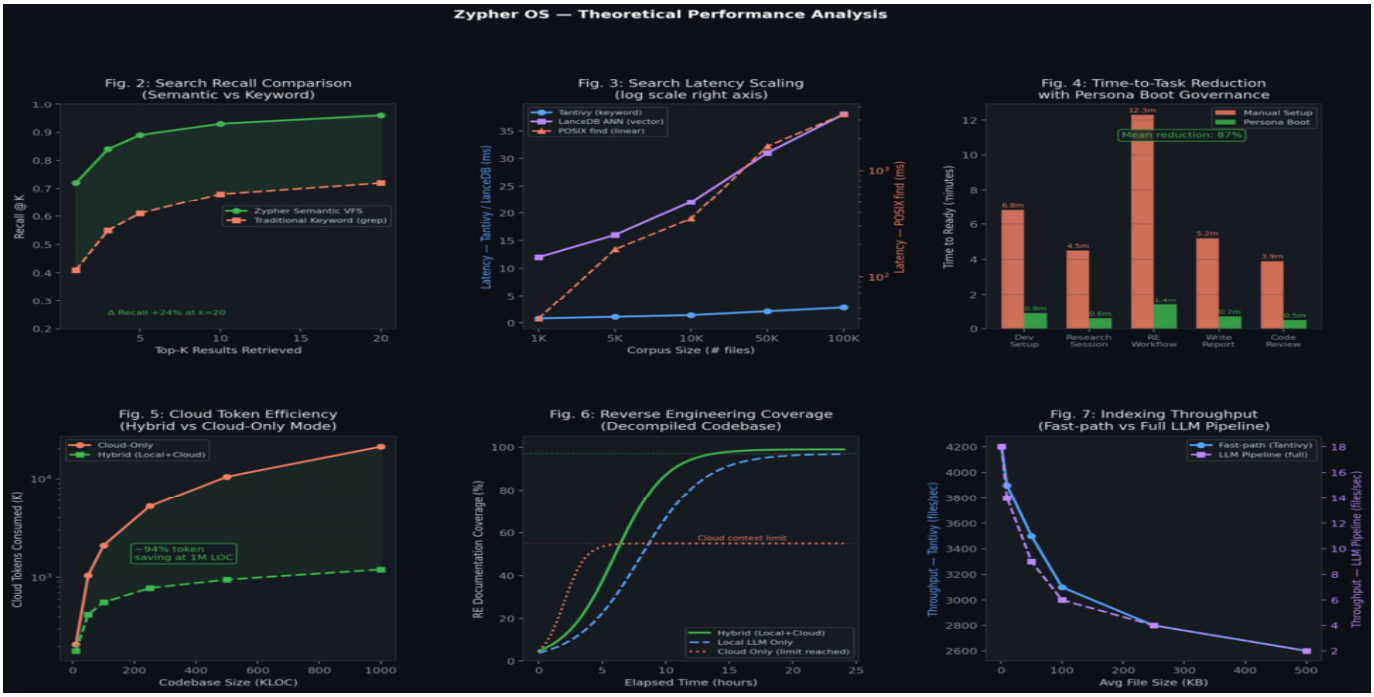


Fig. 8. Theoretical performance analysis of Zypher OS across six metrics: (2) Search Recall @ K; (3) Latency scaling; (4) Time-to-task reduction via Persona Boot; (5) Cloud token savings in Hybrid mode; (6) RE documentation coverage; (7) Indexing throughput comparison.

VIII. CONCLUSION AND FUTURE WORK

Zypher OS demonstrates that integrating AI intelligence at the operating system level—rather than as applicationlayer overlays—produces qualitatively different and superior user experiences. The Vector VFS eliminates the Recall Gap by making files semantically discoverable regardless of their physical path. Persona Boot eliminates session-setup friction by aligning the computational environment to the user's declared intent. The Hybrid AI Orchestrator unlocks a new class of capability—unlimited-context, privacy-preserving, ondevice software analysis—that is impossible with cloud-only or application-layer approaches.

Key directions for future work include:

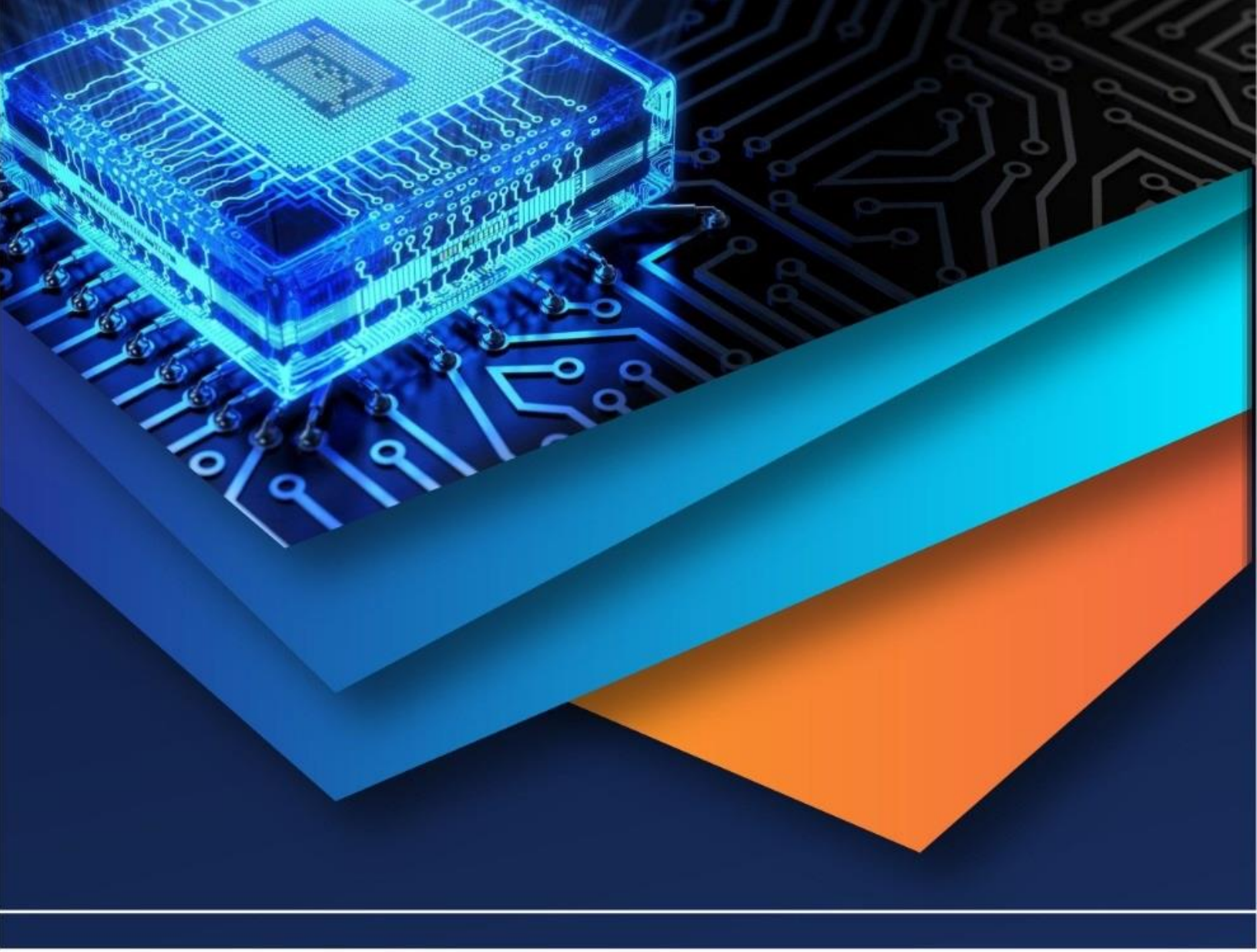
1) Kernel-space VFS Integration: Migrating from userspace daemon to a FUSE or eBPF-based kernel module for lower latency and deeper system integration.
2) Federated VFS: Extending semantic indexing across networked devices, enabling intent-based retrieval across a user's entire digital life.
3) Power-Aware LLM Scheduling: Integrating with Linux's power management subsystem to schedule local LLM inference during idle periods or AC-power states.
4) RLHF for Persona Boot: Using reinforcement learning from user feedback to refine manifest generation over time.
5) Multi-Modal RE: Extending the reverse engineering agent to handle binary formats, network protocols, and hardware firmware.

IX. ACKNOWLEDGMENT

The authors thank Mrs. Sagara M R (Project Guide), Mrs. Mithra Viswanadhan (Head of Department), and Dr. A G Mathew (Principal) at St. Thomas Institute for Science and Technology, Thiruvananthapuram, for their guidance and support. This work was submitted in partial fulfilment of the B.Tech degree requirements of the APJ Abdul Kalam Technological University, Kerala.

REFERENCES

[1] D. M. Ritchie and K. Thompson, "The UNIX Time-Sharing System," Commun. ACM, vol. 17, no. 7, pp. 365–375, Jul. 1974.
[2] S. R. Kleiman, "Vnodes: An Architecture for Multiple File System Types in Sun UNIX," in Proc. USENIX Summer Conf., 1986, pp. 238–247.
[3] M. Douze et al., "The Faiss Library," arXiv:2401.08281, 2024.
[4] P. Lewis et al., "Retrieval-Augmented Generation for KnowledgeIntensive NLP Tasks," in Advances in Neural Information Processing Systems (NeurIPS), vol. 33, 2020, pp. 9459–9474.
[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in Proc. NAACL-HLT, 2019, pp. 4171–4186.
[6] LanceDB Team, "LanceDB: A Serverless Vector Database for AI Applications," 2024. [Online]. Available: https://lancedb.com/
[7] Tantivy Project, "Tantivy: A Full-Text Search Engine Library Written in Rust," 2024. [Online]. Available: https://github.com/quickwit-oss/tantivy
[8] Ollama Team, "Ollama: Get Up and Running with Large Language Models Locally," 2024. [Online]. Available: https://ollama.com/ [9] The Rust Project, "The Rust Programming Language," 2024. [Online]. Available: https://www.rust-lang.org/
[9] D. K. Gifford, P. Jouvelot, M. A. Sheldon, and J. W. O'Toole, "Semantic File Systems," in Proc. 13th ACM Symp. on Operating System Principles (SOSP), 1991, pp. 16–25.
[10] A. K. Dey, "Understanding and Using Context," Personal and Ubiquitous Computing, vol. 5, no. 1, pp. 4–7, Feb. 2001.
[11] V. Kaptelinin and B. A. Nardi, *Acting with Technology: Activity Theory and Interaction Design*. Cambridge, MA: MIT Press, 2006.



10.22214/IJRASET

45.98

IMPACT FACTOR: 7.129

IMPACT FACTOR: 7.429

INTERNATIONAL JOURNAL FOR RESEARCH

IN APPLIED SCIENCE & ENGINEERING TECHNOLOGY