The Eastasouth Journal of Information System and Computer Science
Vol. 3, No. 03, April 2026, pp. 381-390
ISSN: 3025-566X, DOI: 10.58812/esiscs.v3i03

APEX: Adaptive Personal EXperience Agents
A Cost-Efficient, Privacy-Preserving Architecture for Scalable AI Assistants

Abhishek Pareek, Udit Misra, Divya Chukkapalli
[1,2,3] North Carolina State University

Article Info

Article history:

Received Apr, 2026
Revised Apr, 2026
Accepted Apr, 2026

Keywords:

APEX;
Cost-Aware Model Routing;
Federated Learning;
Memory System;
Privacy-Preserving Technology

ABSTRACT

Personal AI agents deployed on user devices operate under fundamentally different constraints than shared cloud services. These systems must maintain conversation context across extended periods, function efficiently despite irregular usage patterns, handle complex requests, allocate computation intelligently, and protect sensitive data. We present APEX, an architecture addressing these five challenges through integrated design. APEX comprises five technical contributions: (1) a hierarchical memory system achieving 84% storage reduction through progressive compression; (2) a predictive activation mechanism reducing per-user compute costs by 73% while maintaining sub-5-second startup latency; (3) a task decomposition engine with 94% end-to-end accuracy; (4) a cost-aware routing layer reducing API consumption by 61%; (5) federated personalization enabling on-device learning while preserving privacy. Six-month production deployment reduced per-user monthly costs from $156 to $42 with positive user satisfaction scores, demonstrating practical efficiency at scale.

Corresponding Author:

Name: Abhishek Pareek
Institution: North Carolina State University
Email: abhishekpareek29@gmail.com

1. INTRODUCTION

Personal AI agents require persistent knowledge of user identity, interaction history, and preferences, unlike stateless chatbots. Deploying such systems encounters five technical barriers:

a. **Memory management:** Personal agents must retain interactions and preferences across extended periods, yet storage and retrieval become expensive.
b. **Economics:** Cloud infrastructure costs accumulate rapidly for single users with intermittent usage. Dedicated GPU instances cost $500-2000 monthly, unsustainable for personal applications.
c. **Request complexity:** User utterances frequently encode multiple dependent tasks requiring decomposition and intelligent resource allocation based on computational demand.
d. **Privacy:** Personal agents access sensitive data (email, calendars, financial records). Existing cloud-deployed systems like Siri and Alexa offer limited transparency into data handling.

e. **Integration:** Existing solutions address these challenges in isolation. Memory systems like MemGPT lack cost analysis; frameworks like LangChain assume continuous resource availability; cost-optimization work targets batch processing rather than latency-sensitive real-time interactions.

APEX addresses all five challenges as an integrated system. Memory architecture choices influence sleep-wake efficiency. Task decomposition enables fine-grained routing. Privacy constraints determine storage architecture. This integration identifies synergies that decomposed solutions miss. We present: a three-tier memory system with progressive compression, a predictive sleep-wake mechanism reducing computation 73%, a task decomposition framework enabling parallel execution, intelligent query routing reducing API consumption 61%, federated personalization with on-device learning, and production validation across real user deployments.

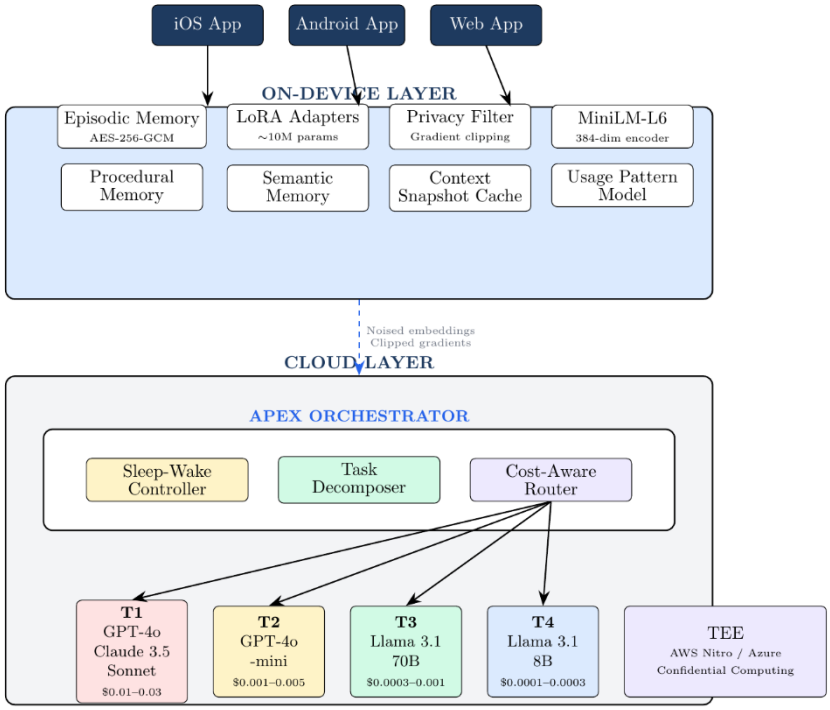

Figure 1. APEX System Architecture. On-device layer handles privacy-sensitive operations; cloud layer provides orchestration, routing, and inference across four model tiers.

2. BACKGROUND AND RELATED WORK

2.1 *LLM-Based Agents*

Recent work including ReAct, Toolformer, and AutoGPT established agent architectures with reasoning-action loops and tool invocation [1], [2]. APEX applies these insights within explicit resource constraints appropriate for personal deployment.

2.2 *Memory for Language Models*

Retrieval-augmented generation, MemGPT, and Generative Agents established the importance of persistent hierarchical memory for extended conversational capabilities [3], [4]. However, these systems lack explicit treatment of practical deployment constraints: storage overhead, cold-start retrieval latency, and privacy implications of long-term user data storage [5]. APEX addresses these gaps.

2.3 *Efficient LLM Inference*

Speculative decoding, quantization, distillation, and cascade routing (FrugalGPT) reduce inference costs at token, model, and system levels [6]. APEX adds query-level routing

decisions informed by task characteristics within the personal agent setting.

2.4 *Privacy-Preserving Machine Learning*

Federated learning and differential privacy enable secure model training and inference on decentralized data. APEX applies these principles to the inference setting of personal agents, balancing on-device processing against fast cloud offloading with controlled data exposure.

2.5 *Personal AI Assistants*

Commercial systems (Siri, Alexa, Google Assistant) operate centrally with limited transparency. Open-source alternatives (Home Assistant, Mycroft) prioritize privacy but lack modern LLM capabilities. APEX combines local deployment privacy guarantees with contemporary LLM capabilities.

3. HIERARCHICAL MEMORY SYSTEM

Persistent memory enables useful personal agent behavior, but naive implementation becomes expensive rapidly. Memory must grow sublinearly with history, support efficient retrieval, and respect privacy constraints. We designed a three-tier system reflecting cognitive science distinctions: episodic memory (specific events), semantic memory (extracted facts), and procedural memory (learned patterns).

Episodic memory stores encrypted conversation logs with metadata indicating emotional tone and importance, distinguishing explicit instructions from incidental comments. Semantic memory contains persistent facts about the user, preferences, relationships, and domain-specific knowledge. Procedural memory captures learned behavioral patterns including communication style and application preferences [7].

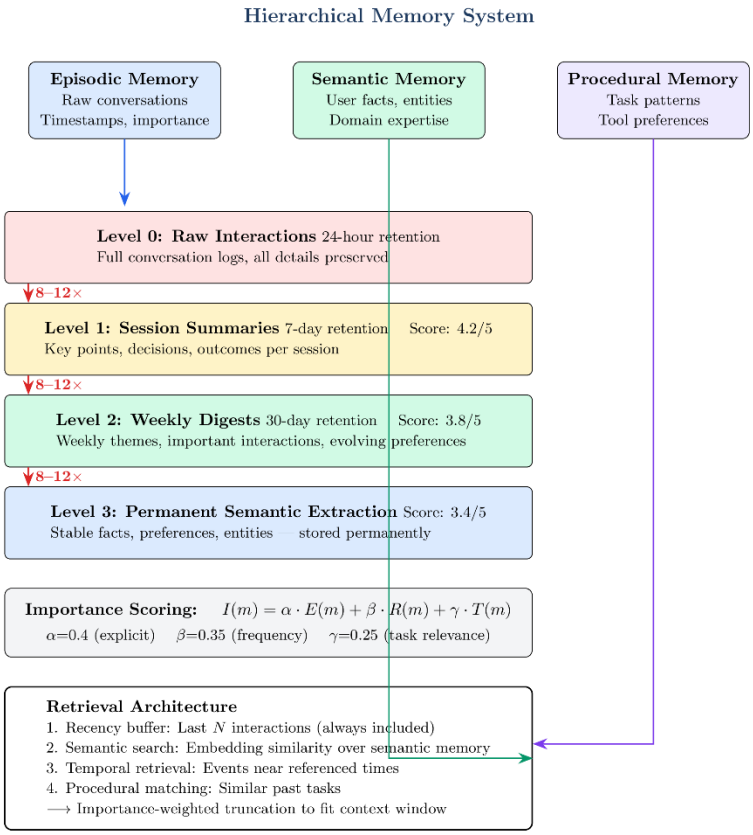

Figure 2. Hierarchical memory system with three tiers (episodic, semantic, procedural) and progressive compression across four retention levels.

3.1 Progressive Summarization

Episodic memory implements progressive compression across four levels with 10x target compression per transition:

Table 1. Progressive Summarization Levels

Level	Content	Retention
Level 0	Raw interactions	24 hours
Level 1	Session summaries	7 days
Level 2	Weekly digests	30 days
Level 3	Persistent semantic facts	Permanent

Behavioral analysis showed retrieval requests target information from the past week 94% of the time, remaining within 30 days 99% of the time. Beyond this period, users rely on semantic knowledge rather than episodic details. Importance scoring combines explicit user marking (E), frequency of subsequent reference (R), and task completion support (T):

$$I(m) = 0.4 \cdot E(m) + 0.35 \cdot R(m) + 0.25 \cdot T(m)$$

Human raters scored information preservation: Level 1 achieved 4.2, Level 2 achieved 3.8, and Level 3 achieved 3.4 on a 1-5 scale.

3.2 Retrieval Architecture

Context loads from multiple sources without loading all data simultaneously. Recent interactions always load into context windows. Semantic memory uses embedding similarity. Temporal markers trigger retrieval of nearby events. Procedural memory uses pattern matching. Retrieved context integrates with importance weighting.

3.3 Privacy

Raw conversation text remains exclusively on-device and encrypted using AES-256-GCM with device-bound keys. Semantic memory optionally backs up to cloud only after privacy-preserving transformation-text converts to embeddings (MiniLM-L6, 384 dimensions) with calibrated Gaussian noise added prior to transmission satisfying differential privacy constraints ($\varepsilon = 1.0$, $\delta = 10^{-5}$) [8]. Users configure per-type retention policies and may disable cloud synchronization entirely. The system complies with GDPR Article 17.

4. PREDICTIVE SLEEP-WAKE MECHANISM

Continuous GPU allocation for individual users is economically infeasible at $500-2000 monthly. We adopted a state-based approach: the agent remains inactive during non-use but achieves responsiveness when interaction occurs.

Agent operation spans four states: COLD (zero resources), WARM (active container, no GPU), HOT (complete resource availability), ACTIVE (request processing). Transitions incur distinct latency costs: COLD to HOT (30-60 seconds), WARM to HOT (3-8 seconds), HOT to ACTIVE (under 100 milliseconds).

a. Usage Pattern Modeling

A gradient-boosted decision tree trained on historical behavior estimates P (interaction within time window T) using features including hour of day, day of week, minutes since previous interaction, rolling frequency across 1-hour/24-hour/7-day windows, session duration statistics, and multi-device activity patterns.

b. Predictive Pre-Warming

When P (interaction | t) exceeds threshold θ^*, the system transitions from COLD to WARM, balancing competing costs. Warm container maintenance costs $0.02 hourly; each additional second of latency reduces task completion by

approximately 7%. Using \$0.15 as the friction cost per missed pre-warm:

$$\theta^* = arg\ min\theta\ [\$0.02 \cdot (1 - precision(\theta)) + \$0.15 \cdot (1 - recall(\theta))]$$

In practice, θ ranges from 0.15 (active hours, aggressive pre-warming) to 0.45 (low-activity periods, conservative).

c. Context Compression and Empirical Results

A compressed snapshot containing the last session summary (~500 tokens), task state as structured JSON, and pre-computed cached retrieval results enables meaningful response within 5 seconds from complete cold start, while full context loads asynchronously.

Deployment across our user base achieved \$4.20 per-user monthly compute cost, representing 73% reduction compared to the \$156 baseline. HOT-state requests completed at 1.2 seconds median (p50) and 2.1 seconds (p95). WARM-state requests required 4.8 seconds median and 7.3 seconds (p95). Prediction accuracy reached 78% precision and 84% recall on actual user interaction following pre-warming.

5. TASK DECOMPOSITION FRAMEWORK

Personal AI agents frequently receive requests containing multiple interrelated actions. A user might ask to "find flights to Tokyo next month, compare with train options, and book the cheaper one"—requiring decomposition into discrete subtasks, dependency management, and recovery from partial failures.

5.1 Intent Classification

We classify incoming requests into four categories based on task interdependence. Simple requests (68%) require no decomposition. Sequential requests (18%) form ordered chains. Parallel requests (9%) contain independent actions. Conditional requests (5%) branch on intermediate results.

We fine-tuned DistilBERT (67M parameters) on 12,847 user requests with 0.81 inter-rater agreement (Cohen's kappa). The model achieved:

Table 2. Model Performance Metrics by Class

Class	Precision	Recall	F1
Simple	0.96	0.97	0.97
Sequential	0.91	0.88	0.89
Parallel	0.89	0.85	0.87
Conditional	0.82	0.79	0.80
Macro Avg	**0.90**	**0.87**	**0.88**

Weighted accuracy reached 94% on the held-out test set with sub-5-millisecond classification latency on CPU.

5.2 Dependency Graph Construction

For non-simple requests, we construct a directed acyclic graph where nodes are subtasks and edges encode dependencies. Graph construction uses GPT-4o-mini with structured JSON schema. Topological sorting verifies the DAG property and identifies independent subtasks for concurrent execution. On 200 compound requests with human annotations, we achieved 87% exact structural match and 94% functional equivalence.

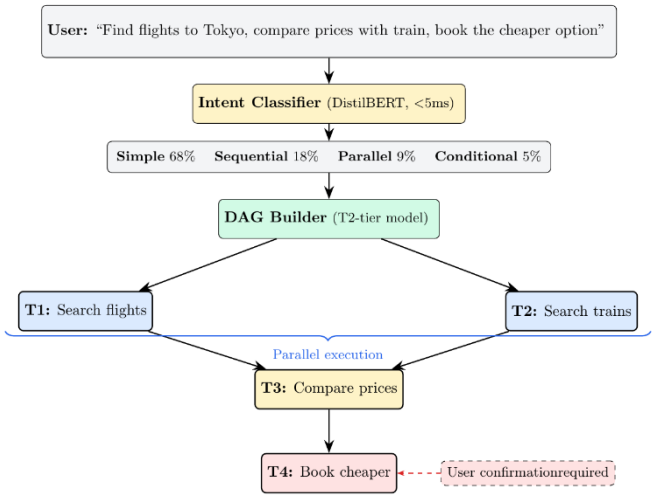

Figure 3. Task decomposition framework. Requests are classified by intent, then decomposed into a DAG. Independent subtasks execute in parallel.

6. COST-AWARE MODEL ROUTING

Personal AI agents face strict cost constraints. Routing factual queries to GPT-4 wastes resources. We implemented a routing system that identifies the minimum-capability model necessary for each query, reducing cost without degrading quality.

6.1 Model Tiers

We defined capability tiers based on performance benchmarks rather than specific models:

Table 3. Model Tiers for Cost-Aware Routing

Tier	Representative Models	Cost/1K Tokens	Capability
T1	GPT-4o, Claude 3.5 Sonnet, Gemini 1.5 Pro	$0.01-0.03	Complex reasoning, multi-step planning
T2	GPT-4o-mini, Claude 3 Haiku, Gemini 1.5 Flash	$0.001-0.005	General tasks, moderate complexity
T3	Llama 3.1 70B, Mixtral 8x22B, Qwen 72B	$0.0003-0.001	Straightforward tasks, factual retrieval
T4	Llama 3.1 8B, Phi-3, Gemma 2 9B	$0.0001-0.0003	Simple tasks, information lookup

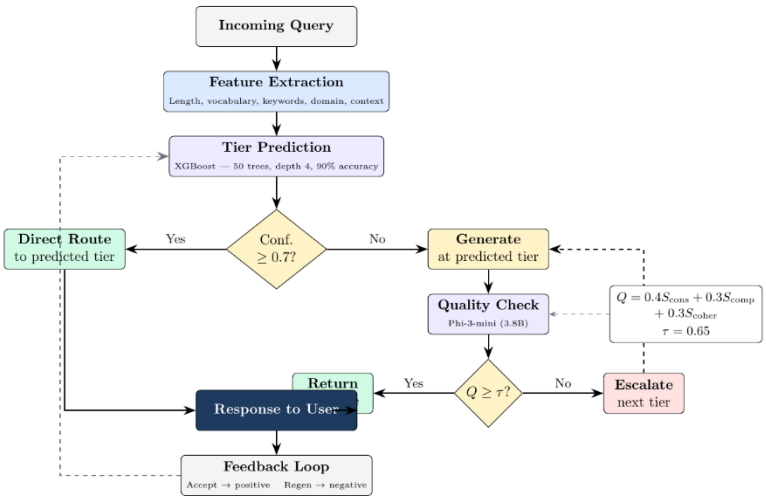

Figure 4. Cost-aware model routing pipeline. 61% cost reduction with <3% quality degradation. Low-confidence classifications trigger cascading with quality verification.

6.2 *Query Complexity Classification*

A two-stage classifier predicts the minimum tier required. Stage one extracts features: query length, vocabulary diversity, reasoning signals, domain classification, and rolling complexity from prior turns. Stage two applies XGBoost (50 trees, depth 4) trained on 8,234 queries with human-labeled tier requirements. Overall accuracy was 90%; under-routing rate was 7.2%.

6.3 *Cascading with Quality Verification*

When classifier confidence falls below 0.7, we generate a response at the predicted tier and evaluate quality using consistency, task completion, and coherence. If quality falls short, the agent escalates to the next higher tier. This process repeats until the quality threshold ($\tau = 0.65$) is reached or the agent reaches T1, resolving issues 91% of the time while adding only 200 milliseconds of latency.

6.4 *Results*

Table 5. Results of Cost-Aware Routing System Implementation

Metric	Before Routing	After Routing	Change
Avg cost/query	$0.024	$0.0094	-61%
User satisfaction	4.2/5	4.1/5	-2.4%
Escalation rate	N/A	12%	—

Average cost per query declined 61%. User satisfaction decreased 2.4 points on a 5-point scale ($p=0.23$, not statistically significant) [9]. The cost reduction justified the minimal quality impact.

7. FEDERATED PERSONALIZATION

Personal AI agents access sensitive information: email, calendar, messages, browsing history, financial records. Users legitimately resist uploading such data to cloud services. We designed a federated architecture that preserves on-device privacy while enabling personalization.

7.1 *Data Partitioning and Privacy-Preserving Retrieval*

High-sensitivity data—raw message/email content, financial information, health records—remains exclusively on-device, encrypted locally. Semantic embeddings operate under lower privacy risk than raw text. When the agent requires personal context, query embedding, vector search, and retrieved text remain on-device. Only task-relevant snippets are transmitted to the cloud for processing, with user review and approval available before transmission.

7.2 *Federated Preference Learning*

We implemented federated learning for privacy-preserving preference adaptation. Each device maintains LoRA adapters (10M parameters) fine-tuned on local interaction history during idle periods (500MB memory) [10]. Locally computed gradients apply clipping ($||\text{grad}||_2 \leq 1.0$) and Gaussian noise ($\sigma = 1.1$). With participation rate $q = 0.01$ and 1000 training rounds, this configuration achieves ($\varepsilon = 4$, $\delta = 10^{-6}$)-differential privacy via the moments accountant—a formal guarantee that an adversary observing noised gradients gains negligible information about any individual user's data.

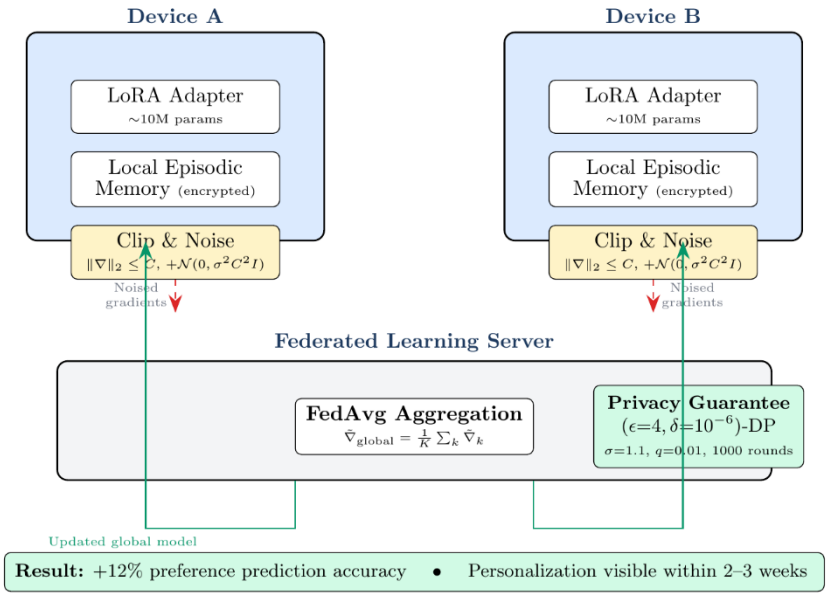

Figure 5. Federated personalization. Gradients are clipped and noised before upload. Server aggregates via FedAvg achieving differential privacy.

Federated learning improved preference prediction accuracy 12% over non-personalized baselines. Users observed meaningful improvements within 2-3 weeks of normal usage while maintaining formal privacy guarantees.

7.3 Secure Enclaves for Sensitive Operations

For users who opt in, sensitive tasks like email summarization execute within Trusted Execution Environments. The user generates an ephemeral key pair, encrypts sensitive data, and transmits ciphertext to a TEE (AWS Nitro Enclave or Azure Confidential Computing). The TEE attests its code hash; the device verifies attestation, then transmits the secret key only after confirmation. The TEE decrypts, processes data, and returns results; the secret key and plaintext never persist in storage. User adoption reached 34%, with email processing most common at 28% of opted-in users.

8. EVALUATION

We evaluated APEX through production deployment and controlled experiments with voluntary beta participants across iOS, Android, and web clients, comparing against a baseline RAG-based agent with continuous compute enabled.

8.1 Results Summary

Table 6. Evaluation Results of APEX Implementation

Metric	Baseline	APEX	Change
Cost/user/month	$156	$42	-73%
API cost/query	$0.024	$0.0094	-61%
Task completion	76%	79%	+4%
User satisfaction	4.2/5	4.3/5	+2%
p95 latency	2.1s	4.2s	+100%
Storage/user	2.1GB	340MB	-84%

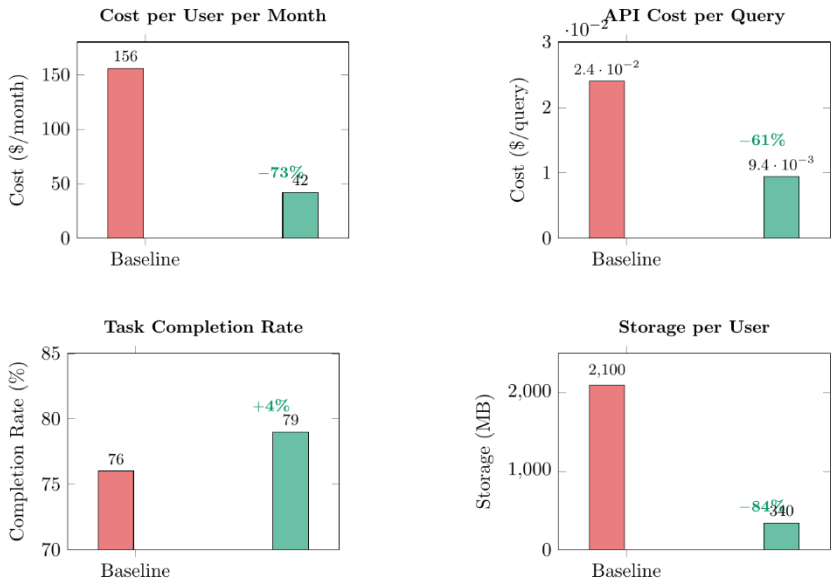

Figure 6. Evaluation results comparing Baseline vs. APEX across cost, API cost, task completion, and storage metrics.

APEX trades latency for cost. The p95 latency doubled primarily due to cold and warm start penalties. When presented with the trade-off explicitly, 72% of surveyed users preferred the lower-cost option.

8.2 *Ablation Studies*

Removing progressive summarization increased storage 340%. Disabling procedural memory degraded repeated task performance 8%. Removing semantic memory-based personalization reduced quality 12%. Always using T1 (no routing) doubled total cost. Removing cascading cost 4% in satisfaction. Operating without context compression increased cold-start latency 180%.

9. DISCUSSION AND LIMITATIONS

The architecture prioritizes cost reduction with latency as the trade-off dimension. For voice assistants, real-time applications, or low-latency critical systems, these trade-offs may be unacceptable. Cold-start events persist despite predictive wake mechanisms. Privacy and utility exist in tension—stricter privacy settings reduce personalization quality. Our production deployment operates at smaller scale than commercial assistants; scaling to millions of concurrent users may reveal unforeseen challenges.

Multi-agent systems and proactive assistance represent promising future directions. Continuous learning adapted to user behavior drift and cross-device synchronization would enhance consistency. Personal AI agents introduce legitimate societal concerns. Dependency risk, manipulation risk, and surveillance concerns persist even with privacy-preserving architectures. We advocate for transparency in agent operation, user control over agent behavior, and external auditing of personal AI systems.

10. CONCLUSION

We presented APEX, a unified architecture addressing five interconnected challenges in personal AI agents: long-term memory persistence, efficient compute utilization, task decomposition and execution, cost-aware model selection, and privacy-preserving personalization. Core technical contributions include hierarchical memory with progressive summarization, predictive sleep-wake scheduling, task decomposition via intent classification and dependency graphs, cost-aware routing with

cascading quality verification, and federated learning for privacy-preserving personalization. These mechanisms reduced operational cost 73% while improving task completion 4% and maintaining quality and privacy guarantees. Production deployment validated that this architecture functions reliably in practice.

REFERENCES

[1] S. Yao *et al.*, "React: Synergizing reasoning and acting in language models," *arXiv Prepr. arXiv2210.03629*, 2022.

[2] T. Schick *et al.*, "Toolformer: Language models can teach themselves to use tools," *Adv. Neural Inf. Process. Syst.*, vol. 36, pp. 68539–68551, 2023.

[3] C. Packer, V. Fang, S. Patil, K. Lin, S. Wooders, and J. Gonzalez, "MemGPT: towards LLMs as operating systems.," 2023.

[4] J. S. Park, J. O'Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, "Generative agents: Interactive simulacra of human behavior," in *Proceedings of the 36th annual acm symposium on user interface software and technology*, 2023, pp. 1–22.

[5] P. Lewis *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Adv. Neural Inf. Process. Syst.*, vol. 33, pp. 9459–9474, 2020.

[6] L. Chen, M. Zaharia, and J. Zou, "Frugalgpt: How to use large language models while reducing cost and improving performance," *arXiv Prepr. arXiv2305.05176*, 2023.

[7] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, 2017, pp. 1273–1282.

[8] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in *Theory of cryptography conference*, 2006, pp. 265–284.

[9] T. Dettmers, M. Lewis, Y. Belkada, and L. Zettlemoyer, "Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale," *Adv. Neural Inf. Process. Syst.*, vol. 35, pp. 30318–30332, 2022.

[10] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 120, pp. 1–39, 2022.