Contents

User Data Tiered Storage Architecture Proposal

Version: 1.0.0
Date: January 24, 2026
Status: PROPOSAL
Scale Target: 1M+ concurrent users

Executive Summary

This document evaluates whether to use Cortex's Hot/Warm/Cold tiered storage architecture for user data (chats, audits, results, uploads) versus dedicated services.

Recommendation: Create a **parallel User Data Service (UDS)** that mirrors Cortex's tiered architecture but is purpose-built for user-generated content with strict security isolation.

---

1. Data Categories Analysis

1.1 Chats (Conversations)

Metric	Value	Implication
Volume	~50 messages/user/day	50M messages/day at scale
Size	~2KB average per message	~100GB/day raw
Access Pattern	90% within 24h, 99% within 30 days	Perfect for tiering
Retention	7 years (compliance)	Cold tier essential
Security	User-private, PHI possible	Encryption + RLS mandatory

Tiered Fit: Excellent

1.2 Audit History

Metric	Value	Implication
Volume	~10 events/user/day	10M events/day
Size	~500 bytes average	~5GB/day
Access Pattern	<1% ever accessed	Cold-dominant
Retention	7 years (HIPAA/SOC2)	Immutable cold storage
Security	Tamper-proof, write-once	Merkle chain + glacier

Tiered Fit: Excellent (but Cold-dominant)

1.3 Results (AI Responses, Artifacts)

Metric	Value	Implication
Volume	~20 results/user/day	20M/day
Size	Variable (1KB - 1MB)	~200GB/day
Access Pattern	Often re-requested within session	Hot cache valuable
Retention	Tied to conversation	Follow chat lifecycle

Metric	Value	Implication
Security	User-private, may contain code	Sandboxed + encrypted

Tiered Fit: Good (follows conversation tier)

1.4 Uploads (Documents, Images)

Metric	Value	Implication
Volume	~2 uploads/user/week	300K/day
Size	100KB - 100MB	~3TB/day
Access Pattern	Burst on upload, then rare	S3 + metadata index
Retention	User-controlled	GDPR right to erasure
Security	User-private, virus scan needed	Isolated S3 + scanning

Tiered Fit: Moderate (S3 is natural home)

---

2. Architecture Options

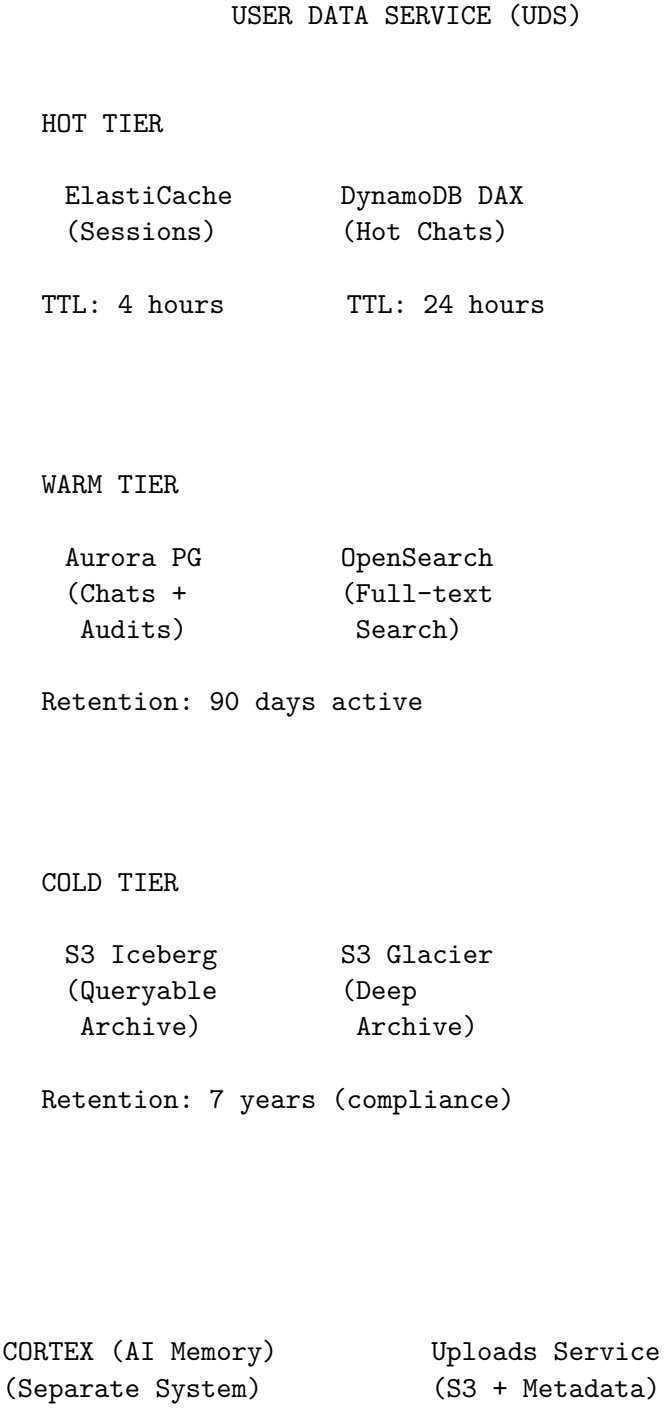
Option A: Extend Cortex for User Data

```
                    CORTEX SYSTEM

    HOT TIER (Redis ElastiCache)
    - AI Memory (current)
    - User Chats (NEW)
    - Session Results (NEW)


    WARM TIER (Neptune + PostgreSQL)
    - Knowledge Graph (current)
    - Chat History (NEW)
    - Audit Trail (NEW)


    COLD TIER (S3 Iceberg)
    - Archived Knowledge (current)
    - Archived Chats (NEW)
    - Compliance Archive (NEW)
    - Uploads (NEW)
```

Pros: - Single system to maintain - Unified tier coordination - Existing GDPR erasure flow

Cons: - Cortex is AI-memory focused, not user-data focused - Different security models (AI memory vs user PII) - Different access patterns (graph queries vs time-series) - Risk of coupling failures - Neptune cost increases significantly

Option B: Parallel User Data Service (Recommended)



Pros: - Purpose-built for user data - Separate security domain - Independent scaling - No Neptune dependency for chats - Cost optimized per data type

Cons: - Two systems to maintain - Need to sync GDPR erasure

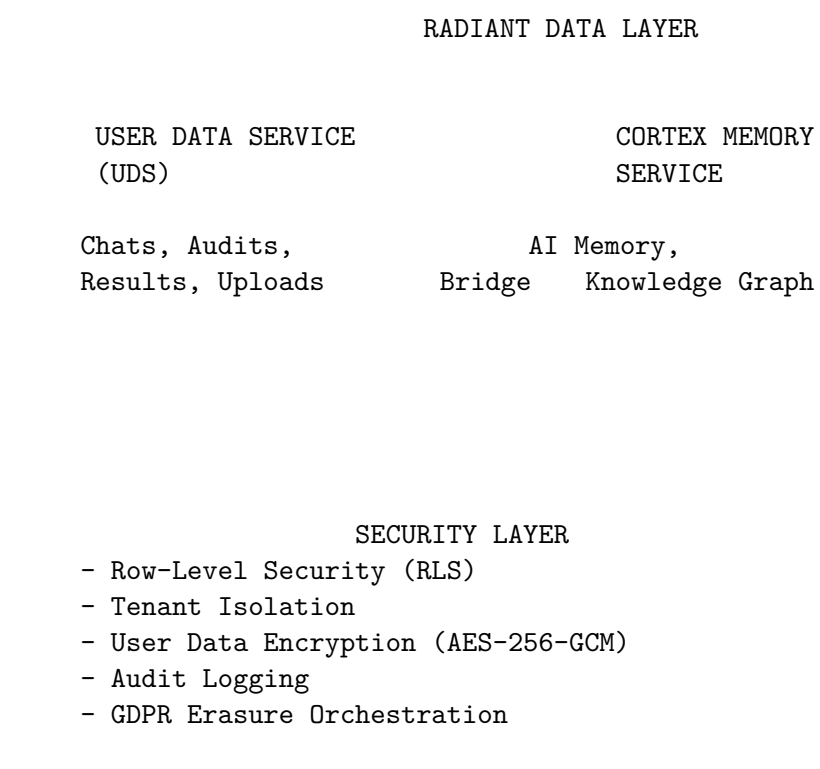
Option C: Hybrid (Cortex for AI, UDS for User)

Best of both worlds: - **Cortex** handles AI memory, knowledge graphs, semantic search - **UDS** handles user-generated content with security isolation - **Bridge Service** coordinates GDPR erasure, data lineage

---

3. Recommended Architecture: Option B + Bridge

3.1 System Overview



3.2 Hot Tier Design

```
// Hot Tier: Session-scoped data (TTL: 4 hours)
interface HotTierConfig {
  // ElastiCache for real-time sessions
  redis: {
    clusterMode: true;
```

```
    shards: 6;  // Scale with users
    replicasPerShard: 2;
    instanceType: 'r7g.xlarge';
    maxMemoryPolicy: 'volatile-lru';
  };

  // DynamoDB + DAX for recent chats
  dynamodb: {
    tableName: 'uds_hot_chats';
    gsi: ['tenant_user_index', 'conversation_index'];
    dax: {
      enabled: true;
      nodeType: 'dax.r6g.large';
      replicationFactor: 3;
    };
    ttl: {
      enabled: true;
      attributeName: 'expires_at';
      defaultHours: 24;
    };
  };
}
```

Hot Tier Data: | Data Type | Storage | TTL | Access Pattern | |———–|———|—–|————–| | Active Session | Redis | 4h | Every request | | Recent Messages | DynamoDB + DAX | 24h | Frequent reads | | Pending Results | Redis | 1h | Write once, read many |

3.3 Warm Tier Design

```
-- Warm Tier: Active data (0-90 days)

-- Conversations table with RLS
CREATE TABLE uds_conversations (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL REFERENCES tenants(id),
    user_id UUID NOT NULL REFERENCES users(id),

    -- Conversation metadata
    title VARCHAR(500),
    model_id VARCHAR(100),
    started_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    last_message_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    message_count INTEGER DEFAULT 0,
    total_tokens INTEGER DEFAULT 0,

    -- Forking support (Time Machine)
    parent_conversation_id UUID REFERENCES uds_conversations(id),
    fork_point_message_id UUID,
```

```
    branch_name VARCHAR(200),

    -- Encryption
    encryption_key_id VARCHAR(100) NOT NULL,

    -- Status
    status VARCHAR(20) DEFAULT 'active',
    archived_at TIMESTAMPTZ,

    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW(),
    updated_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

-- Messages with encryption
CREATE TABLE uds_messages (
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL,
    conversation_id UUID NOT NULL REFERENCES uds_conversations(id),
    user_id UUID NOT NULL,

    -- Message content (encrypted)
    role VARCHAR(20) NOT NULL CHECK (role IN ('system', 'user', 'assistant')),
    content_encrypted BYTEA NOT NULL,  -- AES-256-GCM encrypted
    content_iv BYTEA NOT NULL,         -- Initialization vector
    content_hash VARCHAR(64) NOT NULL, -- For deduplication

    -- Token usage
    input_tokens INTEGER,
    output_tokens INTEGER,
    model_id VARCHAR(100),

    -- Attachments
    attachment_ids UUID[] DEFAULT ARRAY[]::UUID[],

    -- Time Machine support
    sequence_number INTEGER NOT NULL,
    is_checkpoint BOOLEAN DEFAULT false,

    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

-- Partitioning by month for efficient archival
CREATE TABLE uds_messages_archive (
    LIKE uds_messages INCLUDING ALL
) PARTITION BY RANGE (created_at);

-- Audit log (append-only, tamper-evident)
CREATE TABLE uds_audit_log (
```

```
    id UUID PRIMARY KEY DEFAULT gen_random_uuid(),
    tenant_id UUID NOT NULL,
    user_id UUID,

    -- Event details
    event_type VARCHAR(100) NOT NULL,
    event_category VARCHAR(50) NOT NULL,
    resource_type VARCHAR(100),
    resource_id UUID,

    -- Change tracking
    previous_state_hash VARCHAR(64),
    new_state_hash VARCHAR(64),
    changes JSONB,

    -- Merkle chain for tamper evidence
    merkle_hash VARCHAR(64) NOT NULL,
    previous_merkle_hash VARCHAR(64),

    -- Context
    ip_address INET,
    user_agent TEXT,
    request_id VARCHAR(100),

    created_at TIMESTAMPTZ NOT NULL DEFAULT NOW()
);

-- Row-Level Security
ALTER TABLE uds_conversations ENABLE ROW LEVEL SECURITY;
ALTER TABLE uds_messages ENABLE ROW LEVEL SECURITY;
ALTER TABLE uds_audit_log ENABLE ROW LEVEL SECURITY;

CREATE POLICY tenant_isolation_conversations ON uds_conversations
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY user_isolation_conversations ON uds_conversations
    USING (user_id = current_setting('app.current_user_id')::UUID
           OR current_setting('app.is_admin')::BOOLEAN = true);

CREATE POLICY tenant_isolation_messages ON uds_messages
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID);

CREATE POLICY tenant_isolation_audit ON uds_audit_log
    USING (tenant_id = current_setting('app.current_tenant_id')::UUID
           AND current_setting('app.is_admin')::BOOLEAN = true);
```

3.4 Cold Tier Design

```
// Cold Tier: Archived data (90+ days)
interface ColdTierConfig {
  // S3 + Iceberg for queryable archive
  iceberg: {
    bucket: 'radiant-uds-archive-${region}';
    tableNamespace: 'uds_archive';
    tables: [
      'conversations',
      'messages',
      'audit_log'
    ];
    partitionBy: ['tenant_id', 'year', 'month'];
    format: 'parquet';
    compression: 'zstd';

    // Athena integration for queries
    athenaWorkgroup: 'uds-archive-queries';

    // Lifecycle
    storageClasses: [
      { ageRangeDays: { min: 90, max: 365 }, class: 'INTELLIGENT_TIERING' },
      { ageRangeDays: { min: 365, max: 2555 }, class: 'GLACIER_IR' },  // 7 years
      { ageRangeDays: { min: 2555, max: null }, class: 'DEEP_ARCHIVE' }
    ];
  };

  // Compliance archive (7-year retention)
  compliance: {
    bucket: 'radiant-uds-compliance-${region}';
    objectLock: true;  // WORM compliance
    retentionDays: 2555;  // 7 years
    encryptionKey: 'aws/kms/uds-compliance';
  };
}
```

3.5 Uploads Service

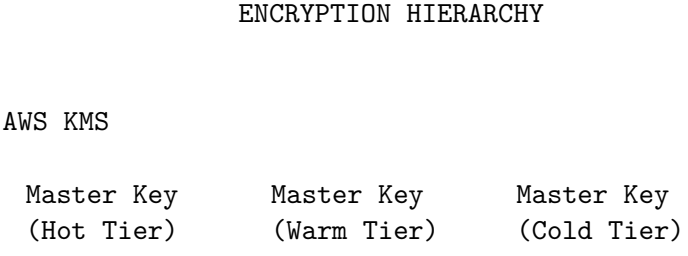
```
// Dedicated uploads handling (not in Cortex)
interface UploadsServiceConfig {
  // Quarantine bucket for virus scanning
  quarantine: {
    bucket: 'radiant-uploads-quarantine-${region}';
    maxSizeMb: 100;
    allowedTypes: ['pdf', 'docx', 'xlsx', 'csv', 'txt', 'png', 'jpg', 'gif'];
    virusScan: {
      enabled: true;
```

```
      engine: 'clamav-lambda';
      quarantineDays: 7;
    };
  };

  // Clean storage
  storage: {
    bucket: 'radiant-uploads-${region}';
    encryption: 'aws:kms';
    keyId: 'aws/kms/uds-uploads';

    // Per-user isolation via S3 prefixes
    pathPattern: '${tenant_id}/${user_id}/${upload_id}/${filename}';

    // Lifecycle
    storageClasses: [
      { ageRangeDays: { min: 0, max: 30 }, class: 'STANDARD' },
      { ageRangeDays: { min: 30, max: 180 }, class: 'INTELLIGENT_TIERING' },
      { ageRangeDays: { min: 180, max: null }, class: 'GLACIER_IR' }
    ];
  };

  // Metadata in PostgreSQL
  metadata: {
    table: 'uds_uploads';
    fullTextSearch: true;
    vectorEmbeddings: true;  // For semantic search
  };
}
```

---

4. Security Architecture

4.1 Encryption



```
Data Encryption Keys (DEKs)
- Per-tenant DEK
- Per-user DEK (optional, for high-security)
- Rotated every 90 days
```

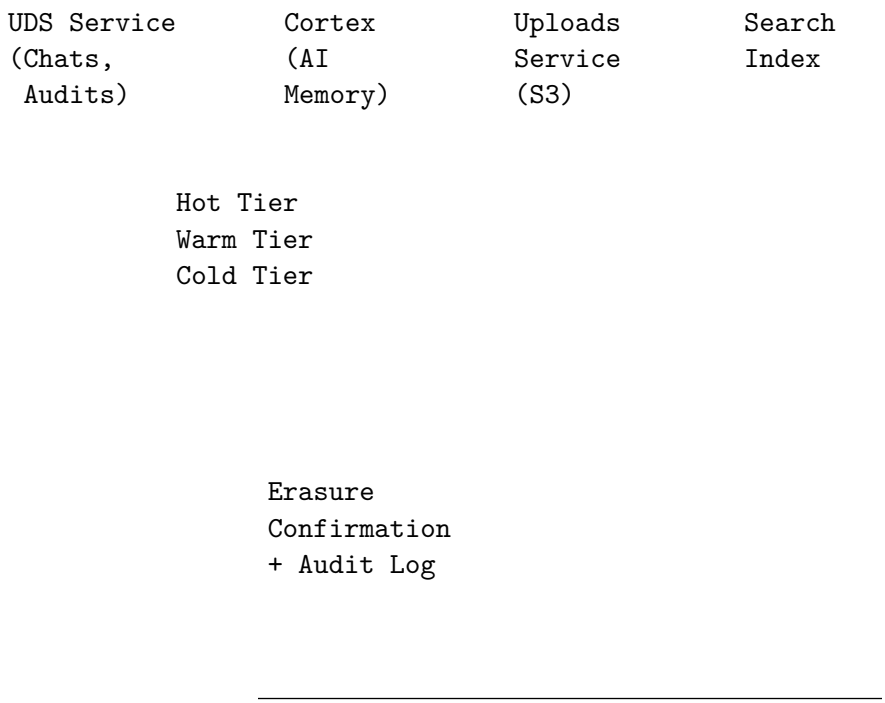
```
Encrypted Data
- Messages: AES-256-GCM with per-message IV
- Uploads: S3 SSE-KMS
- Audit: AES-256-GCM + Merkle chain
```

4.2 Access Control

```typescript
// Row-Level Security enforcement
interface SecurityContext {
  tenantId: string;       // Always required
  userId: string;         // User making request
  isAdmin: boolean;       // Tenant admin
  isPlatformAdmin: boolean;  // RADIANT admin
  permissions: string[]; // Fine-grained permissions
}

// Every database query runs with this context
async function setSecurityContext(ctx: SecurityContext): Promise<void> {
  await executeStatement(`
    SELECT set_config('app.current_tenant_id', $1, true);
    SELECT set_config('app.current_user_id', $2, true);
    SELECT set_config('app.is_admin', $3, true);
    SELECT set_config('app.is_platform_admin', $4, true);
  `, [ctx.tenantId, ctx.userId, ctx.isAdmin, ctx.isPlatformAdmin]);
}
```

4.3 GDPR Erasure Flow

```
                    GDPR ERASURE ORCHESTRATOR

   User Request


    Erasure Request
    Service
```



---

5. Cost Analysis (1M Users)

5.1 Option A: Extended Cortex

Component	Monthly Cost	Notes
Neptune Serverless	$8,000	Increased for chat graph
ElastiCache (larger)	$3,500	More shards for user data
Aurora PostgreSQL	$4,000	Larger instance
S3 + Glacier	$2,000	Combined storage
Total	**$17,500**	

5.2 Option B: Parallel UDS (Recommended)

Component	Monthly Cost	Notes
UDS Hot Tier		
- ElastiCache (6 shards)	$2,400	Sessions only
- DynamoDB + DAX	$1,800	On-demand + cache
UDS Warm Tier		
- Aurora PostgreSQL	$3,500	Optimized for time-series
- OpenSearch	$1,200	Full-text search
UDS Cold Tier		
- S3 + Iceberg	$1,500	Compressed archive

Component	Monthly Cost	Notes
- Glacier	$200	Deep archive
Cortex (unchanged)		
- Neptune Serverless	$3,000	AI memory only
- ElastiCache	$1,200	AI context only
Total	**$13,800**	**21% savings**

5.3 Cost Optimization

Optimization	Savings	Implementation
DynamoDB on-demand	30-50%	Pay per request
S3 Intelligent Tiering	40%	Automatic
Glacier IR vs Glacier	20%	Faster retrieval
Reserved capacity	30%	1-year commit

---

6. Implementation Plan

Phase 1: Foundation (Week 1-2)

- ☐ Create UDS database schema
- ☐ Implement encryption service
- ☐ Set up RLS policies
- ☐ Deploy DynamoDB tables

Phase 2: Hot Tier (Week 3-4)

- ☐ Implement session service
- ☐ Deploy DAX cluster
- ☐ Create hot-to-warm promotion job

Phase 3: Warm Tier (Week 5-6)

- ☐ Migrate existing chat data
- ☐ Implement full-text search
- ☐ Create warm-to-cold archival job

Phase 4: Cold Tier (Week 7-8)

- ☐ Set up Iceberg tables
- ☐ Configure lifecycle policies
- ☐ Implement Athena queries

Phase 5: Security & Compliance (Week 9-10)

- ☐ GDPR erasure orchestrator
- ☐ Merkle chain for audit

☐ Security testing

Phase 6: Integration (Week 11-12)

☐ Bridge with Cortex
☐ Admin dashboard
☐ Documentation

---

7. Recommendation Summary

Question	Answer
Use Cortex for user data?	No - create parallel UDS
Same tiered approach?	Yes - Hot/Warm/Cold pattern works
Share infrastructure?	No - separate for security
Share GDPR process?	Yes - orchestrated erasure

Key Principles

1. **Separate Concerns**: AI memory User data
2. **Security First**: Encryption + RLS + Isolation
3. **Cost Optimized**: Right storage for each access pattern
4. **Compliance Ready**: 7-year retention, audit trail
5. **Scale Ready**: Designed for 1M+ users day one

---

8. Next Steps

1. **Review this proposal** with the team
2. **Approve architecture** option (recommend B)
3. **Create detailed schema** for UDS tables
4. **Estimate infrastructure costs** more precisely
5. **Begin Phase 1** implementation

Document Author: Cascade AI
Review Required: Architecture Team