

# Social Media System Design - Progressive Stages

## Stage 2: Search with Elasticsearch

### Architecture Overview

Added full-text search capability with Elasticsearch, migrated to NoSQL, and implemented event-driven indexing.



### Data Flow - Create Post with Indexing

```
sequenceDiagram
    participant C as Client
    participant A as API Server
    participant M as MongoDB
    participant K as Kafka
    participant I as Indexer Service
    participant E as Elasticsearch

    Note over C,E: Create Post Flow
    C->>A: POST /api/posts
    A->>M: Insert Post Document
```

```

M-->>A: Success (post_id)

par Async Indexing
    A->>K: Publish "post.created" Event
    K->>I: Consume Event
    I->>E: Index Post Document
    E-->>I: Indexed
end

A-->>C: 201 Created

```

### Data Flow - Search Posts

```

sequenceDiagram
    participant C as Client
    participant A as API Server
    participant E as Elasticsearch
    participant M as MongoDB

    Note over C,M: Search Flow
    C->>A: GET /api/posts/search?q=keyword
    A->>E: Query: match(content, keyword)
    E->>E: Rank by relevance,<br/>recency, votes
    E-->>A: Post IDs + Scores

    opt Fetch Full Details
        A->>M: Get Posts by IDs
        M-->>A: Post Documents
    end

    A-->>C: 200 OK + Search Results

```

### Reindexing Flow

```

sequenceDiagram
    participant Cron as Cron Job
    participant M as MongoDB
    participant E as Elasticsearch

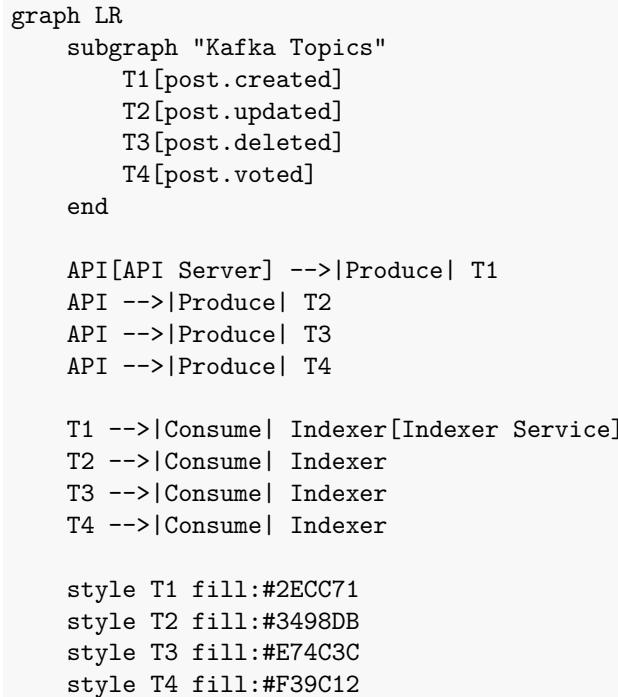
    Note over Cron,E: Scheduled Reindex (Daily 2 AM)
    Cron->>Cron: Trigger Reindex Job
    Cron->>M: Fetch All Posts (Batch)
    M-->>Cron: Posts Data

    loop For Each Batch
        Cron->>E: Bulk Index Documents
        E-->>Cron: Batch Indexed
    end

    Cron->>Cron: Log Completion

```

## Kafka Topics Structure

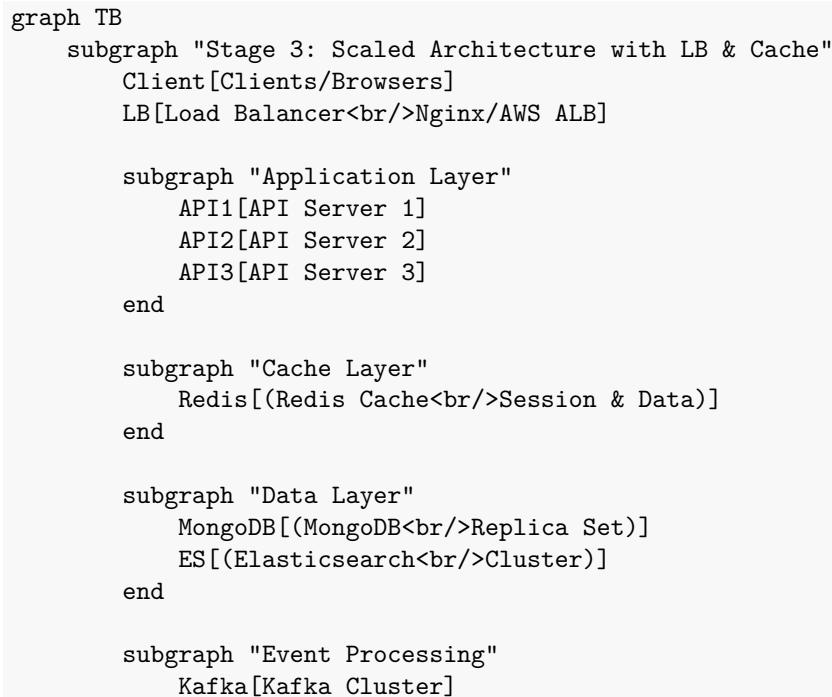


---

## Stage 3: Load Balancing & Caching

### Architecture Overview

Production-ready architecture with load balancing, caching layer, and horizontal scaling.



```

    Indexer1[Indexer 1]
    Indexer2[Indexer 2]
    Cron[Cron Service]
end

Client -->|HTTPS| LB
LB -->|Round Robin| API1
LB -->|Round Robin| API2
LB -->|Round Robin| API3

API1 -.->|Check Cache| Redis
API2 -.->|Check Cache| Redis
API3 -.->|Check Cache| Redis

API1 -->|Write/Read| MongoDB
API2 -->|Write/Read| MongoDB
API3 -->|Write/Read| MongoDB

API1 -->|Search| ES
API2 -->|Search| ES
API3 -->|Search| ES

API1 -->|Publish| Kafka
API2 -->|Publish| Kafka
API3 -->|Publish| Kafka

Kafka -->|Consume| Indexer1
Kafka -->|Consume| Indexer2
Indexer1 -->|Update| ES
Indexer2 -->|Update| ES

MongoDB -->|Read| Cron
Cron -->|Reindex| ES
end

style LB fill:#E67E22
style API1 fill:#4A90E2
style API2 fill:#4A90E2
style API3 fill:#4A90E2
style Redis fill:#DC382D
style MongoDB fill:#47A248
style ES fill:#FEC514
style Kafka fill:#231F20

```

### Detailed Request Flow with Cache

```

sequenceDiagram
    participant C as Client
    participant LB as Load Balancer
    participant A as API Server
    participant R as Redis Cache
    participant M as MongoDB
    participant E as Elasticsearch

```

Note over C,E: Get User Profile (Cache Hit)

C->>LB: GET /api/users/123

LB->>A: Forward Request

A->>R: GET user:123

R-->>A: Cache HIT (User Data)

A-->>LB: 200 OK + User Data

LB-->>C: Response

Note over C,E: Get User Profile (Cache Miss)

C->>LB: GET /api/users/456

LB->>A: Forward Request

A->>R: GET user:456

R-->>A: Cache MISS (null)

A->>M: Find user by ID

M-->>A: User Document

A->>R: SET user:456 (TTL: 1h)

R-->>A: OK

A-->>LB: 200 OK + User Data

LB-->>C: Response

Note over C,E: Search Posts (with Cache)

C->>LB: GET /api/posts/search?q=tech

LB->>A: Forward Request

A->>R: GET search:tech:page:1

R-->>A: Cache MISS

A->>E: Search Query

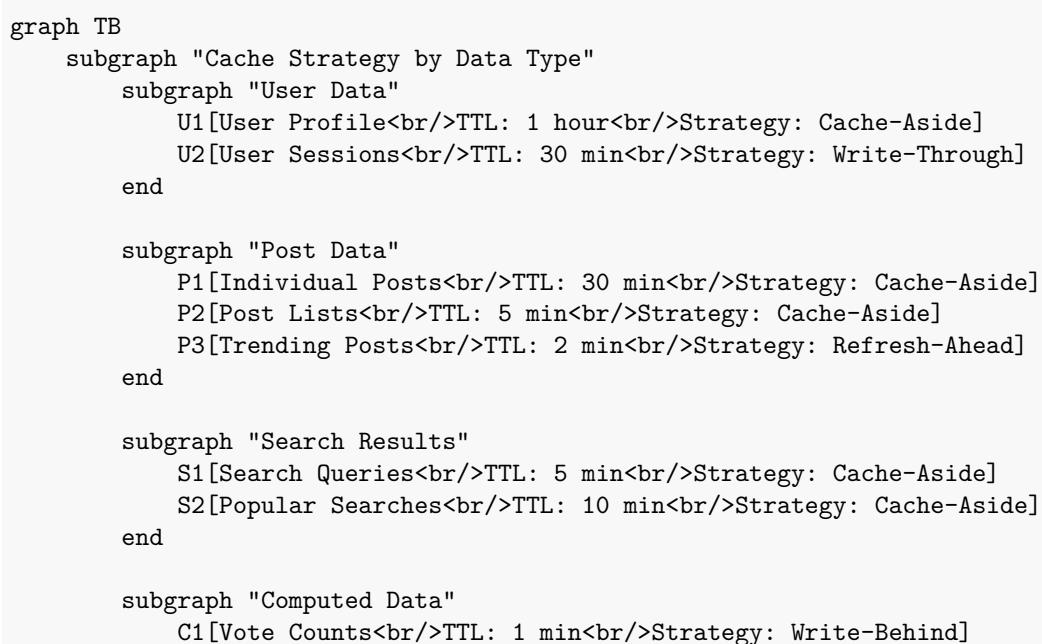
E-->>A: Search Results

A->>R: SET search:tech:page:1 (TTL: 5m)

A-->>LB: 200 OK + Results

LB-->>C: Response

## Cache Strategy Details



```

    C2[User Stats<br/>TTL: 15 min<br/>Strategy: Cache-Aside]
end
end

style U1 fill:#3498DB
style U2 fill:#3498DB
style P1 fill:#2ECC71
style P2 fill:#2ECC71
style P3 fill:#2ECC71
style S1 fill:#F39C12
style S2 fill:#F39C12
style C1 fill:#9B59B6
style C2 fill:#9B59B6

```

## Load Balancer Configuration

```

graph LR
    subgraph "Load Balancer Strategies"
        LB[Load Balancer]

        subgraph "Routing Rules"
            R1[/api/users/* → Round Robin]
            R2[/api/posts/* → Round Robin]
            R3[/api/search/* → Least Connections]
            R4[/api/uploads/* → IP Hash]
        end

        subgraph "Health Checks"
            H1[GET /health every 10s]
            H2[Timeout: 5s]
            H3[Unhealthy threshold: 3]
        end

        LB --> R1
        LB --> R2
        LB --> R3
        LB --> R4
        LB -.->|Monitor| H1
        LB -.->|Monitor| H2
        LB -.->|Monitor| H3
    end

```

## Write Operation with Cache Invalidation

```

sequenceDiagram
    participant C as Client
    participant LB as Load Balancer
    participant A as API Server
    participant R as Redis
    participant M as MongoDB
    participant K as Kafka

    Note over C,K: Update Post Flow

```

```

C->>LB: PUT /api/posts/123
LB->>A: Forward Request

par Database Update
  A->>M: Update Post Document
  M-->>A: Success
and Cache Invalidation
  A->>R: DELETE post:123
  A->>R: DELETE post_list:*
  R-->>A: Keys Deleted
and Event Publishing
  A->>K: Publish "post.updated"
  K-->>A: Acknowledged
end

A-->>LB: 200 OK
LB-->>C: Response

Note over K: Async: Indexer will update Elasticsearch

```

## Complete Data Flow

```

flowchart TB
  Start([Client Request]) --> LB{Load Balancer}

  LB -->|Route| API[API Server]

  API --> CheckCache{Check Redis Cache}
  CheckCache -->|HIT| ReturnCache[Return Cached Data]
  CheckCache -->|MISS| CheckType{Request Type}

  CheckType -->|Search| ES[Query Elasticsearch]
  CheckType -->|CRUD| DB[Query MongoDB]

  ES --> CacheResult[Cache Search Results]
  DB --> CacheData[Cache DB Results]

  CacheResult --> Return[Return Response]
  CacheData --> Return
  ReturnCache --> Return

  API -->|Write Op| PublishEvent[Publish to Kafka]
  PublishEvent --> Indexer[Indexer Service]
  Indexer --> UpdateES[Update Elasticsearch]

  API -->|Write Op| InvalidateCache[Invalidate Cache]

  Return --> End([Response to Client])

  style LB fill:#E67E22
  style API fill:#4A90E2
  style ES fill:#FEC514
  style DB fill:#47A248

```

```

style CheckCache fill:#DC382D
style PublishEvent fill:#231F20

```

---

## Comparison: Stage Evolution

### Architecture Complexity

```

graph LR
    subgraph "Stage 1"
        S1C[Client] --> S1A[API]
        S1A --> S1D[(DB)]
    end

    subgraph "Stage 2"
        S2C[Client] --> S2A[API]
        S2A --> S2M[(MongoDB)]
        S2A --> S2E[(Elasticsearch)]
        S2A --> S2K[Kafka]
        S2K --> S2I[Indexer]
        S2I --> S2E
    end

    subgraph "Stage 3"
        S3C[Client] --> S3L[LB]
        S3L --> S3A1[API-1]
        S3L --> S3A2[API-2]
        S3A1 --> S3R[(Redis)]
        S3A2 --> S3R
        S3A1 --> S3M[(MongoDB)]
        S3A2 --> S3M
        S3A1 --> S3E[(ES)]
        S3A2 --> S3E
    end

    style S1A fill:#4A90E2
    style S2A fill:#4A90E2
    style S3A1 fill:#4A90E2
    style S3A2 fill:#4A90E2
    style S3L fill:#E67E22
    style S3R fill:#DC382D

```

### Performance Metrics Comparison

```

graph TB
    subgraph "Metrics Evolution"
        direction TB

        M1[Stage 1<br/>---<br/>Response Time: 200ms<br/>Throughput: 100 req/s<br/>Availability: 95%]

        M2[Stage 2<br/>---<br/>Response Time: 150ms<br/>Search Time: 50ms<br/>Throughput: 300 req/s<br/>]

```

```

M3[Stage 3<br/>---<br/>Response Time: 50ms avg<br/>Search Time: 30ms<br/>Throughput: 2000 req/s

M1 -.->|Added Search| M2
M2 -.->|Added LB & Cache| M3
end

style M1 fill:#E74C3C
style M2 fill:#F39C12
style M3 fill:#2ECC71

```

---

## Key Features by Stage

Feature	Stage 1	Stage 2	Stage 3
<b>User CRUD</b>			
<b>Post CRUD</b>			
<b>Full-Text Search</b>			
<b>Search Ranking</b>		(Relevance, Recency, Votes)	
<b>Real-time Indexing</b>		(Kafka)	
<b>Batch Reindexing</b>		(Cron)	
<b>Load Balancing</b>			
<b>Caching</b>			(Redis)
<b>Horizontal Scaling</b>		Partial	
<b>High Availability</b>			

---

## Technology Stack Summary

Technology Stack			
Layer	Stage 1	Stage 2	Stage 3
Load Bal.	-	-	Nginx/ALB
API	Node.js	Node.js	Node.js (3x)
Cache	-	-	Redis
Database	PostgreSQL	MongoDB	MongoDB Cluster
Search	-	Elasticsearch	ES Cluster
Queue	-	Kafka	Kafka Cluster
Indexer	-	Node.js	Node.js (2x)
Scheduler	-	Cron	Cron Service