Cloudflare TinyURL - Design Document

Overview

Cloudflare TinyURL is a URL shortening service designed to efficiently generate short URLs, track their usage, and ensure data persistence with high scalability and fault tolerance.

Requirements

Functional Requirements

1. **Create Short URL**: Ensure uniqueness and prevent duplicates.
2. **Redirect Short URL**: Retrieve the original URL and redirect the request.
3. **Click Tracking**:
    - Track the number of clicks in different time frames: last 24 hours, last week, all time, and last minute.
    - Provide a fallback mechanism to query PostgreSQL when Redis is unavailable.
4. **Persistence**:
    - Ensure all URLs and click events are stored permanently in PostgreSQL.
    - Maintain Redis for fast lookup with persistence enabled.
5. **Delete Short URL**: Remove URLs upon request.
6. **Metrics and Logging**: Provide observability into system operations (optional but recommended).

Non-Functional Requirements

- **Scalability**: The system should be horizontally scalable.
- **Caching**: Use Redis to cache URLs for low-latency responses.
- **Event-Driven Architecture**: Process click events asynchronously.
- **Fault Tolerance**: Fallback to PostgreSQL when Redis is unavailable.
- **Testing**: End-to-end system tests to ensure reliability.

API Endpoints

Method	Endpoint	Description
POST	`/api/v1/create`	Create a short URL
GET	`/api/v1/{shortURL}`	Redirect to the original long URL

DELETE	`/api/v1/{shortURL}`	Delete a short URL
GET	`/api/v1/clicks/{shortURL}`	Retrieve click statistics from Redis
GET	`/api/v1/clicks_fallback/{shortURL}`	Retrieve click statistics from PostgreSQL (fallback)

System Architecture

Data Entities

PostgreSQL

1. **`urls` Table**: Stores URL mappings
    - `short_url_id` (Primary Key)
    - `long_url`
    - `created_time`
    - `expire_time`
2. **`clicks_urls` Table**: Stores all click events for fault tolerance
    - `id` (Primary Key)
    - `short_url`
    - `accessed_at`

Redis

1. **Global Counter**: Used to generate unique short URLs.
2. **Per URL Click Counters**:
    - `last_24hrs`, `last_week`, `all_time`, `last_1min` (with TTL)
3. **Redis Queue**:
    - Stores click events that expire to decrement counts asynchronously.

Key Design Considerations

1. **Caching for URL Redirection**:
    - Instead of querying PostgreSQL on every redirect, we cache URL mappings in Redis.
2. **Click Tracking via Redis Counters**:

    - Counting clicks in real-time using Redis counters.
    - Periodically persisting data in PostgreSQL for reliability.
    - Using Redis TTL key expiry events to queue updates for global counters.
3. **Fault Tolerance**:
    - If Redis goes down, click statistics can still be retrieved from PostgreSQL.
4. **Event-Based Updates**:
    - Redis queue is used to track click expiration events.
    - Events are processed asynchronously to decrement click counts.
5. **Distributed System Considerations**:
    - **Short URL Generation**: Base62 encoding + global counter.
    - **Click Events**: Use Snowflake ID for uniqueness in distributed environments.
    - **Horizontal Scalability**: The system can run multiple instances.

Testing Strategy

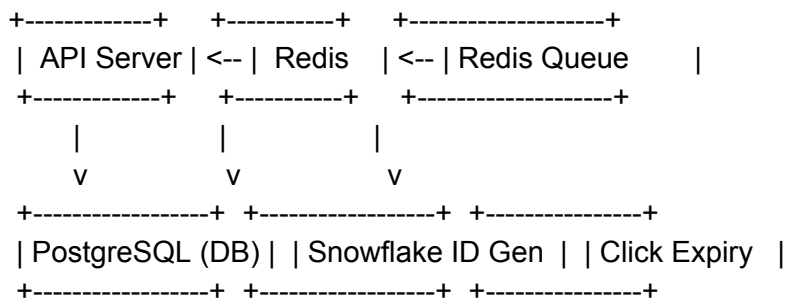
End-to-End Tests

1. **URL Uniqueness Test**: Create 10 URLs and ensure uniqueness.
2. **URL Deletion Test**: Create and delete 5 URLs, then verify they are removed.
3. **URL Redirection Test**: Create 10 URLs and validate their redirects.
4. **Click Counting Test**:
    - Create a URL and redirect it multiple times.
    - Validate click counts at different time intervals.
    - Ensure last-minute counts expire dynamically.

Future Enhancements

1. **Improved Logging and Monitoring**: Add distributed tracing and structured logs.
2. **Scalability Enhancements**: Perform load testing and optimize query performance.
3. **Advanced Event Processing**: Implement streaming solutions (e.g., Kafka) for click tracking.
4. **Security Enhancements**: Implement rate-limiting and authentication.

System Design Diagram



This design ensures efficient URL handling, caching, event-driven updates, and fault tolerance, making Cloudflare TinyURL a scalable and reliable system.