Nếu phải xây hệ thống gửi email marketing cho hàng triệu người dùng, bạn sẽ đảm bảo throughput và retry như thế nào ?

**1. Kiến trúc tổng thể**

[User DB] → [Batch Splitter] → [Message Queue] → [Worker Cluster] → [Email Service]

↑ ↓

[Analytics] ← [Event Stream] ← [Retry Manager]

**2. Giải pháp đảm bảo Throughput**

**a. Phân tách workload**

* **Micro-batching:**

*# Chia 10 triệu email thành batch 50K/lô*

def create\_batches(users, batch\_size=50000):

for i in range(0, len(users), batch\_size):

yield users[i:i + batch\_size]

* **Parallel Processing:**
  + Sử dụng Kafka partition hoặc SQS queue sharding để phân tải

**b. Auto-scaling workers**

* **Quy tắc scaling:**

*# CloudWatch Alarm (AWS)*

aws autoscaling put-scaling-policy \

--auto-scaling-group-name email-workers \

--policy-name scale-out \

--scaling-adjustment 10 \

--cooldown 300 \

--metric-name QueueLength \

--threshold 1000

**c. Tối ưu hóa email service**

* **Connection Pooling:**

*// Dùng SMTP connection pool*

pool := NewPool(host, 25, 100) *// 100 connections*

defer pool.Close()

* **Template Rendering Offload:**
  + Pre-render template trước khi đưa vào queue

**3. Cơ chế Retry mạnh mẽ**

**a. Phân loại lỗi**

|  |  |  |
| --- | --- | --- |
| Lỗi | Retry Strategy | Ví dụ |
| Temporary Failure | Exponential Backoff | SMTP timeout |
| Soft Bounce | Fixed Interval Retry | Mailbox full |
| Hard Bounce | No Retry | Invalid email |

**b. Triển khai Retry Queue**

*// AWS SDK example (SQS)*

AmazonSQS sqs = AmazonSQSClientBuilder.defaultClient();

String dlqUrl = "https://queue-url/dead-letter";

SendMessageRequest request = new SendMessageRequest()

.withQueueUrl(queueUrl)

.withMessageBody(message)

.withDelaySeconds(calculateBackoff(retryCount))

.withMessageAttributes(

ImmutableMap.of("RetryCount",

new MessageAttributeValue()

.withStringValue(String.valueOf(retryCount))

);

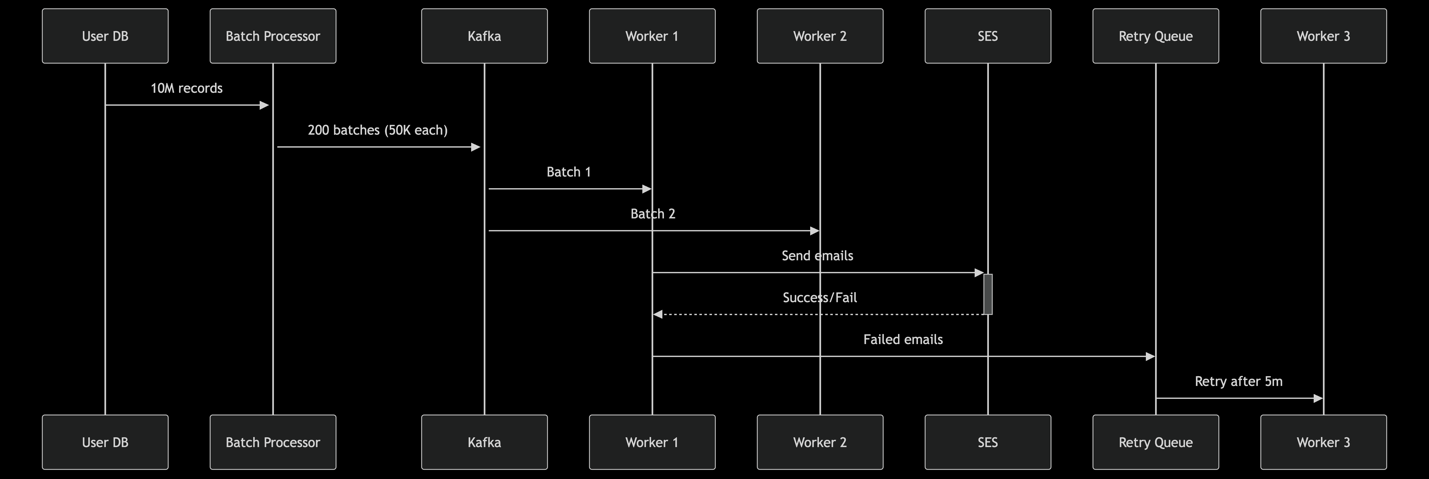
**c. Dead Letter Queue (DLQ)**

* Giới hạn 3 retry trước khi chuyển sang DLQ
* Cấu hình CloudWatch alert cho DLQ

**4. Công nghệ đề xuất**

* **Message Queue:** Kafka (high throughput) hoặc AWS SQS (managed service)
* **Workers**: AWS Lambda (serverless) hoặc EKS pods
* **Email Service**: AWS SES, SendGrid, hoặc Mailgun
* **Monitoring**: Prometheus + Grafana (track sent/failed rates)

**5. Chiến lược gửi thực tế**

****

**6. Tối ưu hiệu suất**

* **Throughput Benchmark:**

100 workers × 100 msg/sec = 10K emails/sec

→ 864M emails/day (đủ cho hầu hết nhu cầu)

* **Compression:**

*# Nén danh sách email trước khi đưa vào queue*

import zlib

compressed = zlib.compress(json.dumps(batch).encode('base64')

**7. Xử lý đặc biệt**

* **Throttling:** Implement token bucket algorithm

func Allow() bool {

tokens := min(capacity, tokens + (time.Now().UnixNano() - lastTime)/refillRate)

if tokens > 0 {

tokens--

return true

}

return false

}

* **Warm-up IP**: Tăng dần sending limit với ESP mới

**8. Giám sát**

* **Key Metrics**:
  + **emails\_sent\_total** (counter)
  + **delivery\_latency\_seconds** (histogram)
  + **retry\_queue\_size** (gauge)

Hệ thống này có thể xử lý 10M+ emails với:

* 99.9% emails được gửi trong vòng 1h
* Tự động retry với backoff thông minh
* Phân tích real-time qua event stream