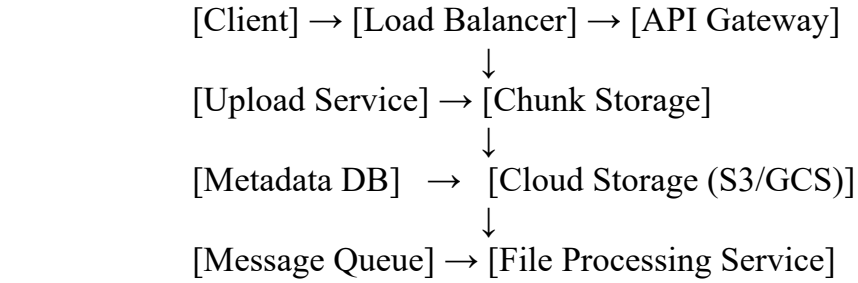
Hệ thống cần xử lý upload file tầm 500MB–1GB, bạn sẽ thiết kế thế nào để đảm bảo ổn định ?

1. Các thách thức chính

- **Network instability:** Kết nối mạng gián đoạn → Upload lại từ đầu.
- **High latency:** Thời gian upload lâu (ví dụ: 1GB với tốc độ 10Mbps ≈ 13 phút).
- **Server overload:** Xử lý đồng thời nhiều file lớn.
- **Data integrity:** Đảm bảo file không bị hỏng khi upload.

2. Kiến trúc tổng thể



3. Chi tiết từng thành phần

3.1. Client-Side (Web/Mobile)

- **Chunking:** Chia file thành các phần nhỏ (ví dụ: 10MB/chunk).
- **Resumable Upload:**
  - Lưu trạng thái upload (session ID, chunk uploaded) vào localStorage.
  - Retry tự động khi mất mạng.
- **Parallel Uploads:** Upload nhiều chunk đồng thời (tăng tốc độ).
- **Checksum:** Tính hash (SHA-256) cho từng chunk và cả file.

Ví dụ code (JavaScript):

```
const uploadFile = async (file) => {
  const chunkSize = 10 * 1024 * 1024; // 10MB
  const totalChunks = Math.ceil(file.size / chunkSize);
  const fileHash = await calculateSHA256(file);

  // Tạo session trên server
  const { sessionId } = await
api.startUploadSession(file.name, file.size, fileHash);

  // Upload từng chunk
  for (let i = 0; i < totalChunks; i++) {
    const chunk = file.slice(i * chunkSize, (i + 1) *
chunkSize);
    await api.uploadChunk(sessionId, i, chunk);
  }

  // Hoàn thành upload
  await api.completeUpload(sessionId);
};
```

3.2. Backend Services

a. Upload Service

- API Endpoints:
  - POST /upload/start: Khởi tạo session, trả về sessionId.
  - PUT /upload/{sessionId}/{chunkIndex}: Upload từng chunk.
  - POST /upload/complete/{sessionId}: Ghép các chunk thành file hoàn chỉnh.
- Xử lý chunk:
  - Lưu tạm chunk vào distributed storage (ví dụ: Redis, S3) thay vì server local.
  - Xác thực checksum từng chunk.
- State Management:
  - Lưu metadata vào database (PostgreSQL/MongoDB):

```
{
  "sessionId": "abc123",
  "status": "uploading",
  "chunksReceived": [0, 1, 3],
  "expiresAt": "2024-10-10T00:00:00Z"
}
```

b. File Processing Service

- **Xử lý hậu kỳ:**
  - Scan virus (integration với ClamAV).
  - Convert định dạng (nếu cần).
  - Extract metadata (EXIF, video duration).
- **Trigger qua Message Queue** (Kafka/RabbitMQ):

```
def process_file(sessionId):
    file = merge_chunks(sessionId)
    scan_virus(file)
    upload_to_cloud_storage(file)
    update_metadata_db(sessionId, "completed")
```

3.3. Cloud Storage & Database

- **Chunk Storage**: S3/MinIO (temporary bucket) → Tự động xóa sau 24h.
- **Final Storage**: S3/GCS với versioning enabled
- **Database**:
  - **Metadata**: PostgreSQL (quản lý session, user info).
  - **File Index**: Elasticsearch (tìm kiếm file theo metadata).

3.4. Xử lý lỗi & Độ tin cậy

- **Retry Mechanism**:
  - Client tự động retry khi chunk upload fail.
  - Giới hạn số lần retry (ví dụ: 3 lần).
- **Dead Letter Queue (DLQ):** Lưu trữ các task xử lý thất bại để phân tích sau.
- **Giám sát:**
  - Theo dõi qua Prometheus/Grafana:
  - Số lượng upload thành công/thất bại.
  - Thời gian trung bình upload.
  - Dung lượng lưu trữ.

4. Tối ưu hiệu suất

- **CDN:** CloudFront/Akamai cho file public (nếu cần phân phối).
- **Parallel Processing:**
  - Upload chunks qua nhiều kết nối.
  - Xử lý virus scanning song song.
- **Cold Storage:** Chuyển file không truy cập thường xuyên sang Glacier/Archive Storage.

5. Bảo mật

- **Authentication**: JWT/OAuth 2.0.
- **Quyền truy cập**:
  - S3 Pre-signed URLs (tạm thời) để upload/download.
  - IAM roles giới hạn quyền.
- **Mã hóa**:
  - TLS cho data in-transit.
  - AES-256 cho data at-rest.

6. Ví dụ thực tế

1. Flow hoạt động khi upload 1GB file:
2. Client chia file thành 100 chunk (10MB/chunk).
3. Khởi tạo session, nhận **sessionId**.
4. Upload 20 chunk đồng thời (parallel).
5. Mất mạng ở chunk 50 → Client tiếp tục từ chunk 50 khi kết nối lại.
6. Server ghép file và lưu vào S3.
7. Gửi notification qua email/webhook khi hoàn thành.

7. Công nghệ đề xuất

- **Cloud**: AWS (S3, EC2, Lambda) hoặc Google Cloud.
- **Database**: PostgreSQL + Redis (caching metadata).
- **Message Queue**: Kafka/RabbitMQ.
- **Monitoring**: Prometheus + Grafana.
- **File Scanning**: ClamAV + Quark Engine.

8. Tối ưu chi phí

Sử dụng **S3 Intelligent-Tiering** để tự động chuyển storage class.
Auto-scaling cho EC2 instances dựa trên CPU utilization.
Xóa các chunk không hoàn thành sau 24h.

Với thiết kế này, hệ thống có thể:

- Xử lý file lên đến hàng chục GB.
- Chịu được network interruption.
- Scale lên hàng ngàn user đồng thời.