# Almacenamiento de Documentos en XAMPP para Grandes Volúmenes

Para manejar más de 1,000,000 de archivos, incluyendo KMZ pesados, te presento las mejores opciones considerando rendimiento y escalabilidad:

## 1. \*\*Sistema Híbrido (Recomendado)\*\*

\*\*Estructura:\*\*

- \*\*Metadatos en MySQL\*\*: Nombre, tipo, tamaño, ruta, etc.

- \*\*Archivos en sistema de archivos\*\*: Organizados en subdirectorios.

\*\*Implementación:\*\*

```python

import os

from flask import Flask, request

import hashlib

UPLOAD\_FOLDER = 'C:/xampp/htdocs/uploads' # Fuera del htdocs para seguridad

SUBDIR\_LEVELS = 3 # Ej: 'a/b/c' para distribuir archivos

def get\_file\_path(filename):

# Crear estructura de subdirectorios basada en hash

filehash = hashlib.md5(filename.encode()).hexdigest()

path = os.path.join(UPLOAD\_FOLDER, \*[filehash[i] for i in range(SUBDIR\_LEVELS)])

os.makedirs(path, exist\_ok=True)

return os.path.join(path, filename)

@app.route('/upload', methods=['POST'])

def upload\_file():

file = request.files['file']

if file:

filename = secure\_filename(file.filename)

filepath = get\_file\_path(filename)

file.save(filepath)

# Guardar metadatos en MySQL

cursor = mysql.connection.cursor()

cursor.execute(

"INSERT INTO documentos (nombre, ruta, tamaño) VALUES (%s, %s, %s)",

(filename, filepath, os.path.getsize(filepath))

)

mysql.connection.commit()

```

\*\*Ventajas:\*\*

- Escalable para millones de archivos

- Fácil backup incremental

- Mejor rendimiento que BLOB para archivos grandes

## 2. \*\*Base de Datos con Tabla PARTITIONED\*\*

Para si realmente necesitas BLOB en MySQL:

```sql

CREATE TABLE documentos (

id BIGINT UNSIGNED NOT NULL AUTO\_INCREMENT,

nombre VARCHAR(255),

archivo LONGBLOB,

fecha\_upload TIMESTAMP,

PRIMARY KEY (id, fecha\_upload)

) PARTITION BY RANGE (UNIX\_TIMESTAMP(fecha\_upload)) (

PARTITION p2023 VALUES LESS THAN (UNIX\_TIMESTAMP('2024-01-01')),

PARTITION p2024 VALUES LESS THAN (UNIX\_TIMESTAMP('2025-01-01')),

PARTITION pmax VALUES LESS THAN MAXVALUE

);

```

## 3. \*\*Sistema de Archivos con Base de Datos NoSQL (Alternativa)\*\*

Usar MongoDB GridFS para archivos grandes:

```python

from pymongo import MongoClient

from gridfs import GridFS

client = MongoClient('localhost', 27017)

db = client['infra\_censo']

fs = GridFS(db)

# Almacenar archivo

with open('ruta.kmz', 'rb') as f:

fs.put(f, filename="proyecto.kmz", proyecto\_id=123)

# Recuperar

kmz\_file = fs.find\_one({"filename": "proyecto.kmz"})

```

## Comparativa de Métodos

| Método | Capacidad | Rendimiento KMZ | Backup | Complejidad |

|----------------------|----------------|-----------------|--------|------------|

| Híbrido (recomendado)| ★★★★★ (10M+ files) | ★★★★☆ | Fácil | Media |

| MySQL Partitioned | ★★★☆☆ (500K+) | ★★☆☆☆ | Complejo | Alta |

| MongoDB GridFS | ★★★★☆ (1M+) | ★★★★☆ | Medio | Media |

## Recomendación Final:

\*\*Para tu caso (1M+ archivos, KMZ grandes):\*\*

1. Usa el \*\*sistema híbrido\*\* con estructura de subdirectorios

2. Configura XAMPP con:

```ini

; En php.ini

upload\_max\_filesize = 512M

post\_max\_size = 520M

max\_execution\_time = 300

```

3. Implementa \*\*compresión\*\* para KMZ:

```python

import zipfile

# Comprimir antes de guardar

with zipfile.ZipFile('documento.kmz', 'r') as kmz:

kmz.extractall('temp\_folder')

# Procesar contenido y guardar comprimido

```

\*\*¿Necesitas que desarrolle más alguno de estos enfoques?\*\* Puedo proporcionarte:

- El script completo de gestión de archivos híbridos

- La configuración óptima de MySQL para BLOB grandes

- Solución de compresión automática para KMZ