

Optimización y Performance - Día 4

Change Status

pending 40 min

Learning Objectives

- 1 Entender conceptos básicos de optimización de performance
- 2 Aprender técnicas de mejora de velocidad en pipelines
- 3 Comprender monitoreo de recursos y cuellos de botella
- 4 Conocer estrategias de escalado horizontal y vertical

Theory	Practice	Evidence	Quiz
--------	----------	----------	------

Activities and Learning

Task 1: Identificación de Cuellos de Botella (10 minutos)

¿Dónde se producen los cuellos de botella?

Los pipelines pueden ser lentos en diferentes puntos:

- Ingesta:** Conexiones de red lentas
- Procesamiento:** Algoritmos ineficientes
- Almacenamiento:** Consultas no optimizadas
- I/O:** Lectura/escritura de disco

Cómo identificar problemas:

```
import time

def medir_performance(funcion, *args):
    """Mide tiempo de ejecución de una función"""
    inicio = time.time()
    resultado = funcion(*args)
    fin = time.time()

    tiempo_total = fin - inicio
    print(f"Tiempo de ejecución: {tiempo_total:.2f} segundos")

    return resultado, tiempo_total
```

Task 2: Técnicas de Optimización Básica (10 minutos)

Optimizaciones comunes:

- Paralelización:** Procesar datos en paralelo
- Caching:** Almacenar resultados intermedios
- Compresión:** Reducir tamaño de datos
- Indexación:** Acelerar búsquedas

Ejemplo de paralelización simple:

```
from concurrent.futures import ThreadPoolExecutor
import requests

def procesar_api_paralelo(urls):
    """Procesa múltiples APIs en paralelo"""
    with ThreadPoolExecutor(max_workers=4) as executor:
        resultados = list(executor.map(procesar_una_api, urls))
    return resultados
```

Task 3: Monitoreo y Alertas (10 minutos)

¿Qué monitorear en un pipeline?

- Performance:** Tiempo de ejecución, throughput
- Recursos:** CPU, memoria, disco
- Calidad:** Tasa de errores, completitud de datos
- Disponibilidad:** Uptime del sistema

Dashboard simple de métricas:



```
metricas_pipeline = {  
    'tiempo_ejecucion': '45 minutos',  
    'registros_procesados': 100000,  
    'tasa_error': '0.1%',  
    'uso_cpu': '65%',  
    'uso_memoria': '70%'  
}
```