Predicción de Consumo Energético en Ciudades Inteligentes

Proyecto de Procesamiento de Grandes Volúmenes de Datos

Autores:

Amalia Beatriz Valiente Hinojosa
Noel Pérez Calvo

1. ¿Qué se pretende lograr?

El objetivo central del proyecto es **predecir el consumo energético en distintas zonas de una ciudad inteligente**, utilizando técnicas de procesamiento de grandes volúmenes de datos mediante las plataformas *Hadoop* y *Apache Spark*. A través del análisis de datos históricos de consumo eléctrico, se busca anticipar la demanda en las próximas horas, detectar picos de consumo y evaluar el impacto de las condiciones climáticas sobre el uso energético urbano.

2. Dataset seleccionado

Nombre: *Household Electric Power Consumption Dataset*

Fuente: Kaggle / UCI Machine Learning Repository

Formato: CSV (valores separados por punto y coma ";")

URL: https://www.kaggle.com/datasets/uciml/electric-power-consumption-data-set

3. Justificación del dataset

Volumen

El conjunto de datos contiene más de **dos millones de registros**, correspondientes a mediciones minuto a minuto del consumo eléctrico de un hogar durante casi **cuatro años** (2006–2010). Su tamaño aproximado de 120 MB en formato txt, y el incremento que supone su almacenamiento distribuido en *HDFS*, permiten simular un entorno realista de **procesamiento de grandes volúmenes de datos**, adecuado para la aplicación de tecnologías como *Hadoop* y *Spark*.

Características

El dataset incluye variables como:

- Fecha y hora de cada registro.
- Potencia activa y reactiva global.
- Voltaje y corriente.
- Energía submedida en tres zonas del hogar (*Sub_metering_1*, *2* y *3*).

Estas variables conforman una **serie temporal multivariable**, adecuada para tareas de predicción y análisis de patrones de consumo. Además, los datos presentan una variabilidad natural y cierto nivel de ruido, lo que refleja condiciones reales de consumo y permite evaluar técnicas de limpieza y modelado robustas.

Pertinencia

El conjunto de datos resulta altamente pertinente para los objetivos del proyecto, ya que:

- Contiene **registros reales de consumo energético**, directamente relacionados con la meta de predecir la demanda eléctrica.
- Su granularidad temporal (minuto a minuto) facilita el análisis de **patrones horarios, diarios y estacionales**.
- Puede combinarse con datos meteorológicos (temperatura, humedad, precipitaciones) para analizar el **impacto del clima** en la demanda, fortaleciendo el enfoque de ciudades inteligentes.

Conclusión

El dataset seleccionado constituye una base sólida para el desarrollo del proyecto, ya que ofrece volumen, calidad y relevancia suficientes para implementar un sistema de predicción energética escalable mediante herramientas de procesamiento distribuido. De esta manera, se busca contribuir al diseño de estrategias de eficiencia y sostenibilidad energética en el contexto de las ciudades inteligentes.

4. Fase 1: Análisis y Definición del Productor de Datos (Producer)

En esta primera etapa del proyecto se diseña el flujo general del componente *Producer*, encargado de la generación y envío continuo de datos hacia el sistema distribuido basado en Hadoop. Este módulo simulará el flujo de información proveniente de sensores inteligentes instalados en una red eléctrica urbana.

4.1. Objetivo del Producer

El objetivo del *Producer* es emular la llegada de mediciones energéticas en tiempo real, transmitiendo registros de consumo eléctrico hacia un sistema de mensajería (por

ejemplo, Apache Kafka) que actúe como capa intermedia entre las fuentes de datos y el almacenamiento distribuido en HDFS.

De esta forma, el sistema completo podrá procesar, analizar y visualizar los datos de consumo en tiempo real, permitiendo detectar patrones de demanda, anomalías y picos de consumo en diferentes zonas de la ciudad.

4.2. Estructura de los Datos

Los datos utilizados provienen de un conjunto de mediciones de energía con las siguientes variables:

- **Date**: Fecha de la medición (formato `YYYY-MM-DD`).
- **Time**: Hora exacta de la medición (formato `HH:MM:SS`).
- **Global_active_power**: Potencia activa global (kW).
- **Global_reactive_power**: Potencia reactiva global (kW).
- **Voltage**: Voltaje promedio (V).
- **Global_intensity**: Intensidad de corriente (A).
- **Sub_metering_1**: Consumo energético parcial del área 1 (Wh).
- **Sub_metering_2**: Consumo energético parcial del área 2 (Wh).
- **Sub_metering_3**: Consumo energético parcial del área 3 (Wh).

Estas variables reflejan el comportamiento energético de una vivienda o conjunto de zonas urbanas a lo largo del tiempo y permiten realizar análisis tanto locales como agregados sobre la red eléctrica.

4.3. Frecuencia de Envío

El flujo de simulación definido enviará una lectura cada **0.5 segundos**, representando un ritmo razonable para la actualización continua de datos de consumo en una red inteligente. Cada mensaje incluirá un único registro (una fila del dataset) en formato JSON o CSV, con su respectivo sello temporal.

4.4. Flujo de Datos

El flujo del *Producer* seguirá las siguientes etapas:

1. **Lectura de datos**: el módulo leerá progresivamente las filas del dataset energético.
2. **Estructuración**: cada fila será convertida en un mensaje estructurado, conteniendo los campos definidos y un timestamp.
3. **Emisión periódica**: cada 5 segundos, el *Producer* enviará el siguiente mensaje al sistema de mensajería (Kafka).
4. **Transporte distribuido**: los mensajes se propagarán a través de los tópicos de Kafka, permitiendo su consumo por distintos módulos del sistema (almacenamiento en HDFS, análisis en Spark, visualización, etc.).

4.5. Diagrama del Flujo de Datos del Producer

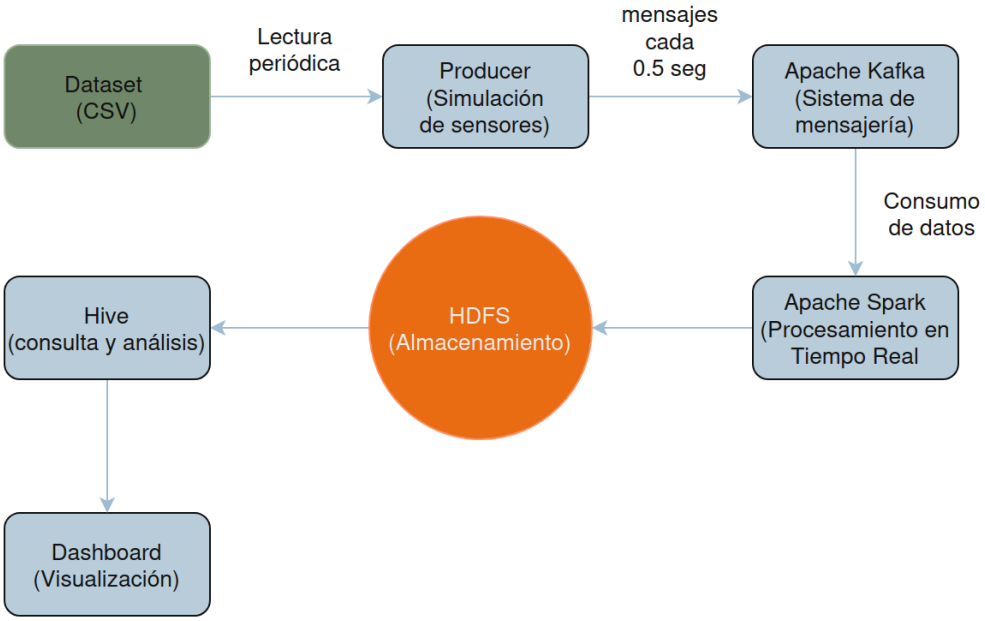

Figura 1: Flujo general del Producer dentro del sistema distribuido basado en Hadoop.

5. Fase 2: Implementación del Entorno Distribuido con Docker

En esta fase se procede al diseño e implementación del entorno de ejecución distribuido que permitirá la interacción entre los distintos componentes del sistema. Para lograr un entorno reproducible y escalable, se utiliza **Docker** como herramienta de contenedorización y orquestación.

5.1. Objetivo de la Fase

El objetivo principal de esta fase es levantar un entorno funcional que permita la comunicación entre el *Producer* y el sistema de mensajería **Apache Kafka**, dentro de un entorno controlado y portable. Este entorno servirá como base para el procesamiento de grandes volúmenes de datos sobre la plataforma Hadoop.

5.2. Componentes del Entorno

El ecosistema de servicios a desplegar mediante Docker estará compuesto por los siguientes contenedores:

- **Zookeeper:** servicio de coordinación y gestión de clústeres necesario para el correcto funcionamiento de Kafka.
- **Kafka Broker:** sistema de mensajería distribuido encargado de recibir, almacenar y distribuir los mensajes provenientes del *Producer*.
- **Producer:** aplicación desarrollada en Python que leerá los datos del dataset energético y los enviará a Kafka con una frecuencia de 5 segundos por mensaje.

Cada contenedor se comunicará dentro de una misma red interna de Docker, permitiendo que el flujo de datos se mantenga encapsulado y fácilmente replicable en distintos entornos.

5.3. Arquitectura del Entorno Docker

El despliegue se realizará utilizando un archivo `docker-compose.yml`, que permitirá definir los servicios y sus relaciones. El esquema de comunicación entre los contenedores se muestra en la Figura **??**.

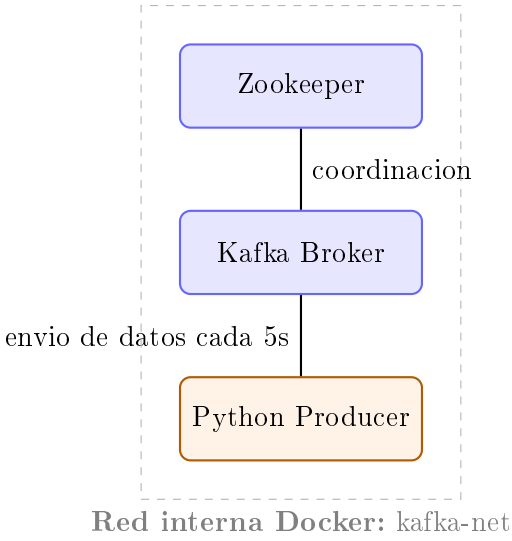

Figura 2: Arquitectura lógica del entorno Docker para la simulación de datos energéticos.

5.4. Configuración de los Servicios

En la práctica, la definición de los servicios en el archivo `docker-compose.yml` permitirá establecer:

- Los nombres y roles de cada contenedor (por ejemplo, `zookeeper`, `kafka`, `producer`).
- Las redes de comunicación internas (`kafka-net`).
- Los puertos expuestos (por ejemplo, `9092` para Kafka y `2181` para Zookeeper).
- Las variables de entorno requeridas por Kafka y Zookeeper para inicializar correctamente el clúster.

5.5. Resultados Esperados

Al finalizar esta fase, se espera haber logrado:

- Un entorno completamente funcional con Kafka y Zookeeper ejecutándose en contenedores Docker.
- Un contenedor adicional preparado para ejecutar el *Producer* que enviará los datos simulados.
- Conectividad validada entre el *Producer* y el Kafka Broker mediante la red interna del entorno Docker.

Este entorno constituye la infraestructura base sobre la cual se implementará el flujo de datos en tiempo real y el almacenamiento distribuido en fases posteriores.