Facultad de Matemática y Computación
Universidad de La Habana

Optimización de inventarios y logística en retail

Desarrolladores

Adrián Hernández Castellanos – C412

Laura Martir Beltrán – C411

19 de octubre de 2025

Índice

Resumen

El presente proyecto implementa un **cluster distribuido (HDFS + YARN + Hive + Spark)** sobre contenedores Docker para simular, procesar y analizar grandes volúmenes de datos de ventas e inventario, y para entregar predicciones de demanda y métricas de logística en tiempo real mediante una interfaz *Streamlit*.

El sistema permite experimentar con ingestión en **batch**, limpiar y transformar los datos con Spark, persistir resultados en *Postgres* y mostrar dashboards interactivos con estadísticas y predicciones. El objetivo final es reducir roturas de stock y optimizar costes logísticos mediante predicción de demanda y políticas de reposición automáticas o asistidas.

1. Dataset seleccionado

El dataset seleccionado ha sido extraído del sitio *Kaggle* y muestra gran similitud con el dataset original propuesto en la orientación de este proyecto.

- **Nombre:** Retail Store Inventory Forecasting Dataset.
- **Fuente:** KAGGLE
- **Formato:** CSV original + ficheros generados y almacenados en HDFS (recomendado transformar a Parquet para procesamiento analítico).

2. Volumen

El CSV base contiene **73 100 entradas**, lo cual lo hace adecuado como semilla para generar volúmenes mayores mediante el componente `data-producer`. Este nodo simula nuevas observaciones aplicando pequeñas variaciones estadísticas, permitiendo generar conjuntos de prueba de diferentes tamaños:

- **Pequeño:** 100k–1M registros, útil para pruebas funcionales y desarrollo.
- **Medio:** 1–10M registros para validación de rendimiento y escalado.
- **Grande:** 10–100M+ registros para simular cargas Big Data y probar comportamiento de YARN/Spark/HDFS bajo estrés.

3. Muestra del esquema (columnas)

El esquema del dataset base contiene los siguientes campos:

Columna	Descripción
Date	Fecha de la observación (formato ISO: YYYY-MM-DD)
Store ID	Identificador de la tienda
Product ID	Identificador del producto
Category	Categoría del producto
Region	Región o zona geográfica
Inventory Level	Nivel actual de inventario (unidades)
Units Sold	Unidades vendidas en el periodo
Units Ordered	Unidades solicitadas/procesadas
Demand Forecast	Valor histórico de pronóstico (si disponible)
Price	Precio unitario
Discount	Descuento aplicado
Weather Condition	Condiciones meteorológicas (ej.: Sunny, Rainy)
Holiday/Promotion	Indicador de promoción o día festivo
Competitor Pricing	Precio de competidor (si disponible)
Seasonality	Indicador de estación/temporada

4. Características

- **Atributos numéricos:** Inventory Level, Units Sold, Units Ordered, Price, Discount, Competitor Pricing, Demand Forecast.
- **Atributos categóricos:** Store ID, Product ID, Category, Region, Weather Condition, Seasonality, Holiday/Promotion.
- **Temporalidad:** Series diarias (con posibilidad de aumentar resolución a horaria si se requiere).
- **Etiquetas / target:** La variable objetivo para forecasting es `Units Sold` en distintos horizontes (T+1, T+7, etc.).
- **Ruido:** Variaciones causadas por clima, promociones, cambios en precios de competencia; el `data-producer` puede introducir ruido controlado.
- **Distribuciones:** Se generan perturbaciones normales centradas en valores base, ajustables por categoría/estación.

5. Pertinencia con el objetivo

El dataset reúne las variables necesarias para construir predicciones de demanda y calcular métricas de inventario. Con los campos de precio, descuento, promoción y condiciones meteorológicas, se pueden generar predicciones y cálculos estadísticos. Además, la estructura temporal y la granularidad por tienda/producto permiten diseñar políticas de reposición, evaluar impactos regionales y calcular indicadores logísticos como días de inventario, tasa de rotación o alertas de stock.

6. Arquitectura propuesta: Batch

A continuación se argumenta la elección de una arquitectura **batch** para este proyecto, con sus ventajas y su integración con tecnologías de Big Data.

6.1. Descripción general

El pipeline propuesto se compone de:

1. Ingesta de datos desde CSV hacia HDFS.
2. Limpieza y transformación con Spark.
3. Registro y consulta mediante Hive.
4. Almacenamiento de resultados en Postgres.
5. Visualización mediante Streamlit.

6.2. Ventajas del enfoque batch

- **Eficiencia computacional:** los jobs pueden agrupar datos y procesarlos masivamente aprovechando los recursos de YARN.
- **Reproducibilidad:** las ejecuciones son determinísticas y trazables, ideales para validaciones y auditorías.
- **Simplicidad de mantenimiento:** los flujos ETL complejos se implementan de forma más limpia que en un sistema streaming.
- **Integración nativa:** HDFS, Hive y Spark están optimizados para este paradigma.

7. Diagrama del sistema y descripción de componentes

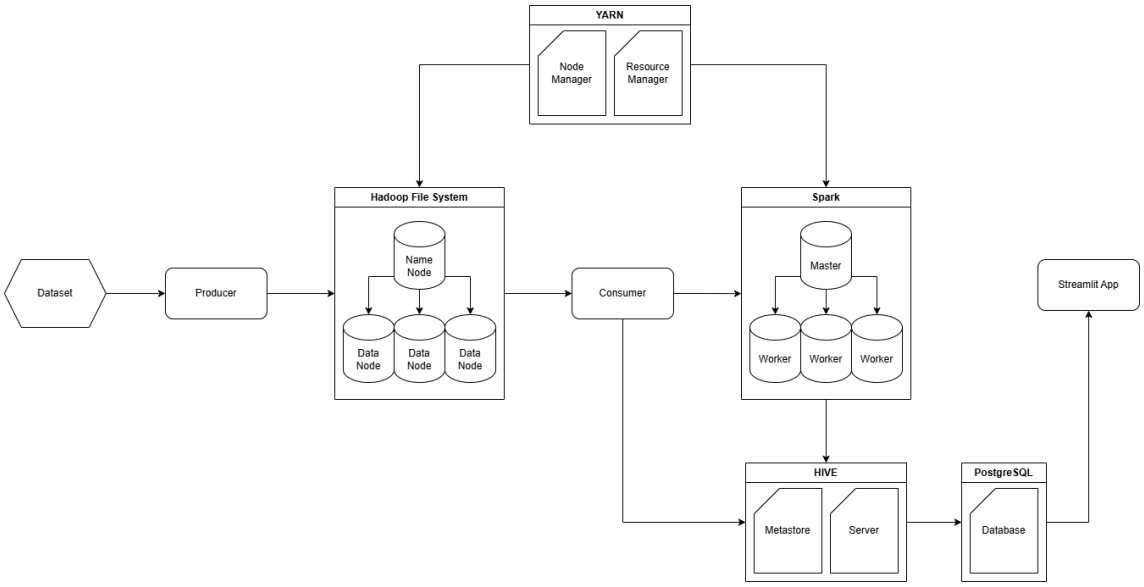

Figura 1: Arquitectura general del sistema distribuido para optimización de inventarios y logística.

Descripción de componentes

- **Data Producer:** Componente encargado de replicar y variar los datos originales del dataset base, aplicando pequeñas perturbaciones controladas (p. ej., fluctuaciones de ventas o clima). Escribe los resultados generados en el sistema distribuido **HDFS**, actuando como fuente continua de datos para las fases posteriores.
- **Data Consumer:** Nodo que lee los datos desde HDFS, los interpreta y limpia, eliminando inconsistencias o valores atípicos. Realiza operaciones de agregación y normalización, guardando los resultados en la base de datos de PostgreSQL a través de HIVE. Su procesamiento está basado en **Spark**.
- **Streamlit:** Interfaz visual que se conecta a la base de datos para mostrar resultados, métricas y predicciones. Ofrece paneles de control con gráficos dinámicos, indicadores de stock y alertas de reposición, permitiendo al usuario interactuar en tiempo real con los datos.
- **Hadoop File System:** Conjunto de nodos de almacenamiento y un nodo maestro, que representan la infraestructura del sistema de archivos de Hadoop.

- **YARN:** Conjunto de contenedores con software encargado de distribuir los recursos del sistema para realizar procesos.
- **Spark:** Componente que realiza los trabajos distribuidos. El consumer se ejecuta sobre este componente.
- **HIVE:** Componente intermediaria entre el output del consumer y la base de datos. Realiza consultas y almacena datos sobre la base de datos de PostgreSQL
- **PostgreSQL:** Base de datos del sistema. Streamlit carga los datos a través de ella.