COMILLAS
UNIVERSIDAD PONTIFICIA
ICAI ICADE CIHS

Sprint 3. Historia de Usuario 4

Elena Conderana Medem y Sergio Cuenca Núñez

Tecnologías de Datos Masivos
Big Data Technology

Índice

Introducción

1. Contexto

El objetivo final del proyecto busca diseñar e implementar una arquitectura Big Data completa, que permita procesar y analizar datos procedentes de `TradingView` para poder asesorar a clientes en el trading de criptomonedas. Este sprint consiste en la implementación del almacenamiento de los datos históricos basado en tres capas mediante Apache Spark, que permitan realizar un posterior análisis técnico eficiente de las criptomonedas.

2. Descripción del problema

El análisis técnico de criptomonedas consiste en el estudio de métricas como la media móvil o el RSI de estas con el propósito de anticipar con mayor probabilidad cambios y tendencias. Para poder implementar este análisis es necesario calcular métricas. Al poseer una gran magnitud de datos es recomendable trabajar en formatos binarios como Parquet, que agilizan el procesamiento de los datos. Por ello los archivos CSV, que se encuentran en la capa Bronce, se transforman y almacenan en formato Parquet en la capa Plata. Sobre estos archivos binarios se realiza el cómputo de distintas métricas que se almacenan en una tabla en la capa Oro.

3. Objetivos

Los objetivos principales de esta práctica comprenden:

- Implementación de almacenamiento en capas Bronce, Plata y Oro.
- Almacenamiento en formato Parquet de archivos CSV en capa Plata.
- Cálculo de 4 indicadores para cada criptomoneda: Moving Average Simple (SMA), Moving Average Exponential (EMA), Relative Strength Index (RSI) y Moving Average Convergence Divergence (MACD).
- Almacenamiento de indicadores en una tabla en la capa Oro.

4. Justificación

El análisis técnico requiere el cálculo de diversos indicadores para poder tomar decisiones informadas. El almacenamiento en el formato binario Parquet permite agilizar el cómputo de los KPIs dada la amplia cantidad de datos. Además, el hecho de almacenar los datos en distintas capas proporciona acceso tanto al formato más bruto de los mismos en la capa Bronce, como a la versión procesada en la capa Oro.

Metodología

En la siguiente sección se procede a describir en detalle cómo se han almacenado los archivos en formato Parquet en la capa Plata y cómo se han calculado los 4 indicadores para cada criptomoneda y se han almacenado en la capa Oro. Se abordarán cuestiones técnicas sobre el proceso de desarrollo y sobre las tecnologías utilizadas.

5. Descripción del entorno de desarrollo

El almacenamiento de los archivos en las distintas capas se realiza en el Clúster ICAI. Para todo el proceso se emplea la interfaz para Apache Sparke en Python `pyspark`. Este marco de computación distribuida permite procesar datos a gran escala de manera ágil con comandos tipo Python y SQL.

6. Diseño de la solución

Para almacenar todos los datos en formato binario se ha implementado un bucle for para leer los archivos CSV de cada criptomoneda y almacenarlos particionados por año nuevamente en Parquet en la capa Plata. Este proceso se muestra en el Código 1.

El cálculo de métricas se ha realizado mediante el uso de dos bucles for. El primer y principal bucle accede a cada criptomoneda de manera secuencial y aplica los cálculos de las métricas. No obstante, antes de realizar dichos cálculos es necesario implementar un segundo bucle interno que recorra los años por si las métricas requieren valores de años previos para realizar el cómputo. No se va a explicar el proceso de cómputo de cada métrica, pues se considera superfluo para el propósito de este informe. Lo único que cabe recalcar es que cada una de estas métricas se ha calculado con frecuencia diaria y se ha almacenado como columna independiente para cada una de las fechas contenidas. Finalmente, las tablas se han particionado por años y almacenado en formato Parquet en la capa Oro. El Código 2 muestra el proceso explicado.

```
cryptos = [
    "BTCUSDT", "ETHUSDT", "XRPUSDT", "SOLUSDT",
    "DOGEUSDT", "ADAUSDT", "SHIBUSDT", "DOTUSDT",
    "AAVEUSDT", "XLMUSDT"
]

for symbol in cryptos:
    # Load data from HDFS
    df = spark.read \
              .option("header", "true") \
              .option("inferSchema", "true") \
              .csv(f"/datos/gittba/gittba04/{symbol}")

    # Transform to Parquet and Load to HDFS again

df.write.mode("overwrite").partitionBy("year").parquet(f"/datos/gittba/gittba04/{symbol}_Silver")

print("Silver Layer Completed")
```

Código 1. Almacenamiento de datos en Parquet en capa Plata

```
from pyspark.sql.functions import col, avg, lit, lag, when, expr, year
from pyspark.sql.window import Window

years = [2021, 2022, 2023, 2024]

for symbol in cryptos:
    for year_value in years:

        df_current =
spark.read.parquet(f"/datos/gittba/gittba04/{symbol}_Silver/year={year_value}")
        if year_value > 2021:
            try:
                df_previous =
spark.read.parquet(f"/datos/gittba/gittba04/{symbol}_Silver/year={year_value-1}")
                df_previous = df_previous.orderBy(col("date").desc()).limit(200)  # Keep only the
last 200 days
                df = df_previous.union(df_current)  # Merge previous & current year data
            except Exception as e:
                print(f"Warning: Could not load previous year's data for {symbol} {year_value}:
{e}")
                df = df_current
        else:
            df = df_current

        df = df.orderBy("date")

        # SMA200 Calculation (Simple Moving Average)
        window_spec_200 = Window.orderBy("date").rowsBetween(-199, 0)  # 200-day rolling window
        df = df.withColumn("SMA200", avg(col("close")).over(window_spec_200))

        # EMA50 Calculation (Exponential Moving Average 50)
        window_spec_50 = Window.orderBy("date").rowsBetween(-49, 0)
        df = df.withColumn("EMA50", avg(col("close")).over(window_spec_50))

        # MACD Calculation (12-day EMA, 26-day EMA, Signal Line)
        df = df.withColumn("EMA12", avg(col("close")).over(Window.orderBy("date").rowsBetween(-
11, 0)))
        df = df.withColumn("EMA26", avg(col("close")).over(Window.orderBy("date").rowsBetween(-
25, 0)))
        df = df.withColumn("MACD", col("EMA12") - col("EMA26"))
        df = df.withColumn("MACD_Signal",
avg(col("MACD")).over(Window.orderBy("date").rowsBetween(-8, 0)))

        # RSI Calculation (Relative Strength Index - 14)
        window_rsi = Window.orderBy("date").rowsBetween(-14, -1)
        df = df.withColumn("delta", col("close") -
lag(col("close")).over(Window.orderBy("date")))
        df = df.withColumn("gain", when(col("delta") > 0, col("delta")).otherwise(lit(0)))
        df = df.withColumn("loss", when(col("delta") < 0, -col("delta")).otherwise(lit(0)))
        df = df.withColumn("avg_gain", avg(col("gain")).over(window_rsi))
        df = df.withColumn("avg_loss", avg(col("loss")).over(window_rsi))
        df = df.withColumn("RS", col("avg_gain") / col("avg_loss"))
        df = df.withColumn("RSI", 100 - (100 / (1 + col("RS"))))

        # Select only relevant columns
        df = df.select("date", "SMA200", "EMA50", "RSI", "MACD")
```

```
        # Add 'year' column before filtering**
        df = df.withColumn("year", year(col("date")))

        # Filter only the correct year's data
        df = df.filter(col("year") == year_value)

        # Store results in Gold Layer
        output_path = f"/datos/gittba/gittba04/{symbol}_Gold"
        df.write.mode("append").partitionBy("year").parquet(output_path)

print("Gold Layer Completed")
```

Código 2. Cálculo de métricas y almacenamiento en tabla en capa Oro.

7. Pruebas realizadas

La correcta implementación de las capas se ha realizado en dos pasos. El primero corroborando que los cómputos para la capa Oro acceden correctamente a los datos almacenados en Parquet en la capa plata. El segundo paso ha consistido en comparar pequeñas muestras de valores de las 4 métricas obtenidas con los valores que figuran en la plataforma `TradingView`.

Resultados

A continuación, se procede a estudiar los resultados obtenidos durante la ejecución de la práctica.

8. Descripción de los resultados

Con la metodología anterior se han transformado los archivos CSV de cada criptomoneda al formato Parquet y se han cargado nuevamente a HDFS manteniendo la partición por años implementada en el sprint anterior. Esta transformación permite acelerar el cómputo de las métricas que derivan en una tabla final para cada criptomoneda con los valores de SMA200, EMA50, RSI y MACD para cada fecha. Dicha tabla también se almacena en formato Parquet.

9. Pantallazos de la Ejecución

En este apartado se muestran dos consultas realizadas con `pyspark` que ejemplifican el funcionamiento del sistema para los valores de la tabla `BTCUSDT_Gold`. En la Figura 1 se muestran las primeras 20 filas de esta criptomoneda con cada uno de sus métricas computadas. La Figura 2 muestra la estructura de la tabla con los distintos campos que la conforman y el tipo de dato que admite cada columna.

date	SMA200	EMA50	RSI	MACD	year
2024-01-01 01:00:00	32026.42679999998	40620.74779999999	55.35557511757836	219.40762820511736	2024
2024-01-02 01:00:00	32119.436349999978	40790.42739999999	58.36250432756507	245.45769230768929	2024
2024-01-03 01:00:00	32201.07754999998	40936.30819999999	64.08089243162314	202.79762820513133	2024
2024-01-04 01:00:00	32290.13319999998	41062.1662	45.95945821323388	223.38237179486168	2024
2024-01-05 01:00:00	32376.636999999984	41221.7982	51.27941595302999	305.839615384597	2024
2024-01-06 01:00:00	32454.938649999982	41368.88620000001	50.7856977747711	234.10211538460862	2024
2024-01-07 01:00:00	32524.614299999987	41516.104600000006	51.19734403397331	258.7264102564077	2024
2024-01-08 01:00:00	32609.944899999988	41707.9282	54.48864528632875	395.2418589743611	2024
2024-01-09 01:00:00	32687.05239999999	41881.15260000001	63.100717899667956	572.0026282051258	2024
2024-01-10 01:00:00	32767.685199999985	42099.3994	64.22845050009995	772.9601923076916	2024
2024-01-11 01:00:00	32847.067699999985	42278.01580000001	63.13323896815093	966.6695512820515	2024
2024-01-12 01:00:00	32909.64139999999	42387.7848	66.09528421899887	954.108461538468	2024
2024-01-13 01:00:00	32970.419149999994	42490.473200000015	52.42015431535244	835.8301282051325	2024
2024-01-14 01:00:00	33028.693849999996	42569.5068	52.39417503448958	588.8593589743468	2024
2024-01-15 01:00:00	33089.01279999999	42670.78020000001	48.25030456121283	605.5471153845938	2024
2024-01-16 01:00:00	33152.34254999999	42788.685200000014	44.299775537326724	548.9583333333285	2024
2024-01-17 01:00:00	33213.29354999999	42887.829800000014	43.75981410803036	480.75647435896826	2024
2024-01-18 01:00:00	33266.845899999986	42957.28700000001	49.72899994590189	352.0212179487062	2024
2024-01-19 01:00:00	33319.36004999999	43035.98840000001	39.05355677992516	214.10423076922598	2024
2024-01-20 01:00:00	33374.00769999999	43096.25880000002	40.59932480669868	-151.50128205128567	2024

only showing top 20 rows

Figura 1. Resultado de la consulta `df_btcusdt_gold.show()`

```
root
 |-- date: timestamp (nullable = true)
 |-- SMA200: double (nullable = true)
 |-- EMA50: double (nullable = true)
 |-- RSI: double (nullable = true)
 |-- MACD: double (nullable = true)
 |-- year: integer (nullable = true)
```

Figura 2. Resultado de la consulta ` df_btcusdt_gold.printSchema()`

10. Discusión de los resultados

La correcta implementación de la capa Plata se corrobora mediante la ejecución exitosa mostrada en las Figuras 1 y 2, que requieren acceso a los datos en esta capa. Además, para verificar que el cómputo de las métricas es correcto se ha comprobado el valor en `TradingView` de algunas fechas mostradas en la Figura 1 para cada métrica. Una muestra de este proceso se aprecia en la Tabla 1. La similitud de los valores y la rapidez en los cálculos evidencian el incremento en eficiencia de usar datos almacenados en binario y la correcta implementación de los cálculos.

		SMA200	EMA50
2024-01-01	Solución Propia	32026,43	40620,75
	TradingView	32026,43	40499,49
2024-01-02	Solución Propia	32119,44	40790,43
	TradingView	32201,08	40759,05

Tabla 1. Comparación de métricas de `TradingView` y la solución desarrollada

Conclusión

Por último, se van a resumir los principales hallazgos y aprendizajes obtenidos durante la práctica, y la relevancia de la solución implementada.

11. Resumen del Proceso

Para el almacenamiento de los archivos en HDFS en formato Parquet, y el cómputo y almacenamiento de distintas métricas como paso previo para el análisis técnico de las criptomonedas se han seguido los siguientes pasos.

1. Identificación del problema: Almacenamiento en formato Parquet y cálculo de métricas SMA200, EMA50, RSI y MACD.
2. Diseño de la arquitectura e implementación: Transformación de CSV a Parquet de los datos históricos y almacenamiento en capa Plata. Carga de datos en Parquet y cálculo de métricas SMA200, EMA50, RSI y MACD para cada criptomoneda. Almacenamiento de los valores por fecha en tablas independientes por criptomoneda y particionadas por año en formato Parquet en la capa Oro.
3. Pruebas y análisis: Ejecución de distintas consultas para comprobar el contenido de las tablas en la capa Oro y comparación de los valores con la herramienta `TradingView`.

12. Principales Logros

La solución implementada ha mostrado su eficacia para almacenar y transforma archivos CSV al formato binario Parquet. Además, se han aplicado mecánicas de procesado de datos en forma del cálculo de métricas, que permitan realizar un posterior análisis técnico independiente para cada una de las criptomonedas. Para concluir se van a recorrer los logros más importantes alcanzados en base a los resultados que se han analizado en la sección anterior.

- Familiarización con capas Bronce, Plata y Oro.
- Familiarización con `pyspark`.
- Transformación y almacenamiento de datos en Parquet.
- Familiarización con métricas para medir rendimiento de criptomonedas.
- Computación de métricas SMA200, EMA50, RSI y MACD.
- Almacenamiento en tablas de métricas para cada fecha de cada criptomoneda.
- Acceso mediante `pyspark` a datos procesados.