Case Study: Optimizing Data Processing Pipeline

Background:

You are working for a company that deals with large amounts of data from various sources. The current data processing pipeline is becoming inefficient and slow, leading to delays in processing critical data. Your task is to optimize the data processing pipeline using Python and related technologies.

Requirements:

1. Problem Statement: The current data processing pipeline reads data from multiple sources, processes it, and stores it in a database. However, the process is slow and needs optimization.
2. Data Processing Details:
   * Data Transformation:
     1. Convert the timestamp column to a datetime object for easier manipulation.
     2. Convert sensor readings to a standardized format if needed.
   * Data Cleanup:
     1. Remove any duplicate or invalid data entries.
     2. Handle missing or null values appropriately (e.g., by filling with a default value or interpolating).
   * Data Aggregation:
     1. Aggregate sensor readings over a specific time window (e.g., hourly or daily averages).
     2. Calculate additional metrics such as maximum, minimum, and standard deviation of sensor readings.
   * Input Data Structure:
     1. The input data consists of CSV files with the following columns:
        1. timestamp (string): Timestamp in ISO format (e.g., "2024-03-25T12:00:00").
        2. sensor\_id (int): Unique identifier for the sensor.
        3. value (float): Sensor reading value.
   * Example Input Data:  
     The input data consists of multiple CSV files, each containing a day's worth of sensor data. Each CSV file is approximately 100 MB in size, containing around 1 million rows of data.

|  |  |  |
| --- | --- | --- |
| timestamp | sensor\_id | value |
| 2024-03-25T12:00:00 | 1 | 23.5 |
| 2024-03-25T12:01:00 | 2 | 18.9 |
| 2024-03-25T12:02:00 | 1 | 24.0 |

* + Example Output Data:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| timestamp | average\_value | max\_value | min\_value | std\_dev |
| 2024-03-25T12:00:00 | 23.75 | 24.0 | 23.5 | 0.25 |
| 2024-03-25T13:00:00 | 22.1 | 23.8 | 20.5 | 1.2 |

1. Debugging Tools/Methods:
   * Use logging to track the flow of data through the pipeline and identify bottlenecks.
   * Present usage of appropriate logging tool/framework according to industry standard.
2. Benchmarks:
   * Measure the time taken for each step of the pipeline before and after optimization.
   * Compare the memory usage of the optimized pipeline with the original implementation.
3. Methods and Tools for Benchmarking:
   * Python’s inbuilt module can be used to measure the execution time of specific code segments.
   * Use memory profiling tools to analyze memory usage during runtime (for eg memory\_profiler).
4. Optimization Techniques:
   * Implement parallel processing using multiprocessing or multithreading to speed up data processing.
   * Use async programming with asyncio to handle I/O-bound operations efficiently.
   * Apply design patterns like singleton pattern, abstract factory pattern, and decorator pattern to improve code structure and maintainability.
   * Utilize asynchronous database management like asyncpg to improve database operations.
5. Security Practices:
   * Ensure that the optimized pipeline handles sensitive data securely.
   * Implement basic security practices such as data encryption and access control.
6. Collaboration and Communication:
   * Work with the team to identify key areas of improvement in the current pipeline.
   * Communicate the progress and challenges faced during the optimization process.
7. Cloud Services Integration (Optional):
   * If applicable, explore integrating the optimized pipeline with cloud services such as AWS, Azure, or GCP to improve scalability and reliability.

Deliverables:

Optimized data processing pipeline code in Python.

Documentation explaining the optimizations made, including code snippets and benchmark results.

Presentation highlighting the key improvements and challenges faced during the optimization process.

Based on the input data size mentioned above, please provide the system requirements (hardware and software) you would recommend for processing this data efficiently.

Timeframe:

You have one week to complete the assignment. Please provide regular updates on your progress and seek assistance if needed.

Evaluation Criteria:

Efficiency of the optimized pipeline.

Correct implementation of parallel processing, async programming, and design patterns.

Adherence to security practices and data handling.

Quality of documentation and presentation.

Communication and collaboration skills demonstrated during the assignment.