Technical Report: Efficient Reading of Large CSV Files in Python

1. Overview

This report describes three efficient techniques for reading very large CSV files (e.g., 5GB or larger) in Python. Loading large files entirely into memory can cause performance degradation or crashes. The following approaches improve memory efficiency and performance: Pandas with chunksize, Dask DataFrame, and Compression.

2. Pandas with Chunksize

Using the chunksize parameter in pandas.read_csv() allows the file to be read incrementally in smaller portions instead of loading the entire dataset at once. Each chunk is processed sequentially and discarded after use, keeping memory usage controlled.

Characteristics: Low memory usage, single-core execution, suitable for sequential processing and systems with limited RAM.

3. Dask DataFrame

Dask extends the Pandas API to support parallel and out-of-core computation. It splits large files into partitions and builds a lazy execution task graph. Computation is triggered only when compute() is called.

Characteristics: Multi-core parallel processing, memory-efficient, suitable for very large datasets and high-performance systems.

4. Pandas with Compression

Compressed CSV files (e.g., gzip format) reduce disk usage and I/O overhead. Pandas can decompress data on-the-fly while reading. However, unless used with chunksize, the entire dataset is still loaded into memory after decompression.

Characteristics: Reduced disk size, higher CPU usage due to decompression, useful for storage optimization and network transfer.