# Scalability and Optimization:

* **Reading the files (location):**

Currently, we are reading files from local which doesn't have performance issue. If the files are available on some other servers like s3. We need to use APIs provided by them to load the file which is not currently supported in my code. And the time to read large files will also be increased which affects the performance of the API. then we should consider moving it to a different architecture where API just acts as a trigger for the job that performs ETL and other methods like mounting the servers where the files are present to the compute where this Flask server is running.

* **Reading large files:**

Reading files with large data may not be supported by the modules I use for example reading the CSV using pandas has a huge impact on the performance. alternative methods like reading the files into chunks or other module which doesn't have impact on the performance can be used.

* **Module Support:**

Currently, I am using modules for parsing dates on the fields. those modules might be successful with our current samples files but it doesn't guarantee successful outcomes in all the cases

* **Transformation:**

Current transformations on the dataframes are simple as the data become more complex and large we need to use multiprocessing to speed up the process or modules like Pyspark, Dask, and swifter.

* **Monitoring:**

Currently, I added monitoring only for the time taken to process each file. once I know more about the data we can set monitoring customer wise or item wise, depending on our need. for example which item has been purchased more times. Currently, I added time taken in seconds but we can move to nanoseconds if need more precise as seconds can be zero for smaller datasets

* **Multiple requests:**

if you send multiple requests at the same time to the same server It causes file errors as we are working on the same file. If each requests have different files and files are loaded on to the server. It should not cause any error.

# Future Improvements:

A diagram of a machine

Description automatically generated

I like to propose to use Multiple engines for transforming the data. The controller will decide which engine to use based on the value in the config, the size of the data, or type of the data. It will inject the engine into the FileProcessing Service. And FileProcessing service will use the Transformation engine injected by the Controller to transform the data. Some of the Engine that were inherited from the Transformation engine are the Multiprocessing engine, Dask Engine, Pyspak engine, pandarallel engine. Based on the different comparisons we can go with one engine. We can introduce new engines which will child classes of Transformation engine