# **Correlation**

(Python/C++)

Suppose we have two programs:

Program1: load matrix data (10000x1000) from file on local disk. (if the file does not exist generate the matrix on the fly.)

Every random 1~5 seconds it updates random half of the cells in the matrix using norm distributed random values and updates the same file on disk.

Program2: every time program1 finishes updating, calculate column-wise Pearson correlation of the latest matrix as fast as possible and dump to local disk.

Implement the two programs (Python or C++):

Use as little I/O as possible, max number of processors or threads is equal to half of the total number of logical processors.

Consider cases below and implement or answer:

1. Think about ways to minimize the risk of the file being deleted accidentally by another process

2. How to roll back to the last update if the server is powered off accidentally.

3. Record timestamps of each update and calculation.

4. Only one instance of each program should be running at the same time.

5. How to make the two programs running on different machines.

6. If the correlation is too slow and doesn't finish before the next update

7. If the matrix dimension can be very large

Please submit all necessary code including makefiles if any.

# **Running Regression**

Implement the following functions in **at least two ways**, with properly designed Python APIs. Assuming there is a Pandas data frame which contains columns of date , ticker, return , and weight , where date column contains integers in format of YYYYMMDD , ticker column contains strings such as AAPL , IBM , etc. and all other columns contain floating-point numbers.

* Market return(date): Calculate the weighted average with weight and return of all rows of the same date .
* Beta(date, ticker): The slope of the running least-square regression (of 128-day window excluding current date, aka., from 128 dates before current date until the date before current date) of individual stock returns (return of the given ticker ) to the above market returns. Start from the earliest date in the Pandas data frame if there are not 128 dates before current date. Notice that date in the data frame may not be consecutive. For example, weekends should be excluded. Given a time series of returns of ticker *T* (say *R*) and the calculated market returns (say *M*) (both having the same length of *L*, where *L* <= 128), for instance, you estimate the return for *T* as *R’ = b \* M + c*, with appropriate *b* and *c*. so that sum of square errors of observed value *R* and estimated value *R’* is minimized. Details of least square regression can be found [here](https://statisticsbyjim.com/regression/least-squares-regression-line/).

The baseline implementation is expected to be based on Pandas built-in functions, and you are expected to further enhance implementation so as to beat the baseline implementation in speed or memory consumption or both. For example, you may choose numpy, JIT compilation, cython, or Python wrapper (CPython is required) on your function in another high-performing language.   
  
You need to create such Pandas data frame for inputs. Please submit both source code and data inputs. If your code needs to be compiled, please include the compilation instructions too.