Taq Data Documentation

2025-08-01

Overview

This document outlines the procedure for extracting, processing, and storing Trade and Quote (TAQ) data at a 1-second frequency for S&P 500 securities using the WRDS Cloud environment. The steps include file transfer, job submission using the WRDS batch system, output directory structure, and compression of results for archival or download.

1. File Transfer to WRDS Cloud

Required scripts and input files must first be transferred from the local environment to the WRDS Cloud.

```
scp [local_path]/taq_chunks.py [username]@wrds-cloud.wharton.upenn.edu:/home/[school_name]/[username]
scp [local_path]/sp500ccm_filtered.csv.gz [username]@wrds-cloud.wharton.upenn.edu:/home/[school_name]/
```

Upon execution, the system will prompt for WRDS login credentials.

> The file `sp500ccm_filtered.csv.gz` is generated using a separate script (`sp500ccm_generator.py`) and contains a list of valid PERMNOs representing current S&P 500 constituents.

2. Connecting to WRDS via SSH

Once the required files are uploaded, establish a remote connection to the WRDS Cloud:

```
ssh [username]@wrds-cloud.wharton.upenn.edu
```

Login credentials will be requested upon connection.

3. Preparing and Submitting Batch Jobs

To avoid memory overload during processing, the data pulling process can be divided into quarterly (3-month) interval jobs. A batch job script should be created using a text editor such as `nano`.

```
nano taq_job.sh
```

Insert the following script into `taq_job.sh`:

```
#!/bin/bash
#$ -cwd
#$ -pe onenode 1
#$ -l m_mem_free=26G
#$ -m abe
#$ -M [email_address]

echo "Starting Job at `date`"
python3 taq_chunks.py "$1" "$2"
echo "Ending Job at `date`"
```

This script is configured to:

- Execute in the current working directory
- Allocate one node with 26 GB of memory
- Send email notifications on job abort, begin, and end
- Pass two date arguments to the Python script

Submit jobs using the following format:

```
qsub taq_job.sh 2020-01-01 2020-03-31
```

Each pair of dates defines the interval of data to be processed in that batch.

4. Job Monitoring

To monitor the status of jobs currently in the queue or executing:

```
qstat
```

To view details about a specific job:

```
qstat -j [job_id]
```

5. Output Directory Structure

The `taq_chunks.py` script automatically creates a structured directory hierarchy to store the output data. The structure is as follows:

```
data/
    2020/
        01/
            01/
            02/
            ...
        02/
        ...
    2021/
    ...
```

Each year (`YYYY`) contains folders for each month (`MM`), which in turn contain folders for each day (`DD`). Inside each day folder are the 1-second resolution CSV files generated for that date.

6. Archiving Processed Data

To facilitate download or archival, processed yearly datasets can be compressed using the `tar` utility.

Example (for the year 2022):

```
tar -czfv compressed/2022.tar.gz data/2022
```

This command creates a GZIP-compressed tar archive of all data from the year 2022 and places it in the `compressed/` directory. Repeat the process for other years as needed:

```
tar -czfv compressed/2021.tar.gz data/2021
tar -czfv compressed/2020.tar.gz data/2020
```

Summary of Workflow

- Uploaded `taq_chunks.py` and `sp500ccm_filtered.csv.gz` to the WRDS Cloud via `scp`
- Connected to the WRDS Cloud environment via `ssh`
- Created and configured `taq_job.sh` to manage batch processing using the Grid Engine
- Submitted jobs using `qsub`, processing data in 3-month chunks to minimize memory issues
- Monitored job status and logs using `qstat` and `qstat -j`
- Output data was automatically organized by year, month, and day
- Final data was compressed using `tar -czfv` for efficient transfer or storage