universität
innsbruck

Project 3 - Stock Predictions

The University of Innsbruck was founded in 1669 and is one of Austria's oldest universities. Today, with over 28.000 students and 5.000 staff, it is western Austria's largest institution of higher education and research. **For further information visit: www.uibk.ac.at.**

Project Report

Team members:

- Heine Maximilian
- Kotrba Stefan
- Larcher Thomas

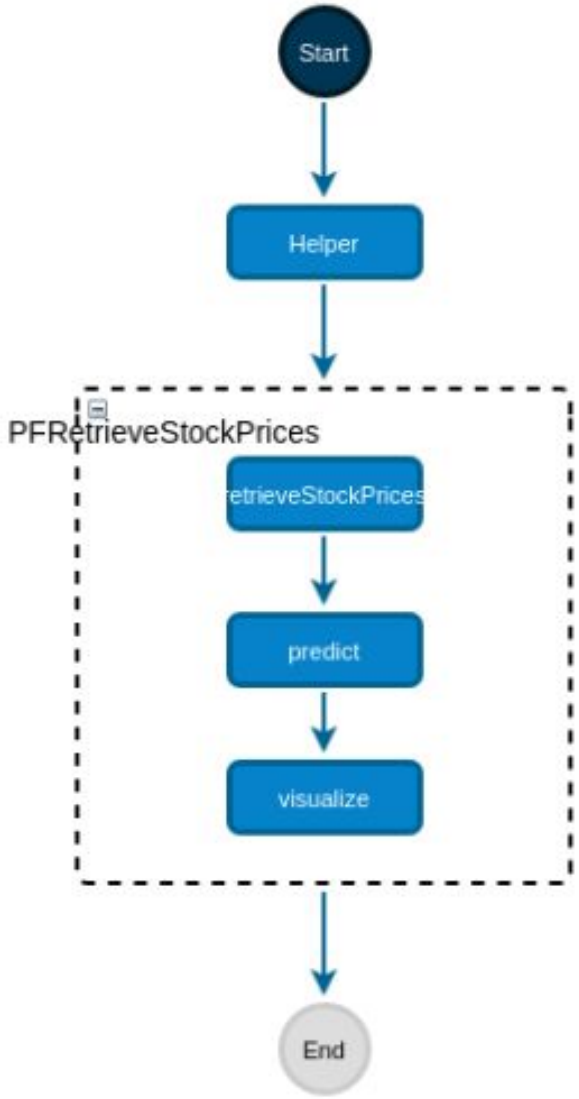
Introduction

- Project Number 3 - calculating Stock predictions and visualizing the results
- written in Python

- Workflow, that is simple to use for an user
- Input: stock symbols and S3 bucketname
- Workflow takes care of data retrieval, training a model, running predictions and visualizing the results
- distributed approach for computation of multiple results in parallel

Workflow

- Helper
- Parallel For
  - Executes for every symbol
- Retrieve Stock Prices
  - Loads Data from with Alpha Vantage
  - Saves in S3
- Predict
  - Executes forecast algorithm
- Visualize
  - Creates Chart with QuickChart



Main challenges

- Forecast:
- AWS Forecast
  - exceeded 15 min execution time
- fbprophet library
  - exceeded 250MB storage
- Solution:
  - AWS Elastic Filesystem
    - No size limit
    - only mount when needed
    - increase in lambda initialization time


KEEP
CALM
IT'S
DEMO
TIME

Performance Measurement 1 (Cold vs Warm start)

Cold Start: Lambda function is not already running (Initialization + Execution)

Warm Start: Lambda function already initialized (Execution)

Less difference between cold and warm start for Helper (no filesystem mounted)

Biggest difference for Predict (uses fbprophet, which is a very big library on the filesystem)

Conclusion: Filesystem increases the initialization overhead of lambda functions

Name	Cold/Warm	Min	Max	Average
Helper	Cold	1989ms	2312ms	2133ms
RetrieveStockPrices	Cold	7016ms	7924ms	7516ms
Predict	Cold	20663ms	23790ms	22558ms
Visualize	Cold	8253ms	9564ms	9008ms
Helper	Warm	1105ms	1623ms	1400ms
RetrieveStockPrices	Warm	545ms	1398ms	1095ms
Predict	Warm	3227ms	3530ms	3400ms
Visualize	Warm	1753ms	2470ms	1953ms

Performance Measurement 2 (Input Sizes)

Input size = number of symbols passed as input to the workflow

Most Functions executed in parallel -> Nearly constant execution time O(1)

Two reasons for Overhead:

(1) Communication Overhead (Download result from Lambda, Upload input to Lambda)
(2) Distribute and collect data at start and end of Parallel For

$$overhead(wf) = exec\_time(wf) - \sum_{n=1}^{n} exec\_time(f_i)$$

Input Size	Average Execution Time	Average Overhead
1	7744ms	418ms
2	8227ms	365ms
3	8151ms	398ms
4	8732ms	426ms
5	8673ms	431ms

www.uibk.ac.at