



Harvard John A. Paulson
School of Engineering
and Applied Sciences

Parallelization of stock prediction

Group 11: Kevin Hare, Sivananda
Rajananda, Junkai Ong

Goals

1. Process scalable quantities of high frequency financial transaction data
2. Utilize multiple nodes and GPU to speed up training of sequential models, which are difficult to parallelize
3. Serve predictions based on trained model in a distributed manner



Existing Work

- LSTMs financial data:
 - Krauss and Fischer (2017)
 - Ghosh et al. (2020) - intraday price movements
 - Lanbouri and Achchab (2020)
 - High resolution prediction, done at 1, 5, and 10 minutes ahead
- Paralellized model training
 - Goyal et al. (2018):
 - Training ImageNet in One Hour - 256 GPUs!
 - Jia et al. (2018):
 - Training ImageNet in **Four Minutes** - Mixed precision, 2048 GPUs!



Phase I: Data Processing

```
for t in tickers:  
    for d in days:  
        for n in sequences_per_day:  
            seq = generate_sequence(t, d, n)
```

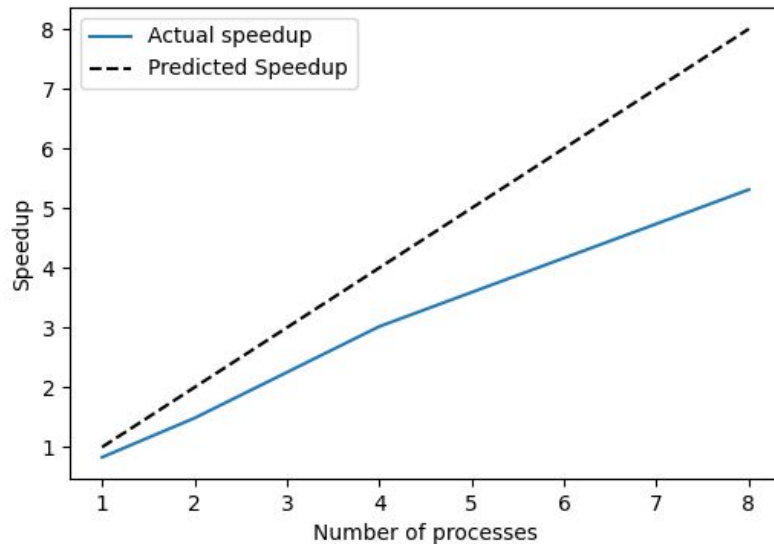


Phase I: A first attempt

Idea:

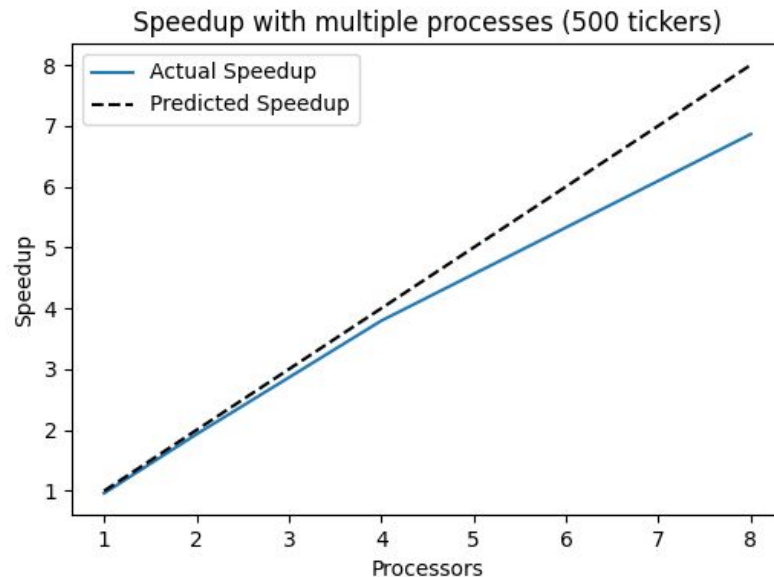
- Break down problem into entirely separate domains for each stock and day
- Python *multiprocessing* module
- Challenge: API ultimately accesses the data separately
 - Large I/O Overhead

Speedup for multiple processes, single thread per process (50 tickers)

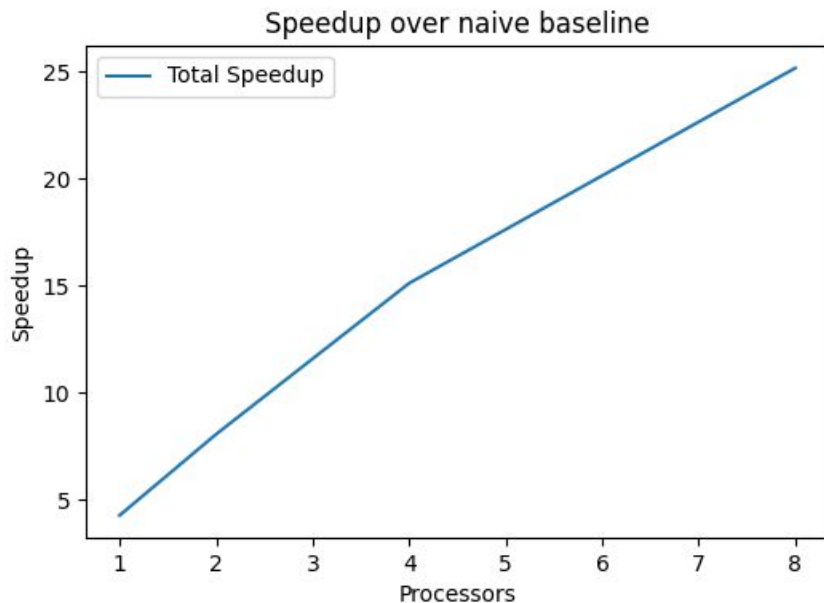


Phase I: Achieving strong scaling

- Solution: *Multithreading*
 - Fine-grained parallelism
 - Optimizes I/O of files
- Pre-implemented in yfinance
- Combine with multiple processes to parallelize both I/O and computation



Phase I: Putting it together



Wall Times:

- Naive Baseline:
 - ~2.5 hours
- Multithreading Baseline
 - 36 minutes
- Fully parallelized:
 - 5 minutes



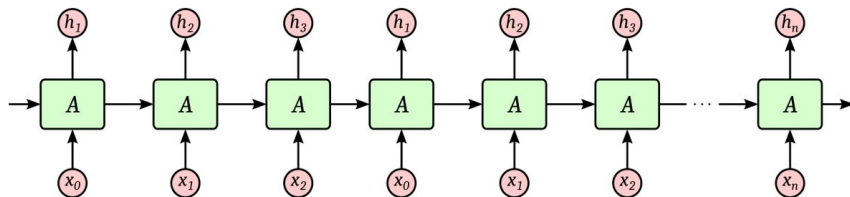
Phase I: Parting thoughts

- Lessons learned:
 - Be wary of I/O, and interacting levels of parallelism
 - Performance gains can enable more experimentation with processing & possibly scientific progress
- Future considerations:
 - *yfinance* implements multithreading on the basis of $2 * \#$ of CPUs available - potentially open to further tuning.
 - Limits to the gains from a practical perspective



Model Training - Overview

- LSTMs are inherently sequential and difficult to parallelize



- Focus on Data Parallelism:
 - Data is distributed across the nodes or GPUs
 - Gradient updates are calculated separately on each node and averaged across all the nodes



Model Training - Programming Model

- Accelerated Computing
 - Parallelization within batches
 - CuDNN, NVIDIA M60 GPU
- Distributed Memory Parallel Processing:
 - Parallelize training between batches
 - 1 - 4 Node
 - 1 - 2 GPUs
 - Horovod
 - Open MPI
 - Ring-AllReduce



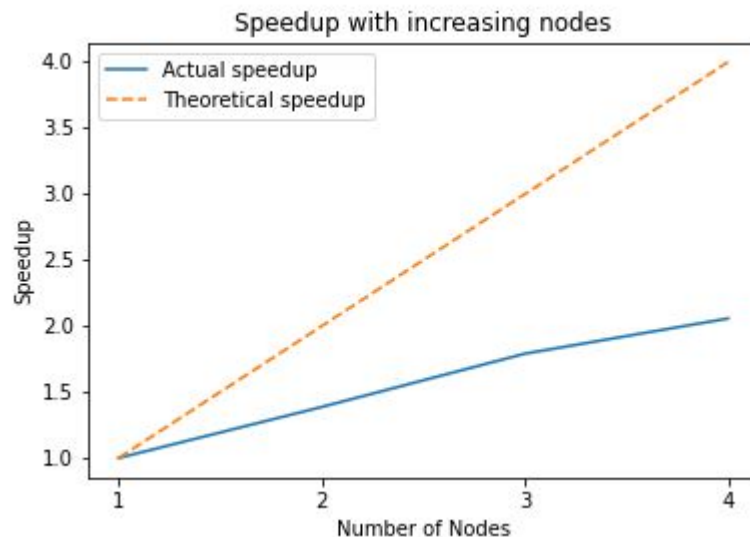
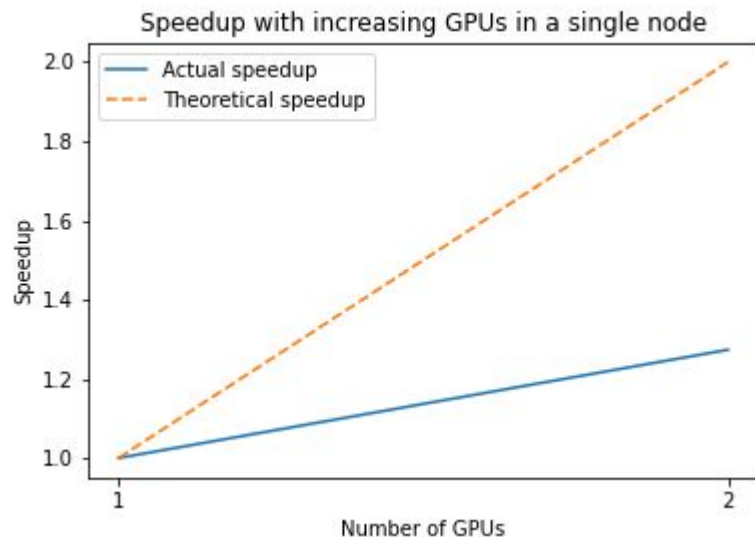
Model Training - Performance

Performance metrics:

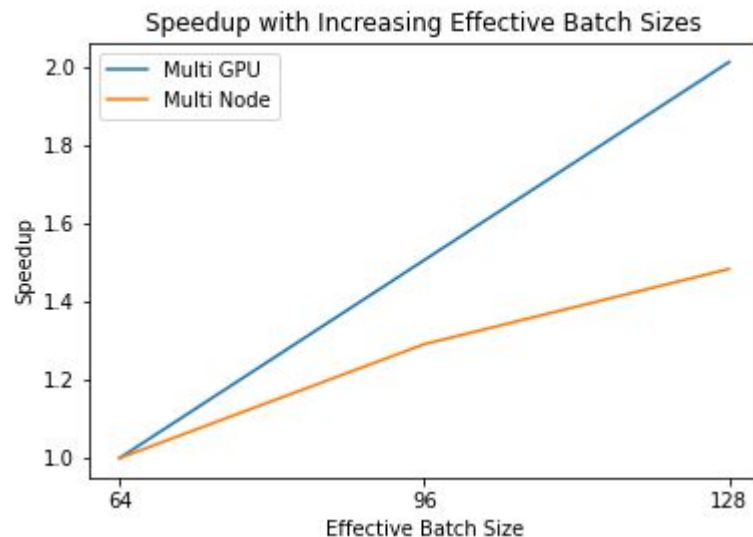
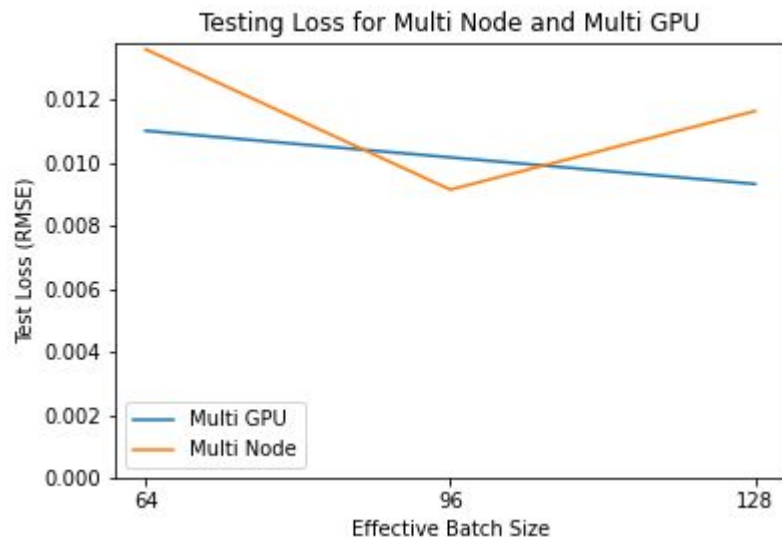
- Speedup
 - Multi-Node vs Multi-GPU
 - Different effective batch sizes
- Time/step & time/epoch
 - Multi-Node vs Multi-GPU
 - Different effective batch sizes



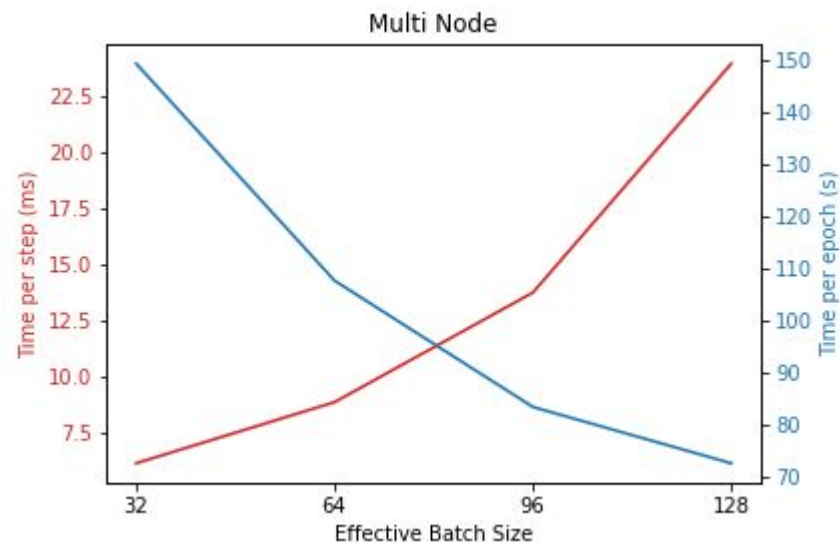
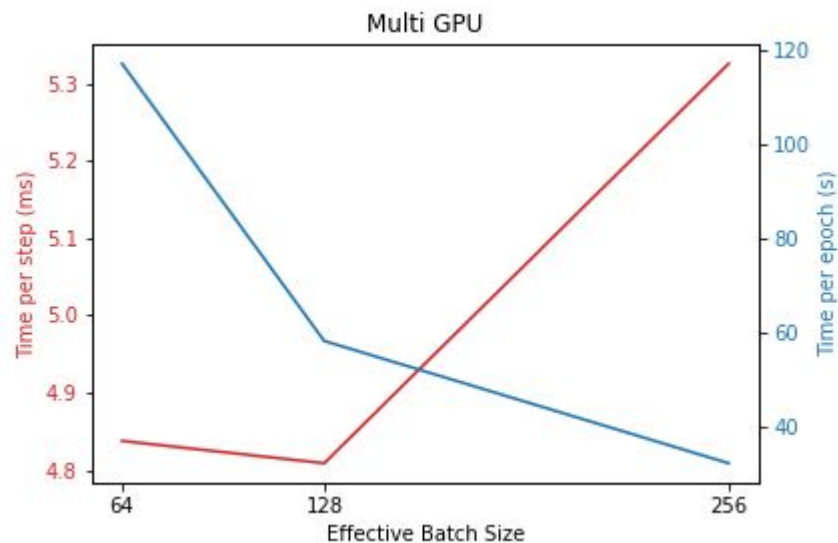
Model Training - Performance



Model Training - Performance



Model Training - Performance



Model Training - Key Findings

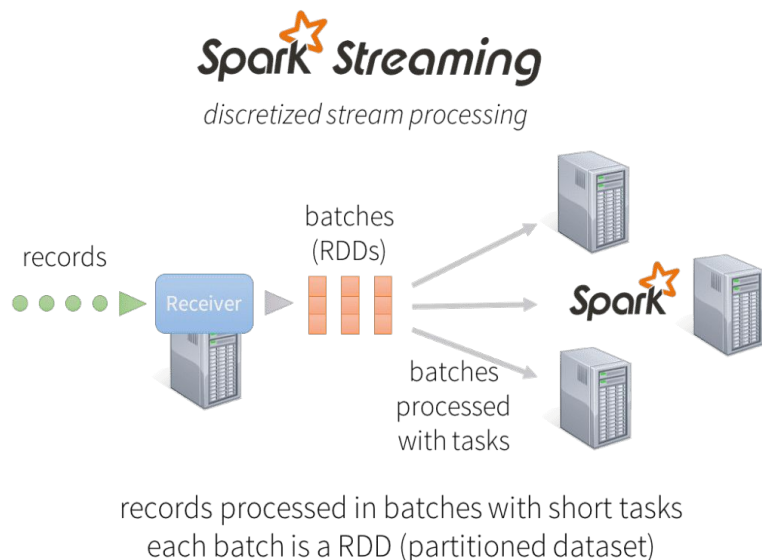
- Scaling efficiency diminishes fairly quickly
- Using larger batch sizes increases speedup (up to a certain point)

Future steps

- Increase accelerated computing:
 - Increasing feature vectors
 - Larger hidden units in the LSTM
- Decrease sequence length



Real-Time Prediction - Spark Programming Model

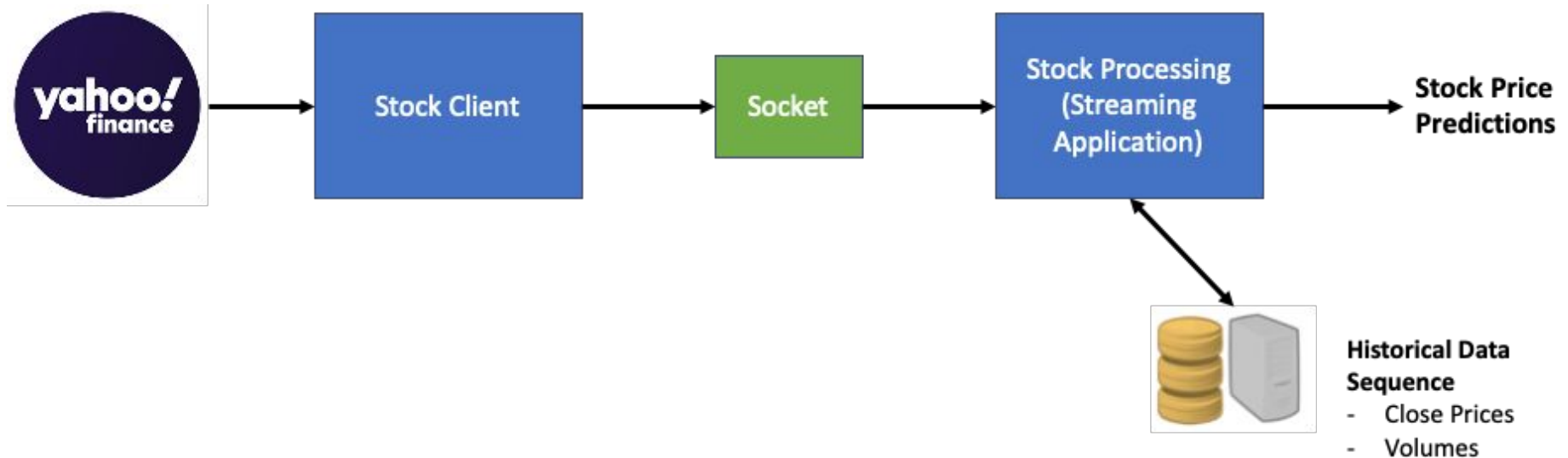


records processed in batches with short tasks
each batch is a RDD (partitioned dataset)

- Streamed Dataset is constructed using data from yfinance API
 - Unbounded dataset
 - Defined as amount of data that has entered system so far
 - Processing is event-based, continues until explicitly stopped
- Stream Processing
 - Predictions are computed based on current stock data
 - Current stock data incorporated into data sequence (60 min)
 - Trained LSTM model predicts stock price using the data sequence



Real-Time Prediction Application Architecture



Real-Time Prediction Demonstration

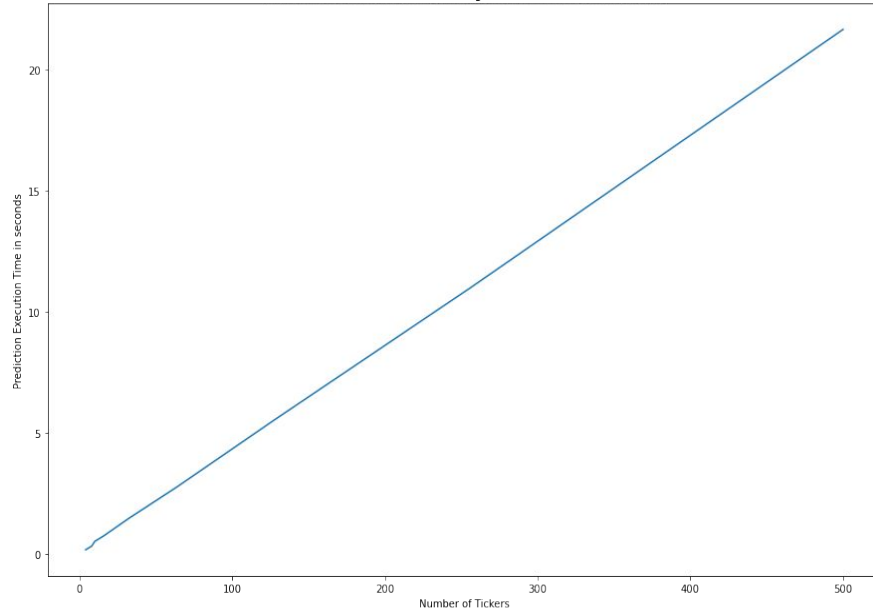
```
junkaliong ~ ubuntu@ip-172-31-51-133: ~/CS205_working -- ssh -i ~/.ssh/CS205-key.pem u...
ABMD>{'Close': 297.95801228783125, 'Volume': 257.0}
ABMD>{'Close': 297.95801228783125, 'Volume': 257.0}\n"
AFL>{'Close': 55.97999954223633, 'Volume': 894.0}
AFL>{'Close': 55.97999954223633, 'Volume': 894.0}\n"
AOS>{'Close': 71.27999877929488, 'Volume': 1484.0}
AOS>{'Close': 71.27999877929488, 'Volume': 1484.0}\n"
ABT>{'Close': 119.01000213623847, 'Volume': 3208.0}
ABT>{'Close': 119.01000213623847, 'Volume': 3208.0}\n"
AKAM>{'Close': 109.94999694824219, 'Volume': 785.0}
AKAM>{'Close': 109.94999694824219, 'Volume': 785.0}\n"
ACNO>{'Close': 291.4849853515625, 'Volume': 1232.0}
ACNO>{'Close': 291.4849853515625, 'Volume': 1232.0}\n"
ABBV>{'Close': 115.5250015258789, 'Volume': 3510.0}
ABBV>{'Close': 115.5250015258789, 'Volume': 3510.0}\n"
AAP>{'Close': 284.2180067138672, 'Volume': 554.0}
AAP>{'Close': 284.2180067138672, 'Volume': 554.0}\n"
ATVI>{'Close': 94.73999786376953, 'Volume': 15416.0}
ATVI>{'Close': 94.73999786376953, 'Volume': 15416.0}\n"
ALK>{'Close': 68.13999938964844, 'Volume': 243.0}
ALK>{'Close': 68.13999938964844, 'Volume': 243.0}\n"
ALB>{'Close': 161.60000618351562, 'Volume': 1197.0}
ALB>{'Close': 161.60000618351562, 'Volume': 1197.0}\n"
AES>{'Close': 25.829999923786855, 'Volume': 2131.0}
AES>{'Close': 25.829999923786855, 'Volume': 2131.0}\n"

junkaliong ~ ubuntu@ip-172-31-51-133: ~/CS205_working -- ssh -i ~/.ssh/CS205-key.pem ubuntu@18.234.193.25 -- 147x49
ubuntu@ip-172-31-51-133:~/CS205_working$ spark-submit stock_processing.py
21/05/07 17:58:18 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
2021-05-07 17:58:17:320850: W tensorflow/stream_executor/platform/default/dso_loader.cc:60] Could not load dynamic library 'libcudart.so.11.0'; dle
rrior: libcudart.so.11.0: cannot open shared object file: No such file or directory
2021-05-07 17:58:17:320859: I tensorflow/stream_executor/cuda/cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set up on you
r machine.
Using Spark's default log4j profile: org/apache/spark/log4j-defaults.properties
21/05/07 17:58:18 INFO SparkContext: Running Spark version 3.1.1
21/05/07 17:58:18 INFO ResourceUtils: =====
21/05/07 17:58:18 INFO ResourceUtils: No custom resources configured for spark.driver.
21/05/07 17:58:18 INFO ResourceUtils: =====
21/05/07 17:58:18 INFO SparkContext: Submitted application: StockStreamApp
21/05/07 17:58:18 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor:
, memory -> name: memory, amount: 1024, script: , vendor: , offHeap -> name: offHeap, amount: 0, script: , vendor: ), task resources: Map(cpus ->
name: cpus, amount: 1.0)
21/05/07 17:58:18 INFO ResourceProfile: Limiting resource is cpu
21/05/07 17:58:18 INFO ResourceProfileManager: Added ResourceProfile id: 0
21/05/07 17:58:18 INFO SecurityManager: Changing view acls to: ubuntu
21/05/07 17:58:18 INFO SecurityManager: Changing modify acls to: ubuntu
21/05/07 17:58:18 INFO SecurityManager: Changing view acls groups to:
21/05/07 17:58:18 INFO SecurityManager: Changing modify acls groups to:
21/05/07 17:58:18 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: Set(ubuntu); grou
ps with view permissions: Set(); users with modify permissions: Set(ubuntu); groups with modify permissions: Set()
21/05/07 17:58:18 INFO Utils: Successfully started service 'sparkDriver' on port 41179.
21/05/07 17:58:18 INFO SparkEnv: Registering MapOutputTracker
21/05/07 17:58:18 INFO SparkEnv: Registering BlockManagerMaster
21/05/07 17:58:18 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
21/05/07 17:58:18 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
21/05/07 17:58:18 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
21/05/07 17:58:18 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-a8fcd5dd-f956-43a2-8b65-78c94bf86026
21/05/07 17:58:18 INFO MemoryStore: MemoryStore started with capacity 366.3 MiB
21/05/07 17:58:18 INFO SparkEnv: Registering OutputCommitCoordinator
21/05/07 17:58:19 INFO Utils: Successfully started service 'SparkUI' on port 4040.
21/05/07 17:58:19 INFO SparkUI: Bound SparkUI to 0.0.0.0, and started at http://ip-172-31-51-133.ec2.internal:4040
21/05/07 17:58:19 INFO Executor: Starting executor ID driver on host ip-172-31-51-133.ec2.internal
21/05/07 17:58:19 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 37285.
21/05/07 17:58:19 INFO NettyBlockTransferService: Server created on ip-172-31-51-133.ec2.internal:37285
21/05/07 17:58:19 INFO SparkEnv: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
21/05/07 17:58:19 INFO BlockManagerMaster: Registering BlockManagerId(driver, ip-172-31-51-133.ec2.internal, 37285, None)
21/05/07 17:58:19 INFO BlockManagerMasterEndpoint: Registering block manager ip-172-31-51-133.ec2.internal:37285 with 366.3 MiB RAM, BlockManagerId
(driver, ip-172-31-51-133.ec2.internal, 37285, None)
21/05/07 17:58:19 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, ip-172-31-51-133.ec2.internal, 37285, None)
21/05/07 17:58:19 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, ip-172-31-51-133.ec2.internal, 37285, None)
datastream RDD received:
```

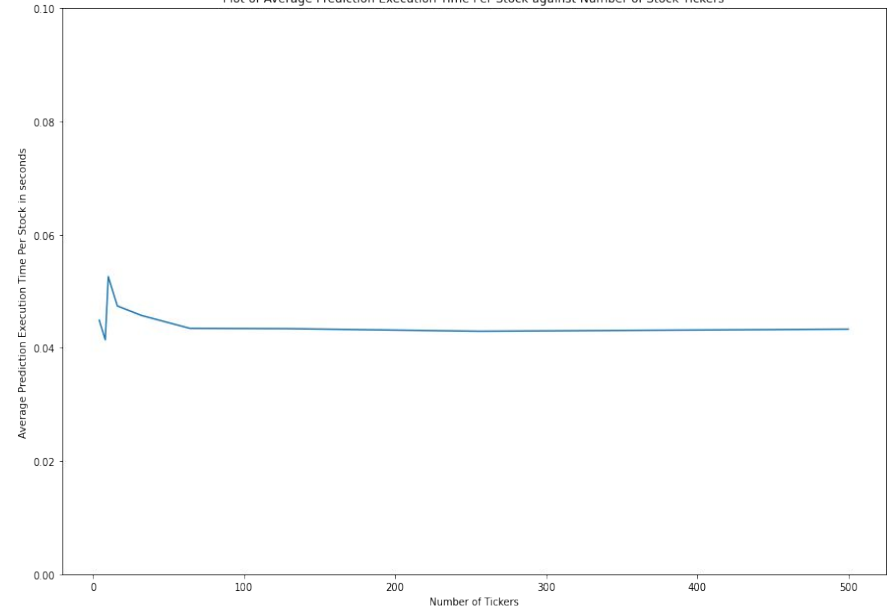


Real-Time Prediction Execution Time

Plot of Prediction Execution Time against Number of Stock Tickers



Plot of Average Prediction Execution Time Per Stock against Number of Stock Tickers



Infrastructure (AWS)

Data Processing:

- t2.2xlarge instances

Model Training

- g3s.xlarge & g3.8xlarge instances
 - NVIDIA Tesla M60 GPU
 - Horovod
 - OpenMPI

Prediction:

- t2.2xlarge instances

Storage:

- S3 Standard



Lessons and Future Directions

- Interaction with external APIs or data sources may set parameters for parallelism outside of the theoretical domain.
- The garden of forking paths is strong:
 - Many ways to develop parallel programs within these constraints.
 - Important to evaluate resources in conjunction with program design.
- Increase the resolution of prediction from 5 minutes ahead to 2 minutes ahead
- A potential future direction is to experiment with the use of Elephas for the prediction phase
 - Potentially allow us to perform model prediction directly with the RDDs received
- Future exploration of using an even smaller machine for prediction
 - Helps to increase access to larger group of end-users and reduce cost



Thank You!



Links to Project Resources

Project Website and GitHub Repository:

https://github.com/vrsivananda/CS205_FinalProject

Project Presentation Video Download:

https://github.com/vrsivananda/CS205_FinalProject/blob/master/presentations/Group11_CS205_FinalProjectPresentation.mp4

Project Presentation Slides Download:

https://github.com/vrsivananda/CS205_FinalProject/blob/master/presentations/Final%20Project%20Presentation%20Submission.pdf

