2019 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)

Financial Quantitative Big Data Platform based on High Performance Computing

Yongze Sun
University of Chinese Academy of Sciences
Computer Network Information Center,
Chinese Academy of Sciences
Beijing, China
sunyongze@outlook.com

Zhonghua Lu
Computer Network Information Center
Chinese Academy of Sciences
zhlu@sccas.cn

***Abstract*—A big data platform to for financial quantitative data is designed and implemented on HPC system. Key technologies including data storage mechanism and distributed computing framework are resolved. Based on the platform, several important feature for financial quantitative strategy research is developed, which are indicator computing, large scale backtest and distributed hyperparameter tuning. Tests shows that the platform can achieve much higher performance than single PC program, and can be used to design strategy base on large scale financial data.**

Keywords—Financial Quantitative Data, Big Data Platform, HPC, Parallel Computing, Quantitative Trading

I. INTRODUCTION

In financial quantitative trading area, traditional research and applications are mostly based on daily indicators, which means for a single securities or asset, only one data point is generated for each day. However, with the development of intraday trading and high frequency trading, we have to process and analyze data of finer grain, based on which to build quantitative trading strategy, that means we have to process much larger scale data than traditional methods. Taking Chinese Stocking Market as example, the number of daily price indicator for the whole markets is around 80 thousands for one year, which can be processed on a single pc with serial programs. Accordingly, there are more than 7 billion transaction records generated in one year, which need 2 TB storage space. Based on fine grained and larger scale data, we can extract much more information and market pattern. If we implement big financial data process and analysis with high performance and efficiency, it will help us extract value from market through building quantitative trading strategy with more return.

To solve the big data problem of financial data, an infrastructural platform which can storage large scale data and capable of data-intensive computing task, have to be built up first. For data-intensive computing task, the most popular solution is the Hadoop[1] eco-system, including MapReduce which is a distributed computing framework, Spark which is a in-memory distributed computing framework, etc. Especially, MapReduce and Spark[2], which are the two most popular distributed computing framework[3-5] for data-intensive task, have constructed parallel programming model for big data, making programming big data program much easier.

Yet traditional high performance computing systems, including super computer, are designed for computing-intensive task, like molecular dynamics simulation and atmospheric simulation. Classic parallel programming frameworks, like MPI and Charm++, are not suitable for data-intensive task to solve big data problem. In order to deal with big data through HPC systems, we must overcome challenges including data storage and deployment of distributed computing framework on HPC system. Related research like transplant open-source big data framework for HPC system[6] have attracted great intention.

In this article, aiming to provide complete solution for process and analysis of financial quantitative big data, we build a platform on a supercomputer system, "Explane", which is governed by Chinese Academy of Sciences. Main contributions of this article are listed below.

1. A highly efficient storage mechanism is built for HPC system.

2. Automatic deployment on HPC system of two distributed computing framework, Spark and Dask.

3. Based on the platform, important functions for financial quantitative research are implemented, including large indicator computing and analysis, large scale quantitative trading strategy test, hyperparameter tuning for quantitative trading. Whole procedure of financial quantitative research with big data can be done on the platform.

II. OVERALL DESIGH OF PLATFORM

We implement the target platform on the supercomputer "Explane". Storing data on the filesystem of "Explain", and using allocated computing nodes by LSF scheduler to deploy computing frameworks and applications, are basic ideas of the platform. "Explain" is a supercomputer system deployed in 2015, which adopt LSF as resource scheduler, and ParaStor2000 as distributed parallel filesystem. The peek speed of "Explane" is 2.36Pflops, the total memory is 140TB, and have 6.3 PB filesystem storage.

Key technologies of the platform includes how to store and manage large scale financial data, how to implement automatic deployment of computing framework with LSF scheduler, and how to implement important functions for quantitative analysis of big data. The overall structure of platform is shown in Figure 1.

978-1-7281-1664-8/19/$31.00 ©2019 IEEE
DOI 10.1109/CSE/EUC.2019.00065

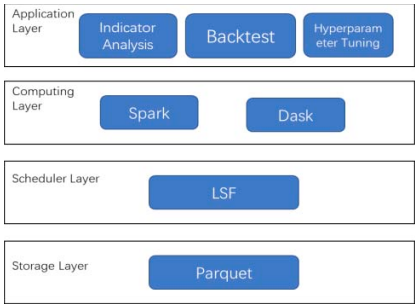

Figure 1: Overall Structure of Platform

III. KEY TECHNOLOGIES OF PLATFORM

A. Design and implenmentation of storage mechnism

The major kind of large scale data for quantitative financial research is Level 2 market data. The raw format of Level 2 market data is saved as csv file. All data from same day are stored in a same folder and each csv file hold daily records for one securities. This storage structure leads to more than ten thousands of small files for each trading day if we directly storage raw files on the supercomputer. And large amounts of small files will severely impact the performance of shared file system. To avoid this problem, raw data files will merged to less than 100 files each day of appropriate size and converted to Parquet format.

Parquet is column-oriented file format designed for big data analysis. Compared to row-oriented file format like csv, only necessary columns will be accessed when reading data. Data which mismatch query conditions can be skipped in file level. Compression is supported by Parquet format. All those character can reduce the IO cost of reding file, leading to higher performance and more convenient way of querying big data.

After raw data are uploaded to "Explane" every day, computing framework Spark will be used to process all daily data, including transaction records, order records and other type of financial quantitative data. Spark will merge these data and convert them to Parquet format, finally delete raw data files. When query data, parquet files will be accessed through type and date, and complicated query conditions can be used to filtering the data.

Actual test shows that the time consuming of reading a single day's data of all securities can be reduced to 40% by adopting parquet format compared to raw data, and side effect of large amounts of small files have been avoided.

B. Design and implenmentation of computing framework

Most financial data are structured data. Traditional analysis tools used by quantitative researchers are mostly python libraries. Pandas is one of the most common analysis tool, based on dataframe which is a table-like data structure. Considering user habits, we choose Spark and Dask as distributed computing frameworks of the platform. Both of them provide dataframe-like data structure and APIs, which don't require advanced parallel programming skills. Depending on either of the distributed computing task, large scale financial data can be processed and analyzed.

Spark is an in-memory distributed computing framework. As the most well-known distributed computing framework for big data, MapReduce[7] implemented parallel computing, fault tolerant, data spreading, data balancing and other necessary mechanisms. However, with the increasing of data scale and more complicated computing model, some problems of MapReduce emerged out, including excess of data IO leading to poor performance, especially for tasks that requires a number of iterative steps. Spark implemented a model called Resilient Distributed Datasets(RDD), which can take full advantage of cluster, reducing the IO caused by data copy and serialization. The performance of Spark is greatly improved compared to MapReduce. Based on the distributed computing framework, Spark also offers a series of tools, including GraphX for graph computing, Mlib for large scale machine learning. Adopting Spark as computing framework, make the platform capable of dealing with complicated task for large scale data. Typical structure of a spark cluster is shown in Figure 2.

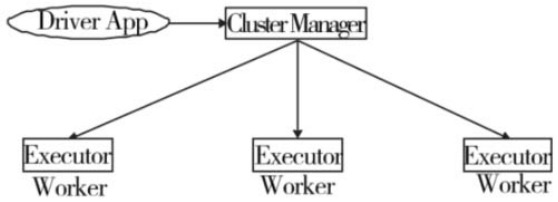

Figure2: Structure of Spark Cluster

In Spark cluster, the core node of the cluster is the cluster manager node, responsible for resource allocating and managing. Computing task is executed by worker nodes. Driver App translated tasks to a directed acyclic graph(DAG) and submitted it to the cluster manager.

Dask[8] is a parallel computing library in Python. Similar to Spark, Dask also translate computing task to a DAG, and distribute parallelizable tasks of each stage to worker nodes. Since Dask is completely built upon Python, python library can be easily integrated with Dask, such as frequently-used quantitative backtest library like zipline and pyalgotrader. Structure of a Dask cluster is shown in Figure 3.

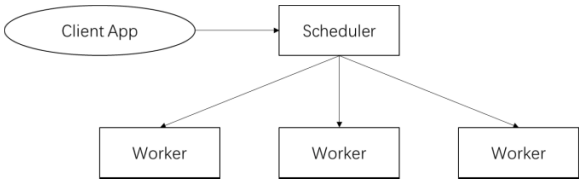

Figure 3: Structure of Dask cluster

From Figure 2 and Figure 3, we can conclude that Spark and Dask cluster have similar master-slave structure, that master node is responsible for scheduling and worker nodes is responsible for computing of sub tasks. Different from directly deploying Spark or Dask to a cluster, the deployment of our platform have to be combined with the LSF scheduler on "Explane" supercomputer. We will abstract the deployment procedure to a LSF job, apply for computing nodes through LSF and automatically deploy Spark and Dask by LSF scripts. Detailed steps are listed below.

1. Apply certain scale of computing resource from supercomputer scheduler. LSF will write hostnames of allocated node to environment variable "LSB_HOSTS",

2. Choose the first node as master node, and other nodes as worker nodes. Generate node list file for Spark and Dask,

3. Invoke the start scripts, and a Spark or Dask cluster will deployed on the supercomputer as a LSF job.

Through above steps, arbitrary scale of Spark and Dask cluster can be implemented as distributed computing framework for the platform.

IV. KEY FUNCTIONS

Based on the large scale level-2 market data stored on the platform, we implement there key functions for financial quantitative research, which are indicator computing and analysis, large scale backtest, and large scale hyperparameter tuning, forming procedures for quantitative trading strategy research and design.

A. Indicator computing and analysis

Utilizing the capability of processing large scale data from Spark and Das, we can compute and build indicator for whole market based on level 2 data, and apply statistical analysis like correlation analysis or hypothesis tests to these indicators. We have used the platform to compute transaction flow indicator for whole market of different time range for each transaction day.

B. Large scale strategy backtest

.Based on Dask and backtest library pyalgotrader, a distributed backtest framework is implemented on the platform, which is capable of parallel testing strategy performance on multiple securities, or different hyperparameter setting for one strategy, leading to high performance of large scale backtest. Main procedures of the distributed backtest framework are listed below.

1. A list of backtest task is built by the client app. Each single backtest task is expressed as a python function that invoke pyalgotrader inside,

2. The task list will be submitted to the scheduler node by the client app,

3. Dask allocates all tasks to Worker nodes to complete each backtest task,

4. The master nodes will collect results of all backtest tasks, generate backtest report and return it to client app.

C. Hyperparameter tuning for quantitative strategy design

Utilizing parallel backtest function, we further design and implement hyperparameter tuning function for strategy design. For a specific quantitative trading strategy, whether appropriate hyperparameter is adopted will decide the performance of the strategy. The most common way to get good sets of hyperparameter is to backtest different combination of hyperparameters on history data, and choose which produce the best performance measure. But with the increasing scale of history data or more complicated strategy, time costs for single group of hyperparameter will be considerable large, which makes using parallel computing to accelerate the search procedure necessary.

Besides search best hyperparameter setting using grid search method, the hyperparameter tuning problem can be viewed as a global optimization problem. Utilizing global optimization algorithm, we can further improve the efficiency of search hyperparameter with high performance. Several different hyperparameter tuning algorithm have been implemented on the platform, which are listed below.

1. Distributed grid search,

2. Distributed random search,

3. Distributed Bayesian optimization[9],

4. Distributed Optimization based on random respond surface method.

All above are asynchronous parallel algorithms, that can search parameter space with high efficiency.

V APPLICATION EFFECT OF PLATFORM

We test application effect of the Financial Quantitative Big Data Platform we have implemented. Performance of large scale indicator computing and hyperparameter tuning are tested. And a strategy for Chinese stocks index futures is developed based on the platform, resulting in satisfying return and sharp ratio.

A. Large scale indicator coputing

Based on level 2 transaction records data, the platform is used to calculate the money flow indicator of whole Chinese stock market each day in 2017. Spark on the platform is adopted. 10 nodes with 240 CPU cores is utilized to deploy the platform. Time consuming of platform and python program of a single PC is compared in Table 1.

Table 1: Performance comparison between platform and single core python program

	PC	Platform
Duration(minutes)	233285	18.5

From Tale 1, the python program on PC cost almost 3 days to process one year level 2 data, meanwhile the platform only took around 18 minutes.

B Hyperparameter tuning for trading strategy

To test the performance of hyperparameter tuning, a MACD strategy for Chinses commodity futures is adopted. This strategy has 7 hyperparameters which are stop loss rate, stop profit rate, holding time, indicator calculating frequency, short ema period, long ema period and difference smoothness scale. 10 nodes with 240 CPU cores is utilized to finish this test. Stop condition is to find sharp ratio greater than 1.4(Grid search need to finish searching all the grid points). Time consuming of platform and program on a single node PC is compared in Table 2.

Table 2: Performance comparison between platform and single node PC

	PC(s)	Platform(s)	speedup ratio
Grid Search	18033	84.66	213.004961

Random Search	7513.75	65.91	114.0001517
Bayes Optimization	4692.2	32.92	142.5334143
SurroOpt2	3877	31.26	124.0243122

From Table 2, it can be concluded that hyperparameter tuning tools based on the platform improve the efficiency greatly

C Use case

A quantitative strategy for Chinese stock indexes futures is developed based on the Financial Quantitative Big Data Platform. Following the procedure, level 2 indicator computing -> indicator statistical analysis -> strategy hyperparameter tuning. The strategy reach satisfying return performance on IF, IH and IC stock indexes futures, the sharp ratio of each are 4.73, 4.28 and 4.47 in backtest, and return curve are shown in Figure 4.

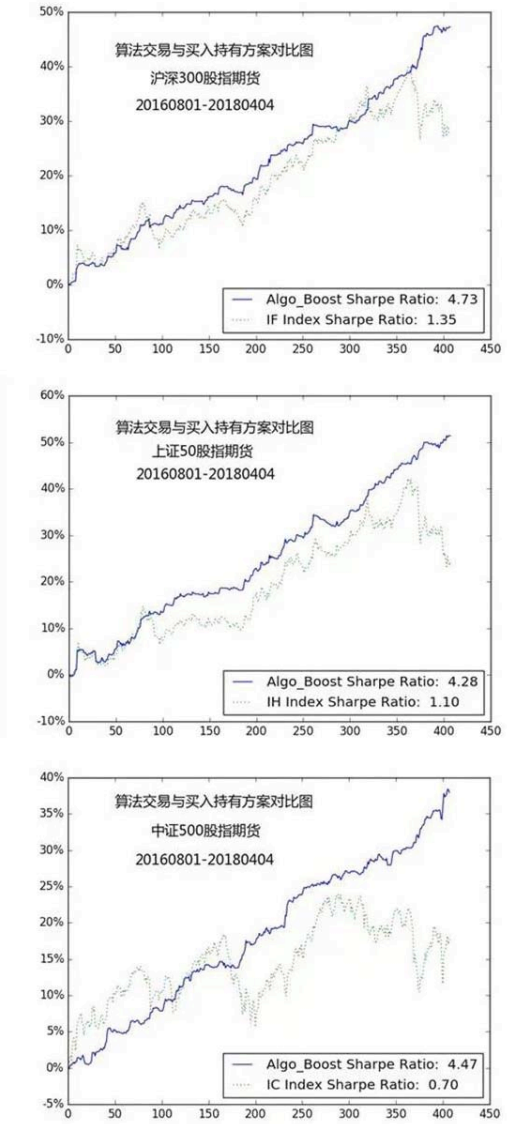

Figure 4: Return curve of strategy on IF, IH and IC

VI Conclusion and Future Work

In this article, a Financial Quantitative Big Data Platform is designed and implemented on HPC system. Storage mechanism of financial quantitative data, computing framework and key functions are implemented to form an intact solution for processing and analyzing large scale financial level 2 market data. Through this platform, the computing resource demand is solved and big data applications on HPC system is promoted.

In the future, we will further research more high performance financial applications based on the platform, including visualizations of large scale financial data, large scale portfolio optimization algorithms and so on. Another important direction is to efficiently processing high throughout real-time data based on HPC system, to address latency problem in intraday and high frequency trading.

Acknowledgment

To whom correspondence should be addressed. This work has been supported by the National Natural Science Foundation of China (Grant No. 61873254).

References

[1] SHVACHKO K, KUANG H, RADIA S, et al. The hadoop distributed file system; proceedings of the MSST, F, 2010 [C].

[2] ZAHARIA M, XIN R S, WENDELL P, et al. Apache spark: a unified engine for big data processing [J]. Communications of the ACM, 2016, 59(11): 56-65.

[3] TAYLOR R C. An overview of the Hadoop/MapReduce/HBase framework and its current applications in bioinformatics; proceedings of the BMC bioinformatics, F, 2010 [C]. BioMed Central.

[4] MALIK M, SASAN A, JOSHI R, et al. Characterizing Hadoop applications on microservers for performance and energy efficiency optimizations; proceedings of the 2016 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), F, 2016 [C]. IEEE.

[5] SHANAHAN J G, DAI L. Large scale distributed data science using apache spark; proceedings of the Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining, F, 2015 [C]. ACM.

[6] TOUS R, GOUNARIS A, TRIPIANA C, et al. Spark deployment and performance evaluation on the marenostrum supercomputer; proceedings of the 2015 IEEE International Conference on Big Data (Big Data), F, 2015 [C]. IEEE.

[7] DEAN J, GHEMAWAT S. MapReduce: simplified data processing on large clusters [J]. Communications of the ACM, 2008, 51(1): 107-13.

[8] ROCKLIN M. Dask: Parallel computation with blocked algorithms and task scheduling; proceedings of the

Proceedings of the 14th Python in Science Conference, F, 2015 [C]. Citeseer.
[9] KANDASAMY K, KRISHNAMURTHY A, SCHNEIDER J, et al. Asynchronous parallel Bayesian optimisation via thompson sampling [J]. arXiv preprint arXiv:170509236, 2017,