E-Fast & CloudPower: Towards High Performance Technical Analysis for Small Investors

Mircea Moca, Darie Moldovan, Oleg Lodygensky, Gilles Fedak

E-Fast & CloudPower: Towards High Performance Technical Analysis for Small Investors

Mircea Moca[1], Darie Moldovan[1], Oleg Lodygensky[2], and Gilles Fedak[3]

[1] Babeș-Bolyai University, Cluj-Napoca, România
{Mircea.Moca, Darie.Moldovan}@econ.ubbcluj.ro
[2] Laboratoire de l'Accélérateur Linéaire
oleg.lodygensky@lal.in2p3.fr
[3] INRIA, Université de Lyon, France
gilles.fedak@inria.fr

Abstract. About 80% of the financial market investors fail, the main reason for this being their poor investment decisions. Without advanced financial analysis tools and the knowledge to interpret the analysis, the investors can easily make irrational investment decisions. Moreover, investors are challenged by the dynamism of the market and a relatively large number of indicators that must be computed. In this paper we propose E-Fast, an innovative approach for on-line technical analysis for helping small investors to obtain a greater efficiency on the market by increasing their knowledge. The E-Fast technical analysis platform prototype relies on High Performance Computing (HPC), allowing to rapidly develop and extensively validate the most sophisticated finance analysis algorithms. In this work, we aim at demonstrating that the E-Fast implementation, based on the CloudPower HPC infrastructure, is able to provide small investors a realistic, low-cost and secure service that would otherwise be available only to the large financial institutions. We describe the architecture of our system and provide design insights. We present the results obtained with a real service implementation based on the Exponential Moving Average computational method, using CloudPower and Grid5000 for the computations' acceleration. We also elaborate a set of interesting challenges emerging from this work, as next steps towards high performance technical analysis for small investors.

Key words: E-Fast, CloudPower, Technical analysis, Moving averages, High Performance Computing

1 Introduction

The nowadays financial markets are amongst the most competitive markets in the world. More, the specific conditions of the financial markets are rapidly changing, so that efficient data analysis, fast and well grounded reactions to

market changes, multiple-source market data integration are actual challenges concerning investors. Obviously, the big players of the market like the large capitalized banks afford to address these challenges by developing and maintaining sophisticated systems relying on modern HPC infrastructures that provide advanced financial analysis capacities. More, they support research departments with specialists that are inter-disciplinary skilled in areas like maths, finance and programming. In contrast, for smaller players that do not have access to the above described resources it is almost impossible to efficiently perform in these market conditions. The poor investment decisions lead to about 80% of the financial market investors to fail [1].

In this paper we propose E-Fast, an innovative platform prototype for online technical analysis for helping small investors to obtain a greater efficiency on the market by increasing their knowledge. The prototype relies on High Performance Computing (HPC), allowing to rapidly develop and extensively validate sophisticated finance analysis algorithms. The novelty stands in the initiative of building a real technical analysis solution based on HPC for helping small investors to cope with hash market challenges. Our approach is designed with the potential of being sustainable through the specific communities' efforts. For instance, we use technologies (like XtremWeb middleware) that allow an easy aggregation of computing resources from the participating individuals.

In this work, we aim at demonstrating that the E-Fast implementation, based on the CloudPower HPC infrastructure, is able to provide small investors a low-cost and meaningful service that would otherwise be available only to the largest finance institutions. We describe the architecture of our system and provide design insights. We also present the results obtained with a real implementation of the Exponential Moving Average computational method, using CloudPower and Grid5000 for the computations' acceleration. We also elaborate a set of interesting challenges emerging from this work and explain our envisioned approach.

Our efforts also come to meet a set of main objectives of the European strategy for the 2014/2020 timeframe [2], focusing on the following three priorities:

- Intelligent and sustainable growth by developing a knowledge and initiative based economy.
- Sustain a competitive and efficient economy that relies on its own resources.
- Social and territorial cohesion support. In this context, direct extensions of our approach can support the building of small investor communities.

The remainder of this paper is organised as follows. Section 2 gives the background of our work, section 3 presents the architecture of the implemented E-Fast prototype, while section 4 presents the experiments run with the prototype and discusses results. Further, section 5 presents related work, section 6 explains relevant future challenges and section 7 concludes this work.

2 Background

In this section we describe the context of our work and define the concepts used in our discussion. In our approach, the user interacts with the system through a set of services. Such a **service** is characterized by:

- Service name,
- Input structure,
- Result (output) structure, and
- Other specific parameters.

The user aims at executing one or more services in order to analyze their specific results. The results can either be directly interpreted by the user or further used into third party analysis tools. The logic of a particular service is given by a stand alone or a mix of **computational method**(s). Examples of methods may be moving averages, genetic algorithms, data-mining algorithms, Monte Carlo simulations and others. For each service, the system expects to a strict definition regarding the input and output data.

A service **execution** is the process of executing the computational method(s) underlying that service on particular input and other user-defined parameters. An execution yields a result (output) with specific semantic and format. The execution is managed by the E-Fast server and may imply the delegation of the effective computational method execution to the HPC infrastructure (see system architecture in section 3.1). The general execution flow for a particular service is detailed in section 3.2.

2.1 Moving Averages

In this work we built a service that is simply defined by the Exponential Moving Average computational method. Forward we give a short description of this method and explain its use in technical analysis.

The employment of moving averages in technical analysis is already a common approach. Their simplicity and efficiency made them successful in industry and economics even before being applied in the financial area. By definition, the moving average (denoted by $MA(k)$) is a method to smooth the variability in time of the analyzed time series by considering the average of the most recent k observations. In technical analysis, to provide a trading signal, two moving averages for the prices of a certain stock are calculated. One of them has a larger number of observations k_l, in order to capture the **long term** trend and the other with a smaller k_s, which is more sensitive to the **recent fluctuations** in price. When the value of the short term moving average passes above the long term moving average, we have a buy signal, suggesting that the price will increase more. On the contrary, when the short term moving average crosses the long term moving average by dropping below its value, it signals sell, forecasting a drop in the price of the stock.

There are several versions of the method, like simple (SMA) and exponential (EMA) [3, 4] moving averages. The simple moving average can be criticized because gives the same importance to all considered observations. In contrast, the exponential moving average gives a higher importance to the recent observations, gradually decreasing the importance for the older ones.

The calculation of EMA is given in eq. 1, where $EMA(k)$ is the exponential moving average calculated for k periods, a is the smoothing constant, showing the importance weight to give to each component and P is the price of the stock at a certain moment.

$$EMA(k) = \alpha P_k + \alpha(1-\alpha)P_{k-1} + \alpha(1-\alpha)^2 P_{k-2} + ... + \alpha(1-\alpha)^{k-1} P_1 \quad (1)$$

The moving averages can be used in technical analysis to calculate trading signals (buy or sell). For this, the moving average must be calculated twice, with different values for k. We call long term EMA (EMA_l) the one calculated on a bigger k and short term EMA (EMA_s) the other one.

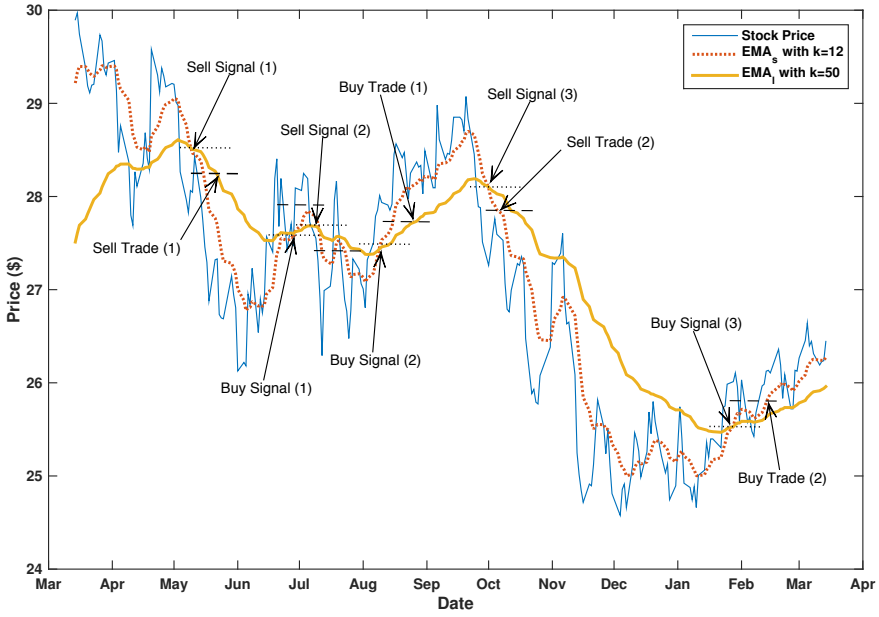

Fig. 1. Stock price, and computed moving averages and signals.

For a better understanding of using EMA in this context, in the following we provide an example, depicted in figure 1. For a particular stock, we show the

calculated moving averages and the buy/sell signals, also marking the trading points. The chart contains data from March 2012 to March 2013 on the stock price of Microsoft, where $k = 12$ for EMA_s and $k = 50$ for EMA_l. We marked every intersection between the lines of the EMAs with a dot line. Depending on the stock's behavior on the market, this method can yield a large number of buy/sell signals. In practice, not all of the generated signals are worthy to be considered. Hence, in order to avoid noisy trading signals, generated by two or more very close in time crossovers between the two lines, we used a supplementary filter (represented by the dash line). Only when EMA_s passes the filter, a trade is generated. We also mention that our strategy permits both long and short positions and we do not discuss the profitability of this particular example.

Going into detail, we notice that the first signal (in May 2012) is a sell signal. The transaction Sell Trade (1) only occurs after the value of the EMA_s dropped below the filter value. After this transaction, we own a short position. Next we have the Sell Signal (2), which does not lead to to a trade since EMA_s doesn't surpasses the filter. Forward, another sell signal is generated, but even if the value of EMA_s went bellow the filter value, the signal is ignored since we already own a short position (and we wait for a buy signal). The following signal, Buy Signal (2) flags the opportunity to buy, which occurs at the moment showed by Buy Trade (1) arrow, in this way closing the short position. Sell Signal(3) is shortly followed by Sell Trade (2), opening a new short position, which is closed only several months later, at Buy Trade (2). Although the strategy is quite simple, finding efficient values for its parameters (like k, the smoothing factor α or the size of the noise filter) can be difficult. In fact, they are very sensitive to the trends inside the analyzed time series, making it difficult for the investor to find the best solution. For this reason, an almost exhaustive search for the best parameter combinations for a strategy that works usually only for a particular stock can be very costly in terms of computational time.

2.2 CloudPower

Big Data and High Performance Computing (HPC) are key factors in knowledge and innovation in many fields of industry and service, with high economic and social issues: aerospace, finance and business intelligence, energy and environment, chemicals and materials, medicine and biology, digital art and games, Web and social networks.

Today, acquiring high-end data centers and supercomputers is very expensive, making Big Data and HPC unreachable to small business for their research and development. For reasons both technical and economic, access to such technologies is difficult to fundamental actors of growth that are small and medium-sized innovative companies. That is why it is important to support them in this process. This a fundamental tool for competitiveness and innovation capacity of service and industry enterprises.

The CloudPower project offers a low cost Cloud HPC/BigData service for small and medium-sized innovative companies. With CloudPower, companies and scientists will run their simulations to design and develop new products on

a powerful, scalable, economical, reliable and secure infrastructure. CloudPower leverages on the open-source software XtremWeb-HEP previously developed by the CNRS and INRIA in France. The principle of the technology is to collect the under-exploited resources on the Internet and Data Centers to build a virtual supercomputer providing HPC and Big Data services on demand. CloudPower is supported by the French National Research Agency (ANR).

3 The E-Fast prototype

In this section we give a detailed description of the proposed E-Fast prototype, focusing on its architecture as well as the service definition and execution.

3.1 System architecture

In this subsection we present the architecture of the implemented E-Fast prototype. This mainly contains three components: the E-Fast client and server and the distributed computing infrastructure.

Figure 2 depicts the architecture of the prototype. First, the E-Fast client runs on the users' machines and allows them to connect to the E-Fast server and *consume* its services. This is a web interface that allows the client to set up the parameters of the execution and to inspect and analyze the results obtained from the executions. Based on the execution parameters received from client, the server extracts the appropriate input data from the database and transfers it to the distributed computing infrastructure. Then, it delegates to the infrastructure the execution of the service on the respective input data. After receiving the results from the infrastructure, the server composes the final result from a user's perspective and delivers it through the client. The database managed by the server mainly contains: financial market historical data, results of the executions and execution meta data.

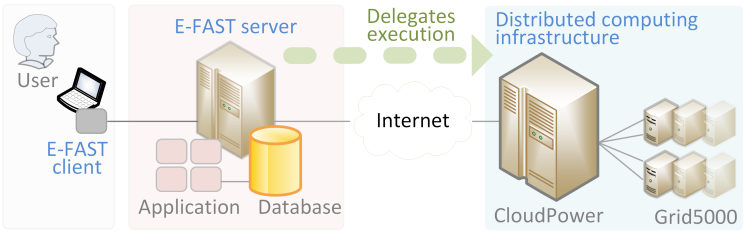

Fig. 2. Overview of the E-Fast prototype architecture.

3.2 Service execution flow

In this subsection we detail the overall approach for a service execution.

Figure 3 presents the general execution flow for a particular service. The process begins with the user, specifying the execution parameters through the E-Fast client. Based on these parameters the E-Fast server makes the decision whether to start a new execution or directly deliver the results corresponding to the received parameters. The direct delivery would occur when the user has already performed an execution for the same parameters. If it's not this case, the E-Fast server selects from the database the appropriate input data (according to the computational method defining the required service and the user's preferences). Forward, according to the computation distribution strategy of the service, the server creates a set of input files and uploads them to CloudPower. Then, the server starts the execution of the service by creating and launching a job on the DCI for each input file. After a while, the E-Fast server receives the results from CloudPower, aggregates them into a final result and presents them to the user within the E-Fast client.

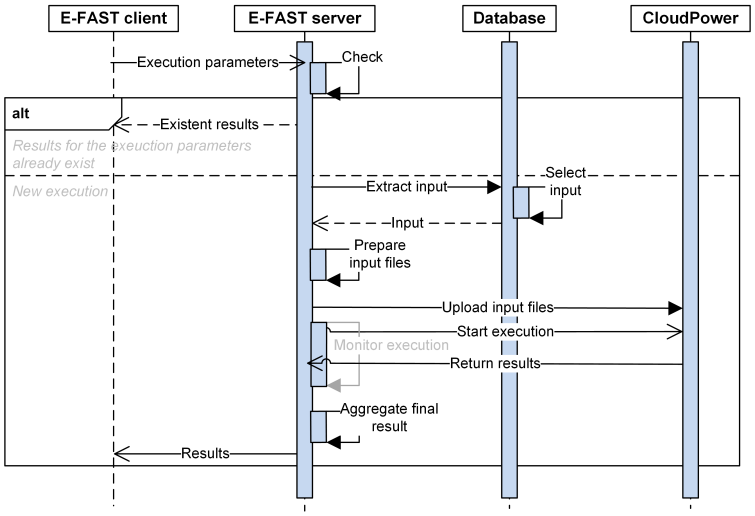

Fig. 3. General service execution flow.

3.3 Computational method distribution

As discussed in section 1, the computational methods used in technical analysis are generally computational intensive. In this sense, our implementation relies on CloudPower for accelerating the service execution. This is capable of executing bag of tasks, these being independent one from another. In this context, for

any given computational method, this aspect brings the challenge of defining a **computation distribution strategy**. This is significantly determined by the method's definition, so when a high execution performance (i.e. in terms of makespan) of the service must be attained, then the distribution strategy may be critical. In this case, it must be tailored to the method's definition. In the following we call **distribution criterion** a concept based on which a system designer defines the distribution strategy.

Since the Moving Averages are computed on a single stock at a time and the obtained results are independent from those of other stocks, we chose the stock to be the distribution criterion. Hence, our implementation executes the computational method on time series, separately for each stock. For example, if the user wants to calculate the rentability of the $S\&P500$ index, then the system runs a task for each stock from the whole set of 502 stocks (this is our experimental data and it is described in section 4).

For each execution, the user specifies a time series from the available historical data. The real execution time of the service on a specific machine from the HPC infrastructure depends on the size of the specified time series. From this perspective, if one aims at optimizing the execution, then the distribution strategy can be changed - i.e. execute a task on a set of stocks, not only a single one. By this, a system designer could adjust the average real execution time of a task in order to fit the HPC infrastructure's requirements in terms of optimality.

3.4 Service architecture

In our approach, the functionality of a service is organized in two components: the **technical** and the **business** layers. While the technical layer implements one or more computational methods, the business layer computes business key-values (like profit), with direct utility for the financial analyst, based on the technical layer's output. Figure 4 depicts the service layers and parameter types. For a better understanding, we explain the parameter types that characterize the service directly referring to the Moving Averages method. Hence, the layers contain the following parameter types:

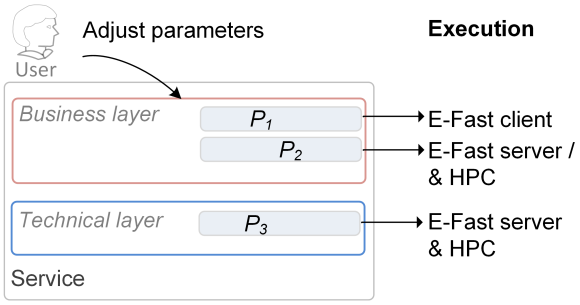

Fig. 4. Service general architecture.

- P_1 is a parameter type that directly affects the execution of the service on E-Fast client side. For instance, the user may vary the commission value applied to the gross profit. These computations have low complexity and can be directly supported by the E-Fast client and the input data is directly available on the E-Fast server.
- P_2 has direct impact on business but the implied computations are more complex compared to those of P_1, so they need to be performed by the E-Fast server with the eventual delegation to the HPC infrastructure. Such a parameter would be the filter used to compute the effective transactions based on the signals provided by the Moving Averages method. The adjustment of the filter may lead to a significantly different number of transactions, making the service more sensitive to the stock's volatility. Consequently, more transactions lead to a greater absolute value of the calculated trading fees. Hence, the impact of this parameter is important.
- P_3 is specific to the technical layer and is related to the optimization of the computational method and generally require the E-Fast server to coordinate the execution on the HPC infrastructure. For instance, finding optimal k_s and k_l values for the Moving Average method and particular market data implies the a costly execution of the service for a relevant set of k_s and k_l combinations on the HPC infrastructure.

4 Experiments and results

The **experimental data** used in this work was historical market data of the companies composing the S&P500 index for the 2006-2014 period. This is one of the major indices in the United States, covering around 75% of the equity market by capitalization. It comprises 502 stocks for which we collected daily pricing information.

In order to obtain the best parameters for the Exponential Moving Average trading strategy, one execution of the E-Fast service computes for one stock from S&P500 the return based on EMA for all combinations of k. More specifically, $k_s = \{2, 3, 4 \dots 50\}$ and $k_l = \{4, 6, 8, \dots 100\}$. This setup led us to a number of 496 603 profitability calculations for the whole S&P500 index. In our experiments we used the same noise filter value, but in a real system, the user must be able to tune this parameter.

Our main success criterion was the profit obtained by each combination. When calculating the profit we used the same percentage fees per trade to be extracted from the results of each sell-buy pair. Other criteria that can be used for evaluating the performance of a trading strategy can be used, depending on the investment objectives. Such criteria can be the maximum drawdown (the maximum loss of a strategy during its life), number of profitable trades / all trades, risk adjusted return, Sharpe ratio etc.

We compared the results obtained with our trading strategies against the buy and hold strategy on the S&P500 index. While the total return for the S&P500

for the entire analyzed period was a profit of 61.84%, the results of our strategies show a profit of 77.3%. Table 1 shows the returns obtained by the best 5 groups of strategies. We observe that there are several k_s and k_l combinations yielding significant returns, the differences among them being relevant. This demonstrates the importance of finding and using the right parameters for a certain strategy.

Strategy group	1	2	3	4	5
Return	77.3%	73.7%	71.3%	69.6%	68.5%

Table 1. Return for top 5 groups of strategies.

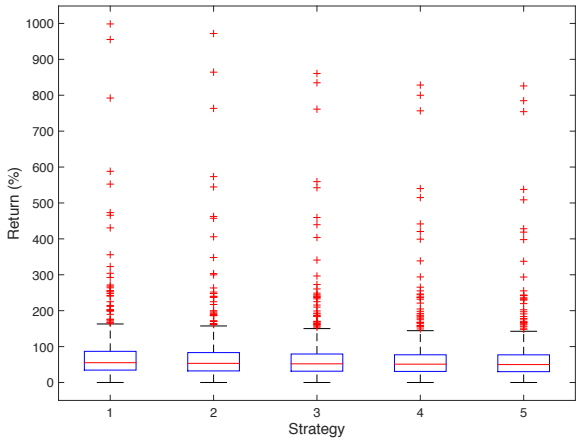

Fig. 5. Returns distribution for the S&P500 index.

In figure 5 we show the distribution of the obtained returns. We note that all strategies obtained positive returns, some of them with outstanding positive performance. These performances can be filtered and used as main target investments by the investors. Although they are shown as outliers, they are actually the most important strategies, the ones that provide the highest returns. For example, the best strategy for the Apple stock (AAPL) is obtained using the short term moving average with $k_s = 14$, long term moving average with $k_l = 52$ and a filter of 0.15%. While the main focus in our work is to show how E-Fast & CloudPower can be used together for solving computational intensive financial problems, we did not focus on refining and optimizing the business results.

In practice, the methods can be optimized by periodically running the service (e.g. weekly), so that it adapts to new, recent market data. Then, the user may retain the best parameter combination till next analysis.

5 Related work

The literature in the field of distributed and parallel computing in finance is not very rich in terms of diversity. Most of the related work is focused on solving particular computational finance problems involving Monte Carlo models for derivative pricing. Since the Black and Scholes [5] model was introduced, it was widely used, along with others, to compute derivatives prices. The iterative character of the method (usually it can reach 1 million runs) can be very consuming in terms of time and computation resources. This can be unpractical in the field of finance, where the result is expected in terms of minutes, seconds or close to real time.

Stokes-Res [6] discusses the parallelization and distribution of Monte Carlo option pricing in the context of several grid environments. Difficulties related to running parallel tasks on the grid were revealed, consisting in synchronization of the start task time and partitioning the computational load. They conclude that several other tests should be carried on this and other computational finance problems in order to have a better image on the performance that can be achieved. Heinecke [7] presents a scalable approach for the Black and Scholes solver for pricing options. They discuss the results obtained in the context of several hardware solutions, emphasizing the robustness of their proposed operator for addressing this particular issue.

In the context of finding the best trading rules by using technical analysis methods, Strassburg [7] employs parallel genetic algorithms. Due to the data dimension, the repetitive characteristics of the algorithm and time-critical answer need, the high computational resources are a must for the institutions and individuals dealing with sophisticated trading decision support methods. The paper presents the efforts to optimize the parameters of the technical indicators used in order to improve the trading results. The tests were conducted on the Madrid Stock Exchange Index data, but only on holding long positions (not allowing short selling) aiming to demonstrate the usefulness of the parallel approach. However, the tests were made on a single machine with four cores, limiting the conclusions to this closed environment. A solution to reduce the computation time for intensive operations in financial analysis was proposed by Moreno [8]. They built a disk memoization solution to reduce the number of repeated computations, allowing the analysts to build and debug their algorithms more quickly.

A new automated trading system architecture was introduced by Freitas [9], who considers dividing the trading problem into several tasks handled by distributed autonomous agents, with minimal central coordination. Their strategy is based on obtaining a consensus trading signal based on other trading signals from multiple strategies. The results were considered very satisfactory after testing moving average crossover strategies on a large database.

6 Future challenges

The work so far consisted in building the E-Fast prototype equipped with a first service based on the Exponential Moving Averages computational method. From both - technical and financial perspectives, the aim was to proof the concept of providing advanced technical analysis services based on distributed computing, for small investors. These achievements led us to a set of interesting challenges and motivates us to build further on this concept. The most relevant development directions on this topic are:

- Building **service composition mechanisms** that allow the user to combine several services in order to obtain advanced knowledge on the market. The interesting aspect of the services in this context is that based on the same market data, several computational methods can be applied in order to obtain similar or **complementary visions** and understanding of the market's behavior. For instance, the user can execute two services in order to obtain results with similar semantic but computed with different methods. In this case, the aim would be to compare the different results obtained for particular input data and assess a **degree of confidence** in them. In other situations, the user might be interested in **combining** two or more **services** by running them and using their outputs together in order to understand the market from different perspectives.
- Building advanced **collaboration mechanisms** to allow the different stakeholders work together and obtain synergy. As mentioned, players on the financial market need high expertise in fields like mathematics, finance and technical/computer science. In this sense we aim at building mechanisms that allow users to **share their resources** (i.e. computing, knowledge, expertise, skills). By this we aim at facilitating the crowding of mutually interested specialists, small investors and resource owners.
- Optimizing the prototype from two perspectives. For each service, from a technical viewpoint, the system designer should optimally decide the computations performed within the two technical and business layers of the service (presented in section 3.4). This is very challenging, since the design decision is specific to each computational method. More, the solution must take into account the specifics of the HPC infrastructure and, at the same time be parametric, in order to allow the user to choose amongst several service flavours. From a financial perspective, the system must keep services up-to-date from a business viewpoint. As explained in section 4, optimal key-parameter values must be found for a computational method in order to efficiently adapt to the recent market behavior. In this sense, the challenge would be to build fully or quasi-automatic mechanisms that keep the services up-to-date and adapt to market fluctuations.

7 Conclusions

In this paper, we proposed E-FAST, a service prototype for on-line technical analysis that can support small investors to obtain a greater efficiency on the market by increasing their knowledge. In this work we created a prototype that relies on High Performance Computing (HPC). This allows one to rapidly develop and extensively validate sophisticated finance analysis algorithms. We aimed at demonstrating that E-Fast based on the CloudPower HPC infrastructure is able to provide small investors a scalable, low-cost and secure service that would otherwise be available only to the largest financial institutions. We presented the architecture of our system. We also presented the results obtained with a real implementation of the Exponential Moving Average computational method, using CloudPower and Grid5000 for the computations' acceleration. Finally we presented a set of interesting challenges emerging from this work and describe our approach to address them.

References

1. Barber, B.M., Odean, T.: Chapter 22 - the behavior of individual investors. Volume 2, Part B of Handbook of the Economics of Finance. Elsevier (2013) 1533 – 1570
2. : "europe 2020 - a strategy for smart, sustainable and inclusive growth". Technical report, "European Commission" (2010)
3. Holt, C.C.: Forecasting seasonals and trends by exponentially weighted moving averages. International Journal of Forecasting **20**(1) (2004) 5 – 10
4. Cox, D.R.: Prediction by exponentially weighted moving averages and related methods. Journal of the Royal Statistical Society. Series B (Methodological) **23**(2) (1961) pp. 414–422
5. Black, F., Scholes, M.: The pricing of options and corporate liabilities. The journal of political economy (1973) 637–654
6. Stokes-Ress, I., Baude, F., Doan, V.D., Bossy, M.: Managing parallel and distributed monte carlo simulations for computational finance in a grid environment. In: Grid Computing, Springer (2009) 183–204
7. Heinecke, A., Jepsen, J., Bungartz, H.J.: Many-core architectures boost the pricing of basket options on adaptive sparse grids. In: Proceedings of the 6th Workshop on High Performance Computational Finance, ACM (2013) 1
8. Moreno, A., Balch, T.: Speeding up large-scale financial recomputation with memoization. In: Proceedings of the 7th Workshop on High Performance Computational Finance. WHPCF '14, Piscataway, NJ, USA, IEEE Press (2014) 17–22
9. Freitas, F.D., Freitas, C.D., De Souza, A.F.: System architecture for on-line optimization of automated trading strategies. In: Proceedings of the 6th Workshop on High Performance Computational Finance. WHPCF '13, New York, NY, USA, ACM (2013) 4:1–4:8