Machine Learning-based Elastic Cloud Resource Provisioning in the Solvency II Framework

Andrea La Rizza*‡, Giuseppe Casarano‡, Gilberto Castellani†‡,
Bruno Ciciani*, Luca Passalacqua† and Alessandro Pellegrini*
*Department of Computer, Control and Management Engineering – Sapienza, University of Rome
Email: larizza.1252622@studenti.uniroma1.it, {pellegrini,ciciani}@dis.uniroma1.it
†Department of Statistical Sciences – Sapienza, University of Rome
Email: {luca.passalacqua,gilberto.castellani}@uniroma1.it
‡Alef S.r.l.
Email: {andrea.larizza,giuseppe.casarano,gilberto.castellani}@alef.it

***Abstract*—The Solvency II Directive (Directive 2009/138/EC) is a European Directive issued in November 2009 and effective from January 2016, which has been enacted by the European Union to regulate the insurance and reinsurance sector through the discipline of risk management. Solvency II requires European insurance companies to conduct consistent evaluation and continuous monitoring of risks—a process which is computationally complex and extremely resource-intensive. To this end, companies are required to equip themselves with adequate IT infrastructures, facing a significant outlay.**

In this paper we present the design and the development of a Machine Learning-based approach to transparently deploy on a cloud environment the most resource-intensive portion of the Solvency II-related computation. Our proposal targets DISAR®, a Solvency II-oriented system initially designed to work on a grid of conventional computers. We show how our solution allows to reduce the overall expenses associated with the computation, without hampering the privacy of the companies' data (making it suitable for conventional public cloud environments), and allowing to meet the strict temporal requirements required by the Directive. Additionally, the system is organized as a self-optimizing loop, which allows to use information gathered from actual (useful) computations, thus requiring a shorter training phase. We present an experimental study conducted on Amazon EC2 to assess the validity and the efficiency of our proposal.

I. Introduction

The European Directive 2009/138 (Solvency II) [1] requires insurance undertakings to evaluate technical provisions in a market-consistent way and to measure the Solvency Capital Requirement (SCR) with the Value-at-Risk approach measured with at 99.5% confidence level over a 1 year unwinding period. Moreover, it is expected that insurance undertakings own an effective risk-management system comprinsing strategies, processes and reporting procedures necessary to identify, measure, monitor, manage, and report on a continuous basis the risk, at the individual and at an aggregated level. Risk depends on all the sources the company is or could be exposed, and their interdependencies ([1], art. 44). It is possible to identify at least five relevant areas that should be covered by such a system: underwriting and reserving, asset-liability management, investment, liquidity/concentration risk management, and risk-mitigation techniques. The features deriving from the Directive become significantly resource-intensive when the undertaking, in addition to the so-called *standard formula* approach detailed in the Directive, calculates technical provisions and SCR using an internal model, either full or partial ([1], art. 112). The internal model is a system used by the undertaking to assess risks and to determine the overall solvency needs, that is supposed to ensure better quality standards by the description of idiosyncratic exposure to risk, that is however subject to the approval of the national supervisory authority ([1], art. 112–127).

A recent proposal [9] has addressed the computational problems deriving from Solvency II compliance in the context of Italian life insurance, introducing DISAR®, a commercial system originally designed to work on a grid of conventional computers. DISAR tackles market-consistent valuation of the complex cash flows using numerical techniques in a stochastic framework, namely Monte Carlo simulation, on a fine-grained time grid. Nevertheless, the underlying assumption behind the efficiency of DISAR is the availability of large-scale computing infrastructures. Due to the periodical nature of the computations related to the Directive, setting up large clusters can easily be seen as a cost-ineffective solution by most companies which are required to adhere to the Directive.

The cloud computing paradigm [14], which allows to use virtualized resources in lieu of physical ones, is now universally recognized as a means to significantly reduce the costs associated with running IT premises, adopting the pay-as-you-go model. Recently, the financial world has shown as well a growing interest towards this paradigm [12]. In this paper we follow this track, and propose a solution to deploy Solvency II-related computations on the cloud, by relying on a distributed version of DISAR. Our proposal has several benefits: i) the deploy of the computation is completely transparent to the end user, so that no actual modification to the original workflow is required; ii) the overall cost faced by companies to comply with the Solvency II Directive is significantly reduced, as dedicated hardware is no longer required; iii) the actual pay-as-you-go cost is optimized, so that our proposal is able to fine tune the amount of resources required for the computation, depending on input data and the characteristics of the available virtualized architectures.

As for the above point iii, we explicitly rely on a Machine Learning (ML)-based system to determine what is the best configuration of the distributed deploy. In particular, we explicitly take into account the *time* required to carry out the Solvency II-related computations, and the expected *cost* to use virtualized resources. Therefore, by using our approach, companies are able to meet the stringent time requirements imposed by the EU Directive, while minimizing the associated

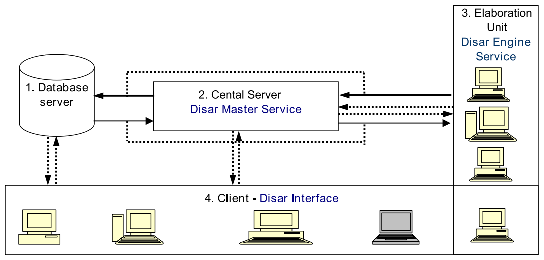

Fig. 1. The DISAR Architecture.

outlay. Moreover, we have organized our system as a self-optimizing loop [7], which allows us to use the data obtained while carrying out useful actual computations to enlarge the knowledge base used by our ML-based prediction models. In this way, every computation that is carried out by a company is used as well to give better predictions for later deploys. This allows to significantly reduce the training phase of the system, thus further reducing the costs associated with the utilization of cloud-based virtualized services. To the best of our knowledge, this is the first attempt at making the requirements of the Solvency II Directive more feasible and more cost-effective, at the same time.

The remainder of this paper is structured as follows. In Section II we give an overview of the DISAR System. Section III discusses the ML-based approach, the used prediction models, and the transparent deploy system. An experimental assessment of our proposal is shown in Section IV.

II. The DISAR® System

The Dynamic Investment Strategy with Accounting Rules (DISAR) System [9], [8] is designed for the evaluation and control of minimum-guaranteed profit-sharing life policies indexed to the returns of dedicated funds (*segregated funds*). It is based on *market-consistent* evaluation criteria under uncertainty in a general *asset-liability management* (ALM) framework. In particular, DISAR considers accounting rules that control the segregated fund budget and the management strategy for the financial contracts portfolio. This is done using a stochastic model considering several sources of financial uncertainty such as interest rate, equity, currency and credit/default risk, in addition to sources of actuarial risks such as longevity/mortality and lapse. Actuarial risks are assumed to be mutually independent, while financial risks are possibly correlated.

DISAR is aimed to the evaluation of risks of profit-sharing life polices, that are the main type of life polices sold in Italy, taking into account the details of the cash flows generated by the contracts. For illustrative purposes, let us consider the very simple case of a single premium pure endowment insurance contract, written a time $t = 0$ on a life aged x, with term T years and initial insured sum C_0, focusing on financial risks. The value at time T of the *benefits* promised by the insurance undertaking to the subscriber can be expressed as:

$$Y_T = C_0 \Phi_T \mathbb{1}_{\{E(T)\}}, \tag{1}$$

where the indicator of the random event $E(T)$ takes into account the actuarial uncertainty (survival/lapse of the subscriber) and Φ_T represents the *readjustment factor* defined as:

$$\Phi_T = \prod_{t=1}^{T}(1+\rho_t) = (1+i)^{-T} \prod_{t=1}^{T}\Big(1+\max\{\beta I_t, i\}\Big), \tag{2}$$

where ρ_t is the *readjustment rate* defined as:

$$\rho_t = \frac{\max\{\beta I_t, i\} - i}{1+i}, \tag{3}$$

$\beta \in (0,1)$ and $i \geq 0$ are respectively known as the *participation coefficient* and the *technical rate* and contractually specified, while $I_t \in \mathbb{R}$—the return of the segregated fund—is a random variable. Letting F_t be the value of the segregated fund at time t (when the premium payed by the subscriber is invested), then the return rate earned by the fund in the period $[t-1, t]$ is:

$$I_t = \frac{F_t}{F_{t-1}} - 1, \quad t = 1, 2, ..., T. \tag{4}$$

When $I_t > i$, a fraction of the return earned in excess of i is credited to the subscriber by increasing the insured sum, so that:

$$C_t = C_{t-1}(1+\rho_t), \quad t = 1, 2, ..., T, \tag{5}$$

A key point is that F_t is not necessarily the market value of the fund, but could be a *book value* (*i.e.* a value that depends only on prices at which assets have been bought and sold), so that the volatility of returns can be strategically controlled by the manager of the segregated fund. Therefore, a proper risk management system should take into account the way the segregated fund is managed.

Equation (1) is fundamental to understand which aspects liability depends on. In general, valuation of risk requires to compute the distribution of the value Y_t at time t of the random variable Y_T (in the Solvency II framework $t = 1$) on which the risk measure is defined. When actuarial risks are also taken into account, the indicator in (1) is replaced by a proper expression and the determination of Y_t becomes more complex, although the general framework remains unchanged.

In DISAR, the distribution of Y_t is determined using nested Monte Carlo simulations [10]. The Monte Carlo technique is used to produce *real-world* or *natural* scenarios corresponding to the possible future evolution of the financial and actuarial risk drivers. For each real-world scenario, a second-stage (nested) Monte Carlo set of scenarios is simulated according to *risk-neutral* probabilities, that are used to obtain the future value of contingent-claim contracts by properly taking into account the premia investors require to face risks. The use of the risk-neutral setting is a standard technique that ensures the correct market-consistent valuation under the hypothesis of absence of arbitrage on the market. A nested Monte Carlo simulation is thus a two stage procedure in which:

1) n_P independent sample paths of the risk drivers are generated from $t = 0$ to $t = 1$ under the real world measure $\mathbb{P}$, conditionally to the information available at time $t = 0$, *i.e.* to the filtration $\mathcal{F}_0$;
2) for each of the n_P paths, n_Q independent sample paths from $t = 1$ to $t = T$ are generated under risk-neutral probability $\mathbb{Q}$, conditional to the filtration $\mathcal{F}_1$.

The set of n_P simulations are referred to as *outer* simulations, while the n_Q ones as *inner* ones. Using this approach, the computation is therefore composed of a $n_P \times n_Q$ simulations, each simulation being composed of the trajectory of each risk driver up to time T, that can easily be as large as 100 years, given actual human lifetime extension. The number of inner and outer simulations should be chosen in order to achieve an adequate precision on the 99.5% quantile of Y_t. If n_Q is too small, a bias is introduced in the determination of the quantile of Y_t, while if n_P is too small the statistical error affecting the determination of the quantile is too large. As a result, the number of total simulations may be impressive. Moreover, the computational effort required to evaluate liability cash flows in each single outer scenario could be very high, depending of the features of the policies involved.

Since the procedure is particularly time-consuming, within DISAR the computational burden is parallelised and distributed. Moreover, the number of inner simulations can be strongly reduced if the so-called *Least Square Monte Carlo* (LSMC) technique is used [5]. With LSMC, the plain Monte Carlo determination of Y_t is replaced by a truncated series expansion in orthonormal polynomials, whose parameters are calibrated with a $n'_P \times n'_Q$ smaller sample obtained by plain nested Monte Carlo simulation. Also in that case the computation is performed in a parallel and distributed way.

From the architectural point of view, DISAR has a client/server distributed organization, which is shown in Figure 1, allowing to concurrently use various workstations. Its main components are:

1) A Database Server, hosting a Relational Database Management System.
2) A Master Server, hosting the Disar Master Service (DiMaS).
3) A set of Computing Units: each unit hosts the Disar Engine Service (DiEng) that manages the Disar Actuarial Engine (DiActEng) and the Disar Asset-Liability Management Engine (DiAlmEng).
4) A set of Clients, each hosting the Disar Interface (DiInt) that allows to set computational parameters and monitors the progress of the elaborations.

DISAR allows an efficient parallelization of the computation because it relies on *elementary elaborations blocks* (EEB), which are a set of elaborations identified by common characteristics that make them identical from the point of view of risks. In particular, two types of EEBs are considered: A) *actuarial valuation*, namely the computation of actuarial-expected cash-flows generated by the contracts, and B) *Asset-Liability Management valuation*, that is the evaluation of market consistent values of contracts.

When the computations is started, DiMaS divides all the input data in EEBs, thus it acts as the orchestrator of the system. It defines as well the elementary elaboration blocks, estimates the complexity of the elaborations, establishes the elaboration schedule, distributes the elementary requests to the processing units and monitors the process. The DiEng component on each node delivers the elaboration to DiActEng or to DiAlmEng depending on the elaboration type:

- DiActEng carries on the computation of type-A EEBs, namely it operates on the policy portfolio related to the segregated funds, it receives as input the contractual information, the consistency of policies and the technical information, and it computes on the related schedule the aggregate *probabilized flows* related to net performance, without loss of information;
- DiAlmEng is in charge of type-B EEBs. It operates on the policy portfolio related to the segregated funds, receiving as input the contractual information, the accounting information, the probabilized flows computed by the DiActEng, the financial hypothesis on the market structure, the features of the management strategy and produces the characteristic quantities useful to evaluate and to manage the risk.

III. The ML-Based Transparent Deploy System

The most time-consuming activities of DISAR are related to type-B EEBs. Since these activities are based on Monte Carlo simulations, they can be parallelized by distributing different work units on the available computing nodes, let them be physical or virtual. Each node computes concurrently average local values, which are then suitably combined to produce the final global results. Since the amount of data to be processed in this phase of the computation is unbounded, the data scattering and gathering can be efficiently supported using Message Passing primitives, such as the Message Passing Interface (MPI) [19].

This data-separation approach is particularly effective in the Solvency II scenario, since aggregation of locally-computed values can be carried out only at the end of local simulations, although the duration of each type-B EEB is not related to the duration of the other ones. As an additional effect, the data used by type-B EEB, although related to the assets of specific companies, do not allow to gather useful information on them. In fact, since the DB is not exported to the cloud, the n_Q inner simulations are *anonymized data*, thus perfectly suitable for processing on a public environment.

We base our transparent deploy system on Starcluster [18], a tool which allows to activate any number of VMs on Amazon EC2. Whenever the user of DISAR starts a new simulation, the interface automatically activates the required number of VMs. The distributed nature of DISAR perfectly fits this deploy system, as every VM will run part of the computation.

To reduce the cost of the simulation, we rely on a set of Machine Learning-based prediction models. These models are integrated into the DISAR interface modules to determine what is the most time- and cost-effective deploy on a cloud-based environment. In particular, the user of the system can specify a set of available virtualized architectures, along with its capabilities (in terms of, e.g., CPU power, and RAM) and cost per hour. This information is stored in a database which is then coupled with runtime data. Whenever a new simulation is run, the system stores the execution time into the database.

To build our ML-based execution time prediction models, we rely on Weka [20], a framework to integrate various ML algorithms with Java-based applications. In particular, we have selected *Multi-Layer Perceptron* [15] (MLP), *Random Trees* (RT) and *Random Forests* (RF) [6], *IBk* [2], *KStar* [13], and *Decision Tables* (DT) [3]. Additionally, we have experimentally selected the characteristic parameters relative to each EEB that induce the highest variability in the execution time of the simulation, namely the number of *representative contracts*—that is, the policies with equal insurance parameters

Algorithm 1 Selection of the best-suited configuration.

```
$X = \{MLP, RT, RF, IBk, KStar, DT\}$
$M = \{...\}$                    ▷ The set of available virtual hardware configurations
$T_{max}$
$N = [1, max]$
procedure PREDICT(CharacteristicParameters f)
    $C = \emptyset$                    ▷ The set of feasible deploys
    for $n \in N$ do
        for $m \in M$ do
            for $x \in X$ do
                $time_x \leftarrow p_x(m, n, f)$
            end for
            $time \leftarrow \frac{\sum_{i \in |X|} time_x}{|X|}$
            if $time \leq T_{max}$ then
                $cost \leftarrow hour\_cost \cdot time$
                $C \leftarrow C \cup \langle m, n, cost \rangle$
            end if
        end for
    end for
    if RAND( ) $< \varepsilon$ then
        $selected \leftarrow$ random element in $C$
    else
        $selected \leftarrow \min_{cost} C$
    end ifreturn $selected$
end procedure
```

(same readjustment rate parameters, same age, gender, etc), the maximum time horizon of the policies, the segregated fund asset number and the number of financial risk-factors. Each execution of a DISAR simulation involves different values of these parameters, depending on the specific company asset which is taken into account. Therefore, whenever a simulation is executed on the cloud, the total execution time is stored into the database along with the values for the above parameters.

We therefore re-train the ML-based models after each execution of the DISAR simulation. Since the duration of each simulation could be significantly long, this approach allows to refine the prediction models while carrying out useful work. Moreover, this approach allows to increase the knowledge base used by the prediction models in a way which is independent of the actual company the simulation is run for. In fact, although the parameters could vary from one company to the other, they are not necessarily bound to a specific one. In this way, refining the prediction models for a given company could provide benefits for Solvency II simulations of different ones.

The knowledge base is then exploited to find a suitable configuration for the deploy on a cloud-based environment. As mentioned before, two constraints should be taken into account, namely the *total expected time* for a given simulation, and the *total expected cost*. We define a *family of prediction models* P which is composed of all the prediction models $p_x : M \times \mathbb{N} \times F \rightarrow \mathbb{R}^+$, where $x \in \{MLP, RT, RF, IBk, KStar, DT\}$ represents the ML algorithm used to build the prediction p, M is the domain of virtualized architectures that can be used to instantiate a VM, the value $n \in \mathbb{N}$ represents the number of instantiated VMs, and F is the set of characteristic parameters of interest for a given computation. The co-domain of each p_x is the expected execution time on the given deploy configuration.

We therefore evaluate every p_x on all the available configurations $m \in M$, and all the natural values in the range $[1, max]$, where max is a maximum threshold that can be specified by the user of the system. To account for possible prediction errors by the various models p_x, we compute a final value $time$ for a given virtualized configuration (namely, virtualized hardware and number of instances) as the average of all the times predicted by the models. This allows to reduce the impact of prediction errors by some of the models, a situation which is expected only at the beginning of the system's lifetime, when the data in the knowledge base is reduced.

Through the DISAR interface, the user can select a maximum execution time T_{max} which is the timing constraint required to carry on the simulation, so as to respect the constraint imposed by the Solvency II Directive. Therefore, all the configurations such that $p_x(m, n, f) > T_{max} \quad \forall x$ are simply discarded. Then, given the fact that each virtualized architecture is associated with a per-hour cost, every configuration $\langle m, n \rangle$ is associated with an additional cost parameter c which tells the expected expenditure to run the simulation on the configuration. Therefore, given all the tuples $\langle m, n, c \rangle$ we select the one associated with the minimum cost c. This allows us to meet the time constraints while selecting the lowest cost possible. We emphasize that this approach explores very different configurations, in which less powerful virtualized architectures could be selected in place of more powerful ones, provided that they allow to meet the time constraints.

To ensure that all the configurations are suitably explored, after filtering out all the configurations such that $p_x(m, n, f) > T_{max}$, with a small probability ε we select a random configuration. This allows to enlarge the knowledge base, possibly reducing the number of false positives on the expected execution time. The complete algorithm to computed the deploy organization is shown in Algorithm 1. As a side note, our DISAR interface allows to supersede the ML-based predicted configuration, so as to allow an early manual training phase, which could be used to artificially grow the knowledge base at the beginning of the lifetime of the system.

Finally, we emphasize that since the ultimate goal of our proposal is to reduce the overall cost of the simulation, having some nodes which finish their local computation far before other ones could be a significant issue. In fact, the nodes which have already completed their tasks would be idle until the slowest one completes, just to execute the data gathering procedure. Cloud-based deploys would increase their cost with no benefit. Our approach allows to transparently capture this issue, without relying on more time-consuming scheduling of MPI-based computations. In fact, configurations which involve a large number of nodes which are idle most of the time are immediately discarded thanks to the models learned by the ML algorithms.

IV. Experimental Assessment

In this Section we present an experimental assessment [17] of the validity and applicability of our approach, carried out on the Amazon AWS infrastructure. We have selected three portfolios mimicking typical Italian insurance company ones, choosing 15 different EEBs. We have set the number of risk-neutral iterations to 50 for all the simulations—a value that introduces an acceptable statistical error within the LSMC approach. Concerning the natural iterations, we have fixed their value to 1,000 for illustrative purposes (typical required values are on the order of 10,000 to 100,000). The virtualized architectures which we have used in our experimentation are:

- `m4.4xlarge`: 16 vCPUs, 64 GiB of RAM;
- `m4.10xlarge`: 40 vCPUs, 160 GiB of RAM;

- `c3.4xlarge`: 16 vCPUs, 30 GiB of RAM;
- `c3.8xlarge`: 32 vCPUs, 60 GiB of RAM;
- `c4.4xlarge`: 16 vCPUs, 30 GiB of RAM;
- `c4.8xlarge`: 36 vCPUs, 60 GiB of RAM.

In order to quantify the goodness of the approach, we show the difference between a predicted execution time and its actual value $\bar{\delta}$, which in turn affects the predicted cost of the cloud-based deploy. We explicitly study this distance on the distributions' tails.

Let us denote a real execution time—which is only known after a simulation is carried out—as Θ, and the value estimated by the prediction model p_x selected according to Algorithm 1 as $\hat{\Theta}$. The distance $\bar{\delta}$ can be calculated as:

$$\bar{\delta} = \frac{1}{N} \sum_{i=1}^{N} \left(\hat{\Theta}_i - \Theta_i \right), \tag{6}$$

where N is the number of samples for each architectural configuration. This value tells how much on average the estimated values deviate from the real ones. Moreover, it allows to understand if the algorithm over or underestimates the execution time. Yet, assessing the validity using only $\bar{\delta}$ is not enough, as outlying points could significantly affect the validity of the prediction. In particular, while an overestimation only implies a higher outlay, an underestimation might violate the timing constraints which are fundamental to meet the deadlines imposed by the Directive.

In Table I we report the average error $\bar{\delta}$ by the models when the knowledge base is composed of around 1500 samples. The results show that the selected ML algorithms show a high accuracy despite the limited size of the training data—compared to total execution times which can last up to several hours—and are therefore suitable for our purpose. This conclusion is backed as well by the data shown in Figure 2, which shows the discrepancy between the time predicted and actual execution time. By the plot we can see that the point cloud is clustered along the theoretical line, which is associated with the ideal model which is able to predict execution time with a 100% accuracy. This is confirmed by Figure 3, where it is shown that around 80% of the predictions have an absolute error smaller than 200 seconds. Higher errors are related to configurations with a small number of samples in the training dataset.

Concerning the performance of our distributed approach, we report in Figure 4 average speedup data for our cloud-based deploy versus a sequential execution of the simulation. By the results, we can see that the cloud-based deploys offer a performance gain which is non-negligible, and therefore—if we associate the obtained speedup with the large number of simulations which are required by the Directive—they allow to meet the stringent time constraints associated with the Solvency II Directive. Moreover, Table II reports the average cost associated with one simulation on each of the selected virtualized infrastructures. The total experiments for this paper, which are composed of 1500 runs, entailed a total cost of 128$. This is an outlay significantly smaller than the cost of any modern high-end computer grid.

As a final comparison, we have forced the execution of a large configuration on the higher-end VM and on the most cost-effective one. Our ML-based prediction selected configurations for the same input data which show a cost

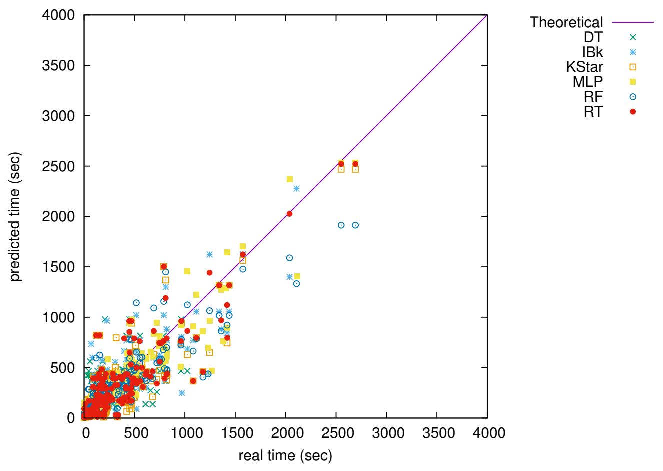

Fig. 2. Plot of real and estimated execution time.

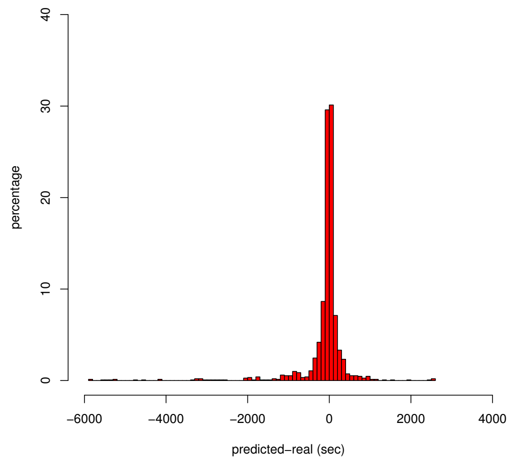

Fig. 3. Distribution of the error associated with the used prediction models.

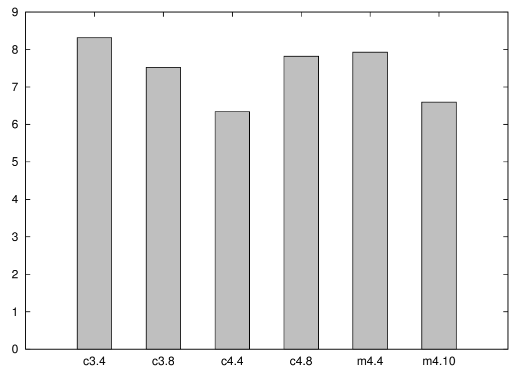

Fig. 4. Speedup of the cloud-based execution wrt the sequential one.

TABLE I
$\bar{\delta}$ REPORTED BY EACH CLASSIFIER ON EACH OF THE SIX TRAINING SET WITH A 40%-60% SPLITTING PERCENTAGE, IN SECONDS.

	m4.4.xlarge	m4.10.xlarge	c3.4.xlarge	c3.8.xlarge	c4.4.xlarge	c4.8.xlarge
IBk	-34.9	-229	-8	12	-241	-253
KStar	-5.3	1.5	1.5	-95	-131	-189
RT	50.0	-156	-13	-85	-186	-279
RF	20.6	-208	-30	-74	-232	-256
MLP	72.0	-28	-113	-30	-161	-227
DT	9.7	-80	63	-43	-317	-293

TABLE II
PER-SIMULATION AVERAGE COST.

m4.4.xlarge	0.052$
m4.10.xlarge	0.120$
c3.4.xlarge	0.041$
c3.8.xlarge	0.121$
c4.4.xlarge	0.066$
c4.8.xlarge	0.086$

decrease up to 54% with respect to the higher-end machine, and an execution time reduction up to 48% with respect to the most cost-effective one. This result supports our claim that our proposal can reduce the outlay associated with Solvency II-related computations, still allowing to meet the time requirements of the Directive.

V. Related Work

The literature is significantly lacking of ML-based approaches to optimize financial applications on cloud environments. At a more broader scope, the works in [4], [16] use ML-based prediction models to determine, at runtime, the resource demand of web-based applications. Although similar in spirit to our proposal, we target a closed system model which has a limited lifetime, and therefore requires a more accurate determination of the needed resources. On the other hand, the target of [4], [16] is an open model entailing an application which could be run indefinitely long. Therefore, we explicitly use more ML-based models at once, while the works in [4], [16] use a single model at a time, although in a repeated fashion.

The work in [11] adopts different analytic approaches to determine an optimal resource provision of cloud-based applications. The analytic approach allows to provision computing resources for stages as well as a long-term plan. Contrarily, we target the resource provisioning in a less dynamic environment, which is nevertheless high resource-intensive, and we do this relying on ML techniques to develop the models, rather than using analytic instruments.

VI. Conclusions and Future Work

In this paper we have presented a ML-based system to transparently deploy on a cloud environment computations related to the Solvency II Directive. Using our approach, it is possible to meet the stringent requirements of the EU Directive, while keeping the outlay associated with the computation low. In particular, we have experimentally shown how our solution allows to accurately predict cloud-based execution times, and is able to explore a large number of different virtualized architectures, so as to account for their different prices.

So far, our system considers homogeneous deploys, namely it does not consider the possibility of employing VMs instantiated using different virtualized hardware configurations. Introducing this additional variability aspect will be the subject of future work.

References

[1] Directive 2009/138/EC of the European Parliament and of the Council of 25 November 2009 on the taking-up and pursuit of the business (Solvency II) of Insurance and Reinsurance. *Official Journal of the European Union*, L351(1), 2009.

[2] D. W. Aha, D. Kibler, and M. K. Albert. Instance-based learning algorithms. *Machine Learning*, 6(1):37–66, 1991.

[3] E. Alpaydin. *Introduction to Machine Learning*. 3rd edition, 2014.

[4] A. a. Bankole and S. a. Ajila. Predicting cloud resource provisioning using machine learning techniques. In *2013 26th IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, pages 1–4. IEEE, may 2013.

[5] D. Bauer, A. Reuss, and D. Singer. On the calculation of the Solvency II Capital Requirement based on Nested simulations. *ASTIN Bulletin*, 42(2):453–501, 2012.

[6] L. Breiman. Random Forests. *Machine Learning*, 45(1):5–32, 2001.

[7] B. A. Caprarescu and D. Petcu. A self-organizing feedback loop for autonomic computing. *Computation World: Future Computing, Service Computation, Adaptive, Content, Cognitive, Patterns, ComputationWorld 2009*, pages 126–131, 2009.

[8] G. Casarano, G. Castellani, L. Passalacqua, F. Perla, and P. Zanetti. Relevant applications of Monte Carlo simulation in Solvency II. *Soft Computing*, pages 1–12, September 2015.

[9] G. Castellani and L. Passalacqua. Applications of Distributed and Parallel Computing in the Solvency II Framework: The DISAR System. In G. et al., editor, *Euro-Par 2010 Parallel Processing Workshops*, volume 6586 of *Lecture Notes in Computer Science*, pages 413–421. Springer Berlin Heidelberg, 2011.

[10] T. Cazenave and N. Jouandeau. Parallel Nested Monte-Carlo search. In *Proc. IEEE Int. Parallel Distrib. Processes Symp.*, pages 1–6, 2009.

[11] S. Chaisiri, B.-S. Lee, and D. Niyato. Optimization of Resource Provisioning Cost in Cloud Computing. *IEEE Transactions on Services Computing*, 5(2):164–177, apr 2012.

[12] R. Gill. Why Cloud Computing Matters to Finance. *Strategic Finance*, (January):43–48, 2011.

[13] G. Gins, B. Pluymers, I. Smets, J. Espinosa, J. Van Impe, and P. Perner. Advances in Data Mining. Applications and Theoretical Aspects. *Advances in Data Mining. Applications and Theoretical Aspects*, 6870(July 2015):314–328, 2011.

[14] B. Hayes. Cloud computing. *Communications of the ACM*, 51(7):9, jul 2008.

[15] S. Haykin. Neural networks expand SP's horizons. *IEEE Signal Processing Magazine*, 13(2):24–49, mar 1996.

[16] S. Islam, J. Keung, K. Lee, and A. Liu. Empirical prediction models for adaptive resource provisioning in the cloud. *Future Generation Computer Systems*, 28(1):155–162, jan 2012.

[17] A. La Rizza. Elastic Cloud Resources Provisioning for Life Insurance Undertaking Applications. Master's thesis, Sapienza, University of Rome, Rome, Italy, 2015.

[18] MIT. Starcluster. http://star.mit.edu/cluster/.

[19] MPI Forum. Message Passing Interface Forum. http://www.mpi-forum.org/, 1994.

[20] I. H. Witten, E. Frank, L. E. Trigg, M. A. Hall, G. Holmes, and S. J. Cunningham. Weka: Practical machine learning tools and techniques with Java implementations. In *Proc ICONIP/ANZIIS/ANNES99 Future Directions for Intelligent Systems and Information Sciences*, 1999.