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

Andrea La Rizza[1,2]
Giuseppe Casarano[2]
Gilberto Castellani[1,2]
Bruno Ciciani[1]
Luca Passalacqua[1]
Alessandro Pellegrini[1]

[1] Sapienza, University of Rome
[2] Alef S.r.l

DCPerf 2016

Rationale

- In 2009, the European Union introduced the Solvency II Directive
- All EU insurance companies have to periodically assess their *risk*
- This is an extremely complex and resource-intensive task
  - technical provisions must be evaluated in a market-consistent way
  - value at risk measured with 99.5% confidence over 1 year unwinding
  - risk depends on all the sources the company could be exposed

- Companies have been required to equip with adequate (costly) IT infrastructures
- The Directive became effective on January 2016

The Goals

1. Move Solvency II-related computation to the cloud
2. Make this migration as transparent as possible to the user
3. Reduce the overall cost faced by companies to enforce Solvency II requirements
4. Fine tune the amount of computing resources taken from the cloud to meet Solvency II time requirements (QoS)
5. Ensure complete data privacy

DISAR—Dynamic Investment Strategy with Accounting Rules

- DISAR targets the evaluation and control of minimum-guaranteed profit-sharing life policies in Italy
  - It is based on market-consistent evaluation criteria under uncertainty in a general asset-liability management framework
  - It relies on a stochastic model considering several sources of financial uncertainty and actuarial risks

- Example: single premium pure endowment insurance contract, focusing on financial risks
- The value at time T of the benefits promised by the insurance are:

$$Y_t = C_o \Phi_T \mathbb{1}_{\{E(T)\}}$$

High Performance Dependable Computing Systems

DISAR—Dynamic Investment Strategy with Accounting Rules

- Φ_T is a readjustment factor:

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

- ρ_t is the readjustment rate:

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

- Valuation of risk requires to compute the distribution of the value Y_t at time t of the random variable Y_T
- The distribution of Y_t is determined using nested Monte Carlo
  - For each real-world scenario, a second-stage Monte Carlo set of scenarios is generated

Parallelizing DISAR

- DISAR relies on *elementary elaboration blocks* (EEBs):
  - they share common characteristics
  - they are identical from the point of view of risk
  - their computation is based on Monte Carlo simulation
- Monte Carlo simulation can be distributed on multiple nodes
- Locally-computed results are then combined together
- Data scatter/gather can be supported using Message Passing primitives
- EEBs are anonymized data

Deploying DISAR on the cloud

- MPI-based nature of EEB computation makes it easy to orchestrate computation on the cloud
  - Starcluster is a valuable tool to technically make it possible

- Determining the best amount of resources is not a trivial task
- We rely on Machine Learning to predict the best-suited amount of VM instances to:
  - meet time requirements related to Solvency II directive
  - keep companies outlay low

- 6 different predictors evaluated: Multi-Layer Perceptron, Random Trees, Random Forests, IBk, KStar, Decision Tables

Deploying DISAR on the cloud

- We populate an execution time database every time a computation is completed
  - This is independent of the actual company
- We define a *family of prediction models* P, where each $p_x : M \times \mathbb{N} \times F \to \mathbb{R}^+$
  - M is the domain of available virtualized architectures
  - $n \in \mathbb{N}$ is the number of instantiated VMs
  - F is the set of parameters of interest of the model
  - x defines the ML algorithm used
- We evaluate each p_x on the whole domain (n is thresholded by the user) and compute the average value on x

Deploying DISAR on the cloud

- A T_{max} threshold specifies the maximum time constraint for the computation
  - Any $\bar{p}_x(m, n, f) > T_{max}$ is discarded
- Each VM instance $m \in M$ is associated with a per-hour cost, which is mapped to the global computation cost c
- Among all the tuples $\langle m, n, c \rangle$, we select the one with lowest cost c

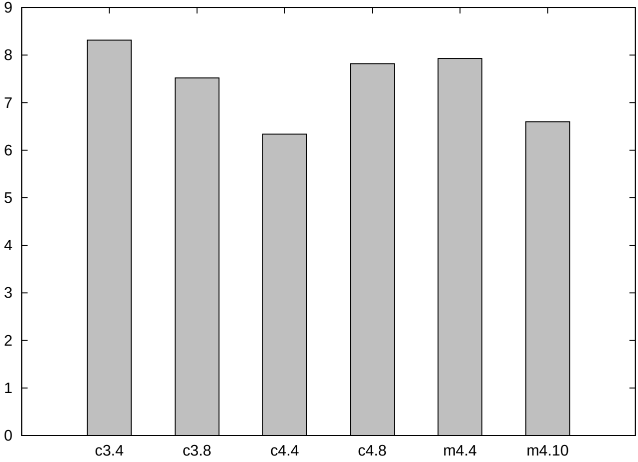
- To account for *exploration*, we enforce ε-greedy policy
- We anyhow ensure the T_{max} constraint

Experimental Assessment

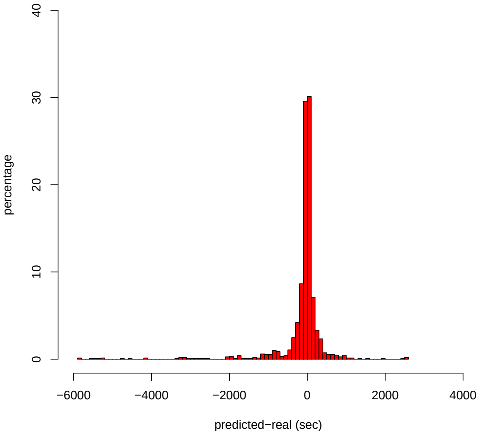
- We have used 3 real-world (Italian) portfolios
- We have picked 6 different virtualized infrastructures from Amazon:
  - Different allocated computing power
  - Different cost per hour
- We focus on prediction error and performance speedup

- Immediate results: the total experimentation is made of:
  - 1500 different runs
  - Total cost is 128$ (way less than any high-end computing grid!)
- "Forced" executions give rise to:
  - Cost decrease up to 54%
  - Time reduction up to 48%

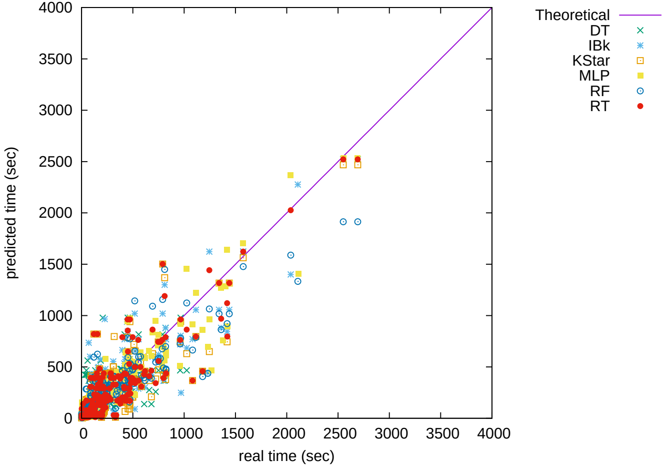
Execution speedup



Prediction Accuracy



Prediction Accuracy



Thanks for your attention

どもありがとうございます

Questions?

pellegrini@dis.uniroma1.it
http://www.dis.uniroma1.it/∼pellegrini