SAPIENZA
UNIVERSITÀ DI ROMA

Elastic cloud resources provisioning for life insurance undertaking applications.

School of Engineering in Computer Science

Master of Science in Engineering in Computer Science

Candidate

Andrea La Rizza
ID number 1252622

Thesis Advisor

Prof. Bruno Ciciani

Co-Advisor

Ing. Alessandro Pellegrini

Academic Year 2014/2015

Thesis defended on 23 October 2015
in front of a Board of Examiners composed by:

Prof. Bruno Ciciani (chairman)
Prof. Fabrizio D' Amore
Prof. Francesco Delli Priscoli
Prof. Leonardo Lanari
Prof. Paolo Liberatore
Prof. Francesco Quaglia
Prof. Marilena Vendittelli
Prof. Andrea Vitaletti

Elastic cloud resources provisioning for life insurance undertaking applications.
Master thesis. Sapienza – University of Rome

This thesis has been typeset by LaTeX and the Sapthesis class.

Version: January 17, 2016

Author's email: andrelr89@gmail.com

to all those
who believe that
nothing is impossible...

Abstract

The **Solvency II** Directive (Directive 2009/138/EC) [14] is a European law adopted in November 2009, and amended by Directive 2014/51/EU of the European Parliament and of the Council of 16 April 2014 (the so-called "Omnibus II Directive"). It was enacted by the European Union to regulate the insurance sector through risk management. Solvency II requires that European insurance companies conduct consistent evaluations and constant monitoring of risk. The main regulation requires the insurance undertaking to compute the **SCR (Solvency Capital Requirement)** and the **pdf (Probability Distribution Forecast)**. The SCR is the capital amount that should hedge the losses up to a given level. To calculate the SCR the directive allows the company to choose between two different computational methodology: the standard formula and the internal model. The pdf is a mathematical function that assigns to an exhaustive set of mutually exclusive future events a probability of realisation. Its computation relies on non closed form calculus based on several Monte Carlo simulations and different hypothesis.

To fulfill the directives, companies should equip themselves with particular computational systems which, due to their complexity, require for their execution significant underlying IT infrastructures. This IT equipments represent a significant outlay for the company, not only for what concerns their direct costs due to the purchase, but also those for power consumption, cooling, maintenance and upgrades. Moreover, the technological progress that characterizes the IT field makes these computing resources become obsolete quickly. On the other hand, we face the risk that such resources become underused since these procedures could be used only periodically (at most monthly). If for the insurance company the utilization of *"on premises"* resources may not be advantageous, for a cloud services provider, increasing steadily its resources, leads to reduce the costs and consequently reduce the rates charged to the user.

The idea of this work is to put together the needs arising from Solvency II accomplishment with the always more advantageous scenario of cloud services. Within the project, as a case study, the focus is on a particular "module" of a more complex system named **DISAR®** [1], that deals with the calculation of the pdf and other significant items for an insurance company. This procedure relies on a Monte Carlo simulation algorithm developed on parallel MPI environment. The goal of the project is to

[1] Dynamic Investment Strategy with Accounting Rules ®, developed by Alef srl [1].

develop a framework that, by exploiting a learning algorithm is able to: calculate the computational power and the size of the IT infrastructure necessary for the execution of the procedure for a given input, deploy and make it available in a few minutes on the cloud, transfer to the grid the data necessary to the simulation, execute it and retrieve the output in a fully automated way transparent to the final user. The latter has the only task of choosing the *segregated funds* [2] the to work on. Moreover after each elaboration, the data relative to it are stored and subsequently utilized to improve the model for the next simulation. The cloud services provider adopted for this project is Amazon Web Services (AWS), however the core algorithm as the methodology can be applied independently of the provider.

[2] segregated fund is the translation for the Italian *gestione separata*. In life insurance a *gestione separata* is a fund specially established by the insurance and managed separately from the overall activities of the company. The segregated funds are used in contracts of class I; They are characterized by a composition of the investment typically prudent. The return obtained by separate management is used to reevaluate the coverage of the contract; generally it is also recognized as a return guarantee of the paid and/or a minimum return and the annual consolidation of results (which means that the returns made are definitely acquired and can not be changed by any loss or by lower yields of the following years).

Contents

Chapter 1

Introduction

1.1 State of the art

The IT services situation lately has quickly changed and it's evolving itself. Increasingly, we talk now of services in the cloud. Cloud computing is a general term for anything that involves delivering hosted services over the Internet. Despite this, the cloud environment is a so vast and heterogeneous field which is difficult to give a generic description of. First of all, to clarify some ideas, we can make a differentiation into three types of cloud services: **IaaS** (Infrastructure as a Service), **SaaS** (Software as a Service) and **PaaS** (Platform as a Service). In a IaaS Cloud, the resource provider focuses on delivering resizable computing capabilities in the form of VMs. SaaS is a software delivery method that provides access to software and its functions remotely as a Web-based service. PaaS provides a platform allowing customers to develop, run and manage Web applications without the complexity of building and maintaining the infrastructure typically associated with developing. For our purpose we are interested in IaaS and all that concerns with automated resource provisioning. IaaS platforms offer highly scalable resources that can be adjusted on-demand. This makes IaaS well-suited for workloads that are temporary, experimental or change unexpectedly. Usually users are charged by the provider only for the real time they use the resources, based on hourly or daily time scale. The pay-as-you-go model eliminates the capital expense of deploying in-house hardware and software. Leading IaaS providers include Amazon Web Services (AWS) [2], Windows Azure [7], Google Compute Engine [4] and Verizon [8]. Despite the presence of so many cloud providers, many large and medium-sized companies, for different reasons, but specifically for data protection policies, prefer private cloud or

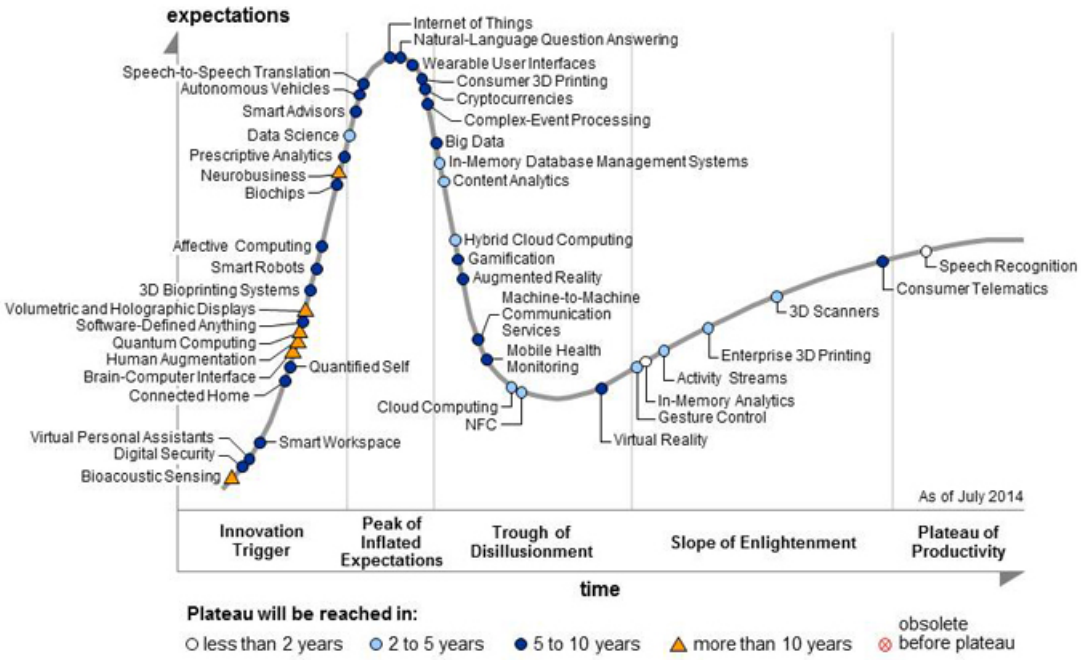


Figure 1.1. Gartner's hype cycle at July 2014.

ad hoc solutions, while many others choose hybrid ones (In figure 1.1 it is possible to observe how hybrid cloud computing is placed still at an highest position in the through of disillusionment with respect to cloud computing). Many companies in recent years work to provide software and operating systems for the management of private clouds. Many of these are based on open source projects managed by the community. In this context, very particular is IBM's position. Users may build their own private cloud or purchase services hosted on the IBM cloud. Users may also purchase IBM hardware, software and services to build their customized cloud environment. Private, public and hybrid clouds are not strictly distinct, as IBM allows the option to build a customized cloud solution out of a combination of public cloud and private cloud elements. For example a company can choose a private cloud solution, owned and operated by itself, or a private cloud solution, owned by itself but operated by IBM. Moreover a private cloud owned and operated by IBM is possible. Finally they offer virtual private cloud services and public cloud services. All these services are grouped under the name of IBM SmartCloud Enterprise [6]. The software module to manage the cloud, named IBM Cloud Manager, is based on the open project Open Stack [5], and can be considered an integration of Open-Stack [11] along with a multitude of value additions that would serve enterprise

customers. OpenStack is an open source cloud operating system. The project was started by RackSpace Cloud [12] and NASA. The idea was that of an operating system that controls large pools of compute, storage, and networking resources throughout a datacenter, all managed through a dashboard that gives administrators control while empowering their users to provision resources through a web interface. Nowdays the OpenStack Foundation counts more than 200 companies, among which Red Hat, SUSE Linux, VMware, Cisco, Dell, HP, IBM, AMD, EMC and Oracle. Another open source project, but of a slightly different nature is Nimbus [9]. It consists of some tools designed to launch, control and monitor cloud applications in single cloud or multi-cloud scenario. The project currently supports only OpenStack and Amazon EC2. More complex for offered services is Open Nebula [10]. It offers provisioning services for Amazon EC2 infrastructure, as well the possibility to create and manage private and hybrid cloud solution. It is able to orchestrate storage, network, virtualization, monitoring, and security technologies allowing to combine both data center resources and remote cloud resources. An other service able to manage Amazon EC2 instance is Ansible [3]. Written in Python too, it is an open source complex project to manage machines and nodes in general. Among its features it allows to create, terminate, start or stop an instance in EC2. Despite this large amount of tools that allow you to manage resources in EC2, for this work we chose the one that is closer to our goal, that is the deployment of cluster for parallel computing: StarCluster.

Starcluster We chose StarCluster [27] for the project, in order to simplify the automated boot process on Amazon Web Services. It is an open source software tool developed by the Massachusetts Institute of Technology (MIT) , released under the LGPL license. It has been designed to automate and simplify the process of building, configuring, and managing clusters of virtual machines on Amazon's Elastic Compute Cloud (EC2). StarCluster allows anyone to easily create a cluster computing environment in the cloud suited for distributed and parallel computing applications and systems. It is entirely developed in Python. In order to use it, a config file must be customized, more than one cluster description template could be defined in it. The file must be filled with all the necessary informations such as the AMIs (described later in section 4.3) to use for the master and for workers nodes, the machine typology, and the cluster dimension. Moreover the AWS login credentials must be provided to the file. Once the configuration is well defined,

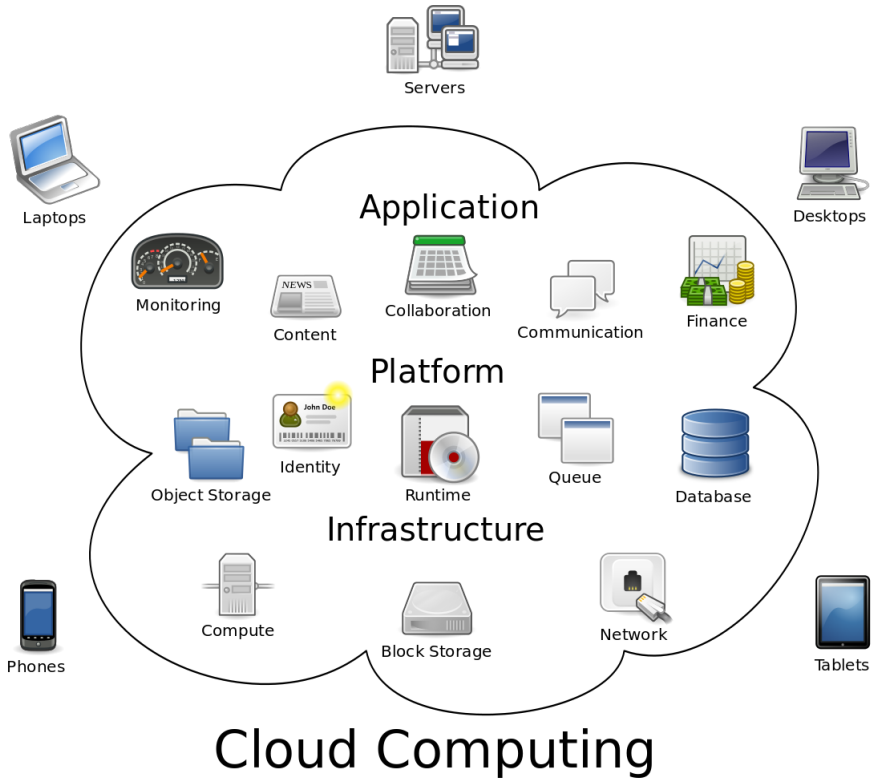


Figure 1.2. Cloud computing logical schema.

Starcluster offers some APIs to easily manage the AWS resources. We have seen

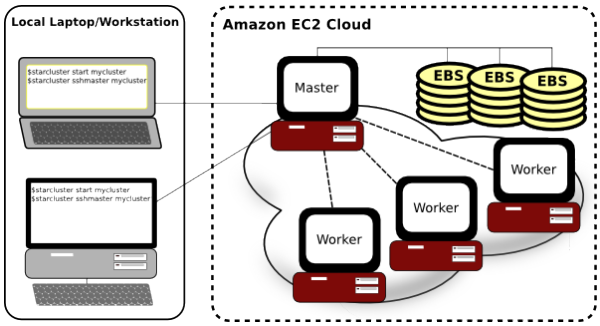


Figure 1.3. Starcluster working schema.

how many software has been designed to easily handle the services offered by the major resource provider in the cloud, and in particular AWS. However, there are many studies that attempt to address the issue of cloud resources provisioning in a theoretical and generic way, trying to develop models and frameworks that that are interposed between the user and the service provider. Some of them are

focused on providing a multicloud based service, what results interesting are the different criteria that are adopted in the different projects to choose among the different providers. For example in [28], the key criterion is security. What they want to do is to bind security in SLA, making it a measurable parameter. The core project is a framework accessible from a Web GUI. Through the GUI the costumer can negotiate different security levels. The user submits his security requirements for VMs, the system processes the request and splitting the request in technical constraints tries to solve them keeping as goal optimal costs. After the customer submits the proposed solution, the engine instructs OpenStack to launch the VM. A monitoring agent periodically keeps under control the SLA requirements and sends data to a monitor server. In an other work [31] the authors propose a SMS (Service Management System). The system collects and unifies the infrastructure providers informations. The project is aimed at acting in dynamic pricing multi-cloud environment. Moreover since it is thought to deal especially with Web services, it takes into account some performance indicators, such as requests/second. The number and the features of the VMs can dynamically change. The matching criteria to make the choice on how to develop the VMs is based on two target function: keeping always verified the performance constraints, minimize the cost of the resources yield by the first function over multiple cloud providers. Experimental results has showed how this approach leads to save from 6% to 8% of budget compare with AWS itself strategy.

1.2 Aim and motivations

As we have seen in section 1.1, there are many studies and projects trying to automate the resources provisioning on cloud, each one stressing different aspects, such as multi provider provisioning. We have shown how someone has focused on the particular problem of the provisioning for scientific applications [21]. Nevertheless each problem and application is different from the other, and it is difficult, or almost impossible, to think of a general solution that fits in all contexts without requiring significant efforts in machine-learning and AI fields. Even if some existing projects could be suitable for our purpose for what concerns the deployment, it is not the same in relation to resources estimation and sizing. In fact, as we will see in the following sections, our application problem is quite peculiar, although it is not difficult from this point of view. However regardless the financial context, since

the core algorithm is based on Monte Carlo simulations, the estimation algorithm, keeping into account some thin differences, should be easy adapted and reused in other works that rely on Monte Carlo algorithms. What would change would be only the image from which the VM would be deployed, leaving unchanged the technical framework. Moreover in developing our project we will base on Starcluster, a pre-existing tool we have introduced in 1.1. As mentioned in the abstract, we will work on a commercial Italian product named DISAR®. Its architecture and functioning will be deeply explained in section 3.1.3. To introduce the project aim it is sufficient to say that it is a modular insurance platform that is oriented to distributed computing. It it composed of some clients used to pilot the computation, a database, an orchestrator, and several nodes containing the engines. The orchestrator interacts with the database and distributes the computation on the engine nodes. There are two types of engines, the first typology computes what we will call the *flows*, the second one, starting from *flows* results, computes what we will call the *values*. In turn, the system provides a second branching, distributing layer, in fact a distributed version of the second kind of engine has been developed relying on MPI library. A procedure has been realized that, after the flows computation, reading from the database, prepares the input data for the values computation. Such procedure makes it possible to completely make the MPI engines autonomous, since the engines don't have to interact directly with the database and the orchestrator. At a computational level, the values computation is the most resources and time consuming step of the procedure (we will see why in section 2.2). Currently the insurance companies run Disar on *on premise* architectures, and based on the input complexity, the procedure can take from several hours until days to complete. The core idea of this project is the *outsorcing* of part of the system in a cloud environment. Obviously, the part interested to the migration is the one that involves the distributed MPI engines since it is the most resources consuming. As previously said, this is much more possible thanks to the fact that it can be completely made autonomous, allowing to keep besides the database *in place*. Finally our intention is to the reverse the resource/time relation. In fact if currently the architecture is given, and the time necessary is a result, what we want to reach is a scenario in which we decide the target time and the necessary resources are a direct result estimated and deployed by the framework. Moreover the resource will be deployed *on demand* permitting the core cloud formula *"pay as you go"* hold.

Chapter 2

The financial context

Before describing the details of the project, it's necessary to introduce some preliminary financial notions useful for the reader for better understanding the context in which the it arises. Moreover, the elements we are going to introduce here make the reader's mind build an idea of the computational complexity of the calculation problem we face when we approach the pdf elicitation.

2.1 The Solvency II directive

In this section the Solvency II directive is introduced. The Directive in its entirety is quite complex and intricate. However, for the purposes of this work we just give a brief overview. Also we will address only the relevant aspects in order to have a preliminary background needed to understand what we will discuss later in the work. In 2.1.1 we give the directive overview, in 2.1.3 we explain the difference between the *standard formula* and the *internal model* in the Solvency II framework.

2.1.1 The directive

The main purpose of the Solvency II directive is the protection of the policyholders and the stability of financial market. The key instrument of the regulation is the measure and control of the risk faced by the insurance and reinsurance firms. The regulation process involves the national supervision agencies with an European dimension, in fact, the insurance companies are supervised by their national regulation organisms like the Italian **Istituto per la Vigilanza sulle Assicurazioni (IVASS).** However, in accordance with its European nature, a supra-national authority, the **Eu-**

ropean and Occupational Pensions Authority (EIOPA) ensures the convergence of the regulation standards. As stated by De Felice and Moriconi [20], the directive has seven keywords: the *risk*, the *governance system*, the *market-consistency*, the *model*, the *data quality*, the *advanced technology* and the *knowledge*. However the main aim of the directive is the control of risk, the definition of risk is not stated in it. A definition of risk can be find in *A Global Framework for Insurer Solvency Assessment* [26] that define the risk as *the chance of something happening that will have an impact upon objectives*. After we have established what the risk is, it is important to identify how to *measure* it. The directive stated the *risk measure* as *a mathematical function which assigns a monetary amount to a given probability distribution forecast and increases monotonically with the level of risk exposure underlying the probability distribution forecast* (Article 13, paragraph 39). Article 45, paragraph 1 says that *every insurance undertaking shall conduct its own risk and solvency assessment.* Moreover *Member States shall require all insurance and reinsurance undertakings to have in place effective system of governance which provides for sound and prudent management of the business* (Article 41). Summarizing we have that insurance undertakings have to use a governance system to evaluate and manage the risk, the methodology applied must be compliant with some well defined data quality constraints.

2.1.2 Solvency Capital Requirement (SCR)

In the risk valuation a central concept is represented by the Solvency Capital Requirement, the SCR computation requires two different phase: the measure of the risk through the **Value at risk** and secondly the **aggregation of risks.** Solvency at high level can be defined as the ability of an insurance undertaking to discharge its indebtedness. The directive does not explicitly define the SCR, the Article 101 states: *The SCR shall be calibrated so as to ensure that all quantifiable risks to which an insurance undertaking is exposed are taken into account. It shall cover existing business, as well as the new business expected to be written over the following 12 months. With respect to existing business, it shall cover only unexpected loss. It shall correspond to the Value at Risk of the basic own funds of an insurance and reinsurance undertaking subject to a confidence level of* 99.5% *over a one-year period*. In a easier way, we can say that it determines the amount of capital that ensures that an undertaking will be able to meets its obligations over one year with a probability of at least 99.5%, which limits the chance of falling into financial ruin to

less than once in 200 cases. [19] Let us denote with X the assets of the insurer and with Y its liabilities. We introduce a new quantity that we will address as **NAV (Net Asset Value)**. If we define with $V(t;\mathbf{X})$ and $V(t;\mathbf{Y})$ respectively the *market consist values* of assets and liabilities at time t, the NAV can be expressed with the formula:

$$NAV(t) = V(t;\mathbf{X}) - V(t;\mathbf{Y}), \tag{2.1}$$

where

$$V(t;\mathbf{X}) = \sum_{k=1}^{n_x} V_k^x(t), \qquad V(t;\mathbf{Y}) = \sum_{k=1}^{n_y} V_k^y(t), \tag{2.2}$$

with n_x (n_y) the number of assets (liabilities) and with $V_k^x(t)$ ($V_k^y(t)$) the value of the $k-$th asset contract (liability contract). The *SCR* estimation involves the evaluation of the expected value of *NAV* (denoted with $E[NAV(T)]$ and of a percentile, at a prefixed confidence level $\alpha(=0.005)$, of the *NAV* distribution at a future time $T>0$ ($NAV_\alpha(T)$ with $T=1$ year).

Using eq. 2.1 the SCR in $t=0$ over an horizon $T>0$ is:

$$SCR_\alpha(0,T) = [E[NAV(t)] - NAV_\alpha(T)]v(0,T), \tag{2.3}$$

where $v(0,T)$ is the risk-free appropriate discount factor prevailing on the financial market at time zero for the maturity T.
In figure 2.1 is represented the graphical illustration of the “fundamental” measures of Solvency II.

In the representation, assets and liabilities are put beside for emphasizing the need to adopt an unified set of valuation criteria able to measure all elements (‘"assets" *vs* "technical provisions" + *SolvencyCapitalRequirement*) in the logic of "total balance sheet", implementing a consistent plan of "asset-liability management" (ALM)[1].

[1]The asset-liability management, in the Professional Actuarial Specialty Guide [30], is defined as "the practice of managing a business so that decisions on assets and liabilities are coordinated; it can be defined as the ongoing process of formulating, implementing, monitoring and revising strategies related to assets and liabilities in an attempt to achieve financial objectives for a given set of risk tolerances and constraints".

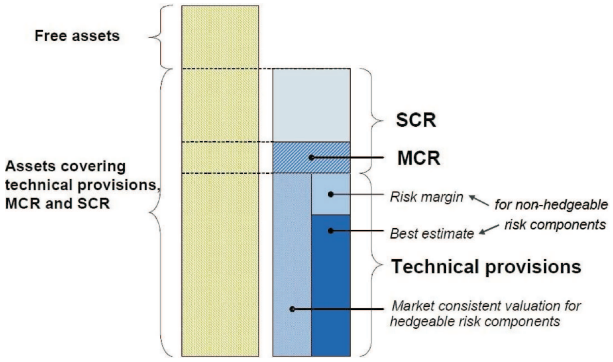


Figure 2.1. Fundamental measures in Solvency II

2.1.3 Standard formula and internal model

Article 100 of the directive defines two classes for the SCR computation: *The Solvency Capital Requirement shall be calculated , either in accordance with the standard formula [...] or using an internal model.* The standard formula is a method defined by the EIOPA, in the internal model, the undertaking, upon approval of the supervision authorities, is allowed to substitute some or all modules of the standard formula. The standard formula is specified by the European Commissions. According to the Article 103, *the Solvency Capital Requirement calculated on the basis of the standard formula shall be the sum of the following items: (a) the Basis Solvency Capital Requirement , as laid down in Article 104; (b) the capital requirements for operational risk, as laid down in Article 107, (c) the adjustment for the loss absorbing capacity of technical provisions and deferred taxes, as laid down in article 108.* The module (a) can be in turn divided into the following sub modules: (a) non-life underwriting risk, (b) life underwriting risk, (c) health underwriting risk, (d) market risk, (e) counter party default risk. In alternatives, the insurance companies can use a full or partial internal model. Article 112 states that *insurance and reinsurance undertakings may use partial internal model for the calculation of one or more of the following: (a) one or more risk modules, or sub-modules, of the Basic Solvency Capital Requirement , as set out in Article 104 and 105; (b) the capital requirement for operational risk as set out in Article 107; (c)the adjustment referred to in Article 108. In addition, partial modeling may be applied to the whole business of insurance and reinsurance undertakings, or only to one or more major business units.* However, the model has to be approved by the supervision authorities and must be compliant with several requisites.

2.1.4 Probability Distribution Forecast (PDF)

As previously stated in the abstract, the *PDF* is defined in the art. 13 of the Directive Solvency II: "probability distribution forecast means a mathematical function that assigns to an exhaustive set of mutually exclusive future events a probability of realization"; it is considered a fundamental component of the "internal model". The art. 121 says that the calculation of the probability distribution forecast has to "be consistent with the methods used to calculate technical provisions" and the art. 122 says that "where practicable" the *SCR* should be evaluated "directly from the probability distribution forecast generated by the internal model" using the Value-at-Risk approach at a confidence level of 99.5%, over a one-year period. An evaluation principle must be introduced since, for estimating TP (Technical Provisioning)), NAV, SCR, the valuation of both assets and liabilities is required. The valuation is performed assuming that all the random variables concerning the valuation problem are defined on a probability space$(\Omega, \mathscr{F}, \mathbb{P})$, where Ω denotes the space of events, $\mathscr{F}$ the σ-algebra of the measurable events and $\mathbb{P}$ is the physical probability measure (also known as the real-world measure). The most relevant technical issues in the evaluation can be illustrated – without loss of generality – taking into account only the "risk drivers" of financial risk. The risk drivers (interest rate, inflation, stock price, exchange, credit, contract specific risk sources) are modeled by $Z(t)$, a multivariate stochastic process, eventually a Markov one. The value $V(t)$ of a generic contract (asset or liability) in $0 \leq t \leq H$ with term H, under the necessary assumptions, is given by

$$V(t) = E^{\mathbb{P}}\left[\xi(H)V(H)\middle|\mathscr{F}_t\right], \tag{2.4}$$

where $\xi(t)$ is a suitable "state-price deflator". In this market, we assume the existence of a suitable equivalent martingale measure $\mathbb{M}$ (*risk-adjusted*, *forward* o others) under which

$$V(t) = N(t)\,E^{\mathbb{M}}\left[\frac{V(H)}{N(H)}\middle|\mathscr{F}_t\right], \tag{2.5}$$

where $N(t)$ is the corresponding numéraire such that the process $Y(t) = V(t)/N(t)$ is a $\mathbb{M}$-martingale; $\mathscr{F}_t$ is the filtration at time t. If the time $t = 0$ is the current time, $V(0)$ is known with certainty, while $V(T)$, $0 < T \leq H$ is a random variable depending on the $Z(t)$ trajectory in $[0,T]$. In many cases of practical interest, the

closed-form solution (or accurate approximations at least) of assets value is available, while liabilities value, due to the complexity of payoff, cannot be computed analytically; thus a viable approach is to rely on numerical simulations. The numerical approach for evaluating liabilities, shown below, can also be applied to assets when no closed-form solution on the asset side is accessible. $V(t)$ is the expected value of a multivariate distribution that, as is the case of insurance contracts, is defined on a very complex domain and with a cumulative distribution function not available in closed-form. This leads to having to necessarily use Monte Carlo simulation to computing the integral in (2.5), possibly in combination with other techniques for the management of complex payoff. The *SCR* estimation requires the evaluation of the expected value $E[NAV(T)]$ and the percentile $NAV_\alpha(T)$. In general, the *NAV* distribution is not available even in cases in which the joint distribution of risk drivers is known and a closed-form valuation of *NAV* is available. So also in this case it is necessary to use simulation methods to numerically calculate approximate values of *SCR*. Monte Carlo simulation is the most suitable one to elicit an empirical probability distribution of *NAV*. In summary, considering the typical composition of insurance company portfolio, in the evaluation process of *NAV*, the liability value $V(t;\mathbf{Y})$ has to be evaluated using Monte Carlo simulation (more detailed in section 2.2.1), either for $t = 0$ or for $t = T$, while in general the asset value $V(t;\mathbf{X})$ can be calculated in closed-form otherwise using in the same way Monte Carlo simulation.

2.2 The nested Monte Carlo simulation and the "big computational problem"

In this section we will explain which are the issues that led us to develop this project in a cloud scenario. In section 2.1.2 we have show how the SCR computation requires to evaluate all the risks concerned with the business of the company. This involves the valuation of life insurance policies. In 2.2.1 we show how to valuate a *life insurance policy contract*. In 2.2.2 we introduce the key points of a nested Monte Carlo simulation, we will give an idea of its working algorithm highlighting which are the aspects that can make its application very computationally demanding. finally we will address the aspects of policy valuation that make Monte Carlo complexity increase to a point such that, it is non still convenient for a company to run it on an on premise infrastructure, rather a cloud solution become more suitable.

2.2.1 Valuation of a life insurance policy

The purpose is to introduce the life insurance policy, in particular the Italian *gestione separata*, and point out how it depends on the performances of a segregated fund. To better understand the scenario we give a formal definition of a life insurance policy:

Definition 1. ***A life insurance policy is a contract promising a financial compensation, i.e. the benefits, against some random events which may affect the life of its subscriber. The cost of this protection has the form of a premium paid on a periodic basis and/or a capital transferred to the insurance company.***

Let us define by the event $E(T)$ "the insured event at time T" and by $\mathbb{1}_{E(T)}$ its indicator function. Typical cases of the event $E(T)$ are life, death and lapse that correspond to the fact that the insured person at time T is alive or dead, or he request the early lapse of the contract. These possible events, that are not certain and need to be evaluated, are denoted as *technical risks*. The contract implies cash-flows exchanges between the insurance firm and the policyholder in case these events occur. Benefits and premia are linked to the yield of a segregated fund. Let us consider a contract written at time 0 with term T years and initial sum insured C_0. The benefit C_t that should be paid by the insurer to the policyholder at time T is determined by incrementing each year the sum insured by a fraction β of the interest earned by the insurer on the investment of the premium. The technical reserve is invested in a reference fund, directly managed by the insurance company. The performance of this fund depends on those who are denoted as *"financial risks"*. Thus the service eventually received by the subscriber can be defined as:

$$Y_T = C_0 \Phi_T \mathbb{1}_{E(T)}. \tag{2.6}$$

Φ_T in eq. 2.6 represents the readjustment factor in turn defined as:

$$\Phi_T = \prod_{t=1}^{T}(1+\rho_t) = (1+i)^{-t}\prod_{t=1}^{T}(1+max\{\beta I_t, i\}). \tag{2.7}$$

where i is a basic technical rate. If the market value of this fund at time t (when the premium is invested) is F_t, the rate of return earned by the fund in year $[t-1,t]$ is:

$$I_t = \frac{F_t}{F_{t-1}} - 1; \tag{2.8}$$

The Italian profit sharing system called *"con minimo garantito"* provide that if $I_t > i$, part of the extra earned interest is credited to the insured by increasing the insured sum by a readjusting factor at the end of the year following the rule:

$$C_t = C_{t-1}(1+\rho_t), t = 1,2,...,T, \tag{2.9}$$

where ρ_t is the readjustment rate defined as:

$$\rho_t = \frac{max\{\beta I_t, i\} - i}{1+i}; \tag{2.10}$$

where $\beta \in (0,1)$ is the *"participation coefficient"* that together with i is contractually specified at time 0. The quantity βI_t is the portion of the fund credited to the policyholder. The threshold rate i is a lower bound that guarantees that the sum insured cannot decrease even if the rate i is not realized by the fund in that year. Equation 2.6 is crucial to understand on which aspects liability depends. It is affected by "financial" and "technical" (actuarial) uncertainty. What we are interested in is to assign a value at time t to the random variable Y_T. Since C_0 is known at time 0, what we have to evaluate are both risk categories. If for the first category we are able to evaluate them with a closed form, the variables for the second category are several and not a priori known, thus since we have not a closed formula to evaluate them, Monte Carlo simulation is the most suitable tool to address the problem.

2.2.2 Nested Monte Carlo simulations

The nested Monte Carlo simulation is at the current time, for insurance undertakings, the most suitable approach to measure the Solvency Capital Requirement with the Value-at-Risk approach as required by the Directive Solvency II, since it allows to elicit an empirical probability distribution function of contracts values in future time, and then the corresponding moments and percentiles. It is used to understand the impact of risk and uncertainty in forecasting models. In developing forecasting models we make some assumption about the investment return. In a Monte Carlo simulation the model is calculated based on some random variables. The result of the model is recorded, and the process repeated hundreds or thousands of times, each time using different values. Once a simulation is terminated, we have a large number of results, each based on random input values. These results are used to describe the probability of reaching various results in the model. The nested Monte

Carlo simulation is actually the most suitable tool for the **SCR** computation with the Value-at-Risk approach. [16] A nested Monte Carlo simulation is based on two operations:

1. the simulation of n_P sample paths $(Z(t)^{(i)}$, $i = 1,...,n_P$ from $t = 0$ to $t = T$ under the *real world* measure P, conditional to F_0.

2. for each of these n_P sample paths $(Z(t)^{(i)})$ from $t = 0$ to $t = T$, the simulation of n_M sample paths $(Z(t)^{(i,j)})$, $j = 1,...,n_M$, from $t = T$ to $t = H$ under the equivalent martingale measure M (for example the risk-neutral probability Q), conditional to F_T.

The n_P simulations are identified as the *outer* simulations while the n_M are are the *inner* ones. Monte Carlo simulation results in a nested stochastic simulation with a large number of inner simulations for each outer scenario for the risk drivers valuation. In insurance field, the total simulations number may be impressively if we want to obtain reliable estimates. Moreover, the computational effort required to evaluate liability cash flows in each single outer scenario could be very high. This, along with the number of simulations, is to impact dramatically on resources and time required to perform the Monte Carlo. What is particularly interesting, we will address this deeper in section 4.1, is the fact that Monte Carlo is very suitable for parallelization, in fact due to its schema, it provides a good scaling factor. The development of a system able to deal with the requirements of the directive requires a strong synergy between high-level theory and high-level technology, that is a synergy between models and techniques of quantitative finance, computational schemes and data management. The appropriateness of data quality and models as well as accuracy and efficiency of computation and the adequacy of the IT infrastructure are more and more preconditions for an efficient governance of insurance companies. In section 3.1.3 we will introduce and describe Disar, a simulation system designed for computing *Technical Provisions* and *Solvency Capital Requirement* in compliance to the *Own Risk and Solvency Assessment*, as requested by the Directive to the "internal model". The simulation process in Disar is based on stochastic models for values and risks evaluation, to achieve this, Monte Carlo simulation is used.

Chapter 3

Overview of the Insurance Data System

3.1 IDS®

The Insurance Data System (IDS®) is an IT platform developed by Alef, an Italian company that produces solutions to the financial problems of public and private firms, banks and insurance companies. The system aim is to manage: data bases, data quality processes, computation's engines and functions for analysis results. Its design is based on a modular an pluggable logic that allows to easily develop new components in case new needs arise. Before the elaboration, the centralized system receives data from all the companies belonging to the group through well defined protocols that ensure some specific data quality constraints.

3.1.1 IDS®architecture

The system's architecture is composed of five logical components:

1. user interface
2. calculation's orchestrator
3. engines for data management
4. computation engines
5. database

The user interface is the component that provides management of procedures and processes, the orchestrator disciplines and manages the execution of distributed and parallel computing engines, engines for data management protect the data quality and the organization of the results, computation engines realize the algorithmic processes, the database contains all system's data. All the necessary data, included data produced by the system, are inside the database that is a Relational database Management System (RDBMS) Oracle. The user interfaces are Windows and Web applications, the engines for data management are written in PL-SQL, ANSI C, C++, the computational engines are realized in ANSI C, C++, FORTRAN, while the orchestrator is written in C++.

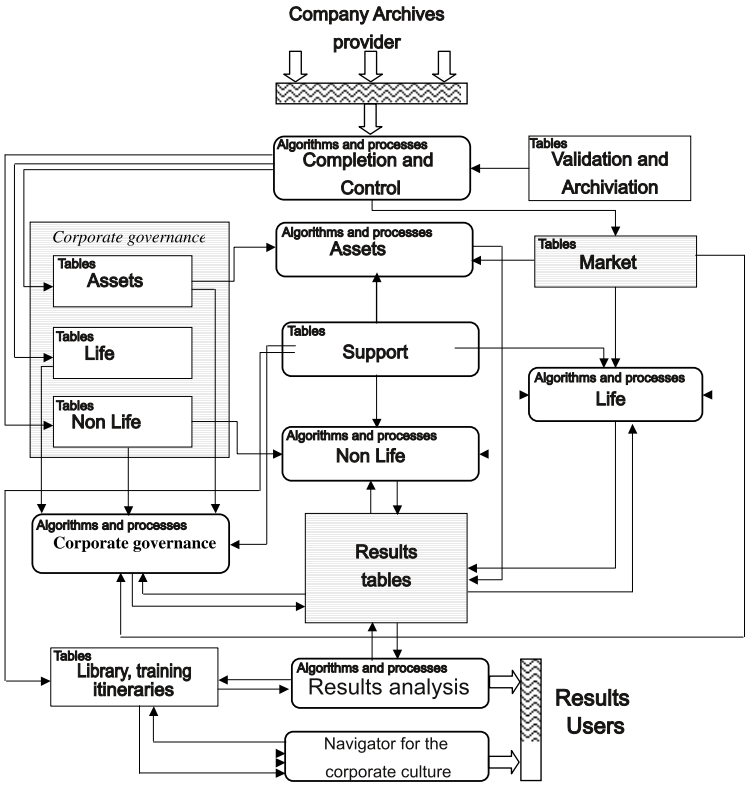


Figure 3.1. Architecture of the system Insurance Data System .

For the production environment the most recent version of each procedure resides on a reference repository for the system. The procedures that reside on the client are automatically updated at execution time. The database is made of approximately 2000 "objects" that consist in tables, views and stored procedures. As previously mentioned, the system's modularity allows to divide it in four well distinct subsystems:*assets*, *life*, *non-life* and *corporate governance*. Each procedure

is composed of objects. An object may be common to more procedures. In the database for each procedure there is an association with all of the component objects. Each object is characterized by a version number. The database through mechanisms of authentication and authorization, allows the certification of data and results. It can be divided into 8 main components:

- the component **Validation and storage** contains tables for decoding and standardize input data. It realizes the preliminary checks and the storage of data in the database.
- the section **Support** contains tables of related informations, regulations, conventions and rules for the calculations.
- the **Market** section is designed to contain quantitative information on the markets situation, data are distinguished by date of detection, by type, and source.
- the **Assets** archive is a database that contains demographic and business information on standards and structured financial contracts. Demographic data include information on the contract nature while, the corporate archives contain information on contracts-be. Corporate data are distinct by consistency date, portfolio and undertaking.
- the **Life** archive contains life and business data related to life insurance contracts, that can be of 3 typologies: traditional, unit-linked or index-linked. Personal data include information on the characteristics of the contracts that derive from the technical note. The corporate archives contain information on existing contracts. Corporate data are distinct by consistency date, portfolio and undertaking.
- the **Non-Life** (in Italian commonly defined "Danni") archive contains informations on the damage payments, in the traditional form of triangles. Moreover it contains data on reserves, data on premiums, expenses, assumptions about the persistence and the new activities and reinsurers ratings. Stored informations, and in particular those about the triangles, are distinct by enterprise, branch, detection date and data type. The information necessary to assess the catastrophic risks (sums insured, estimated earned premiums, ...) are stored together with informations, that about the reinsurance contracts, that are used for the evaluation of specific risks.

- the **Tables of results** contain data produced by the calculation engines. Data are distinguished by calculation procedure and by processing key. These data are used for defining and editing staff reports.

- section **Library and training itineraries** contains technical documentation of the calculation procedures and their user manuals.

3.1.2 Subsystems and procedures

IDS is composed of 37 procedures, organized into 7 subsystems. Twelve of this are for the management and protection of data quality and they are the following:

1. **CD-Merc-I**, procedure for the identification of market data required by elaboration.

2. **CD-Merc**, procedure for market data storage.

3. **CD-Titoli-C**, procedure for personal details of the assets portfolios completion.

4. **CD-TitoliAn**, procedure for the acquisition of personal data relative to assets portfolios.

5. **CD-TitoliAz**, procedure for the acquisition of enterprise data relative to assets portfolios.

6. **CD-IndexAn**, procedure for the acquisition of personal data relative to structured contracts.

7. **CD-IndexAz**, procedure for the acquisition of enterprise data relative to structured contracts.

8. **CD-VitaAn**, procedure for the acquisition of personal data relative to life contracts.

9. **CD-VitaAz**, procedure for the acquisition of enterprise data relative to life contracts.

10. **CD-StatVi**, procedure for the acquisition of demographical statistical bases for the life business.

11. **CD-NonLifeAz-gf**, procedure for fine grained data acquisition for the non-life business.

12. **CD-NonLifeAz-ag**, procedure for aggregated data acquisition for the non-life business.

Six procedures are instead dedicated to econometric and statistical estimates:

1. **E-CIR**, procedure for the estimates of parameters relative to the model for interest rate expiration.

2. **E-VolCor**, procedure for volatility and correlations estimates on historical series.

3. **E-Mor**, procedure for mortality tables calibration.

4. **E-Lapse**, procedure for lapse tables calibration.

5. **E-USPReserve**, procedure to calculate the undertaking specific parameters for the reserve risk.

6. **E-USPPremium**, procedure to calculate the undertaking specific parameters for the premium risk.

Three procedures belong to the Assets subsystem:

1. **T-VRStandard**, procedure for the portfolios evaluation containing standards financial contracts.

2. **T-VRIndex**, procedure for the evaluation of structured contracts.

3. **T-CredRisk**, procedure for the control of credit risk.

Non Life subsystem is composed of seven procedures:

1. **D-Reserve**, procedure for the calculation of the characteristical quantities of the claims provision (riserva sinistri).

2. **D-Premium-TS**, procedure for the SCR calculation for premium. (on historical series)

3. **D-Premium-FS**, procedure for the SCR calculation for premium. (with Frequency-Severity model)

4. **D-CatRisk**, procedure for the SCR calculation for catastrophe risk.
5. **D-LapseRisk**, procedure for the SCR calculation for lapse.
6. **D-RCBusiness**, procedure for the SCR calculation for business.
7. **D-RCTasso**, procedure for the SCR calculation for in-force business rate and renewals.

The Corporate Governance subsystem is composed of two procedures:

1. **G-StanFor**, procedure for the standard formula calculation.
2. **G-Consol**, procedure for the consolidation of the characteristical quantities of Solvency II.

The Life subsystem, that is the one we are particularly interested in, counts the following six procedures:

1. **V-Disar**, it is a procedure for the evaluation and risk management of segregated funds (see section 3.1.3).
2. **V-Index**, it is a procedure for the evaluation and risk management of indexed-linked contracts.
3. **V-Unit**, it is a procedure for the Fevaluation and risk management of unit-linked contracts.
4. **V-NonInd**, it is a procedure for the evaluation and risk management of non-indexed contracts.
5. **V-Hypothesis**, it is a procedure for the management of segregated funds.
6. **V-ProfitTest**, it is a procedure for planning and do economic evaluation of life insurance policies.

3.1.3 Disar®: a procedure of IDS®life subsystem

For the purpose of this thesis we treat deeply **Disar®** that, as we have previously seen, is a procedure of the IDS®life subsystem. The Disar (Dynamic Investment Strategy with Accounting Rules) procedure is aimed at the evaluation and control

of revaluable life policies hooked to *"segregated funds"*. It is based on *"market-consistent"* evaluation criteria under uncertainty and on *"asset-liability" management schema*. It takes into account accounting rules that control the segregated fund budget and the management strategy for the financial contracts *portfolio*. The adopted stochastic model considers more sources of uncertainty such as the interest rate risk, equity risk, currency risk, credit risk, share price, exchange rate, inflation, default of counterparts, furthermore actuarial risks such as longevity/mortality and lapse are treated. Actuarial risks are assumed to be independent between each other, while financial risks are possibly correlated. All this is done in compliance with the Solvency II regulations.

The Disar procedure consists of two main phases:

- phase A – Actuarial valuation, that is the calculation of actuarially expected cash-flows generated by the contracts.
- phase B – Alm valuation, that is the evaluation of market consistent values of contracts.

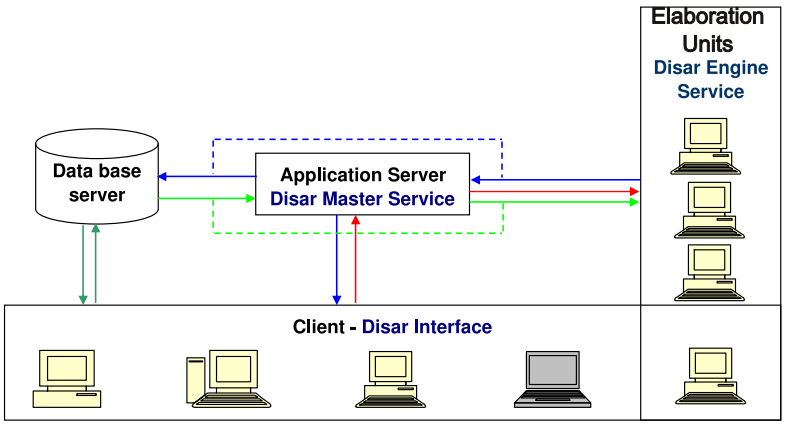


Figure 3.2. Disar® Architecture.

Disar® architecture

Disar® has a client/server working paradigm that allows simultaneous utilization from different concurrent workstations. The distributed computing structure is of *grid type*, the nodes in the grid contain the engines that are utilized through an *"at service"* logic. In figure 3.2 the following main logical components are represented:

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 Alm 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 procedure

The whole procedure is managed through the Disar Interface that access the database and allows the user to have a full control on the computation. As we can see in figure 3.3 the user can choose a set of company's portfolios on which he wants to perform the elaboration. Other masks allow to set the elaboration hypothesis and many others input parameters.

At this point it is useful to introduce a definition that we will encounter several times during this work:

Definition 2. *An **elementary elaboration (eeb)** is a set of a segregated fund and computational and market hypothesis.*

A mask of the interface is fully dedicated to the grid configuration, it displays a list of DiMaS servers and elaboration units, a working configuration must contain a DiMas and a pool of elaboration units.
When the computations is started, what the DiMaS do is to divide all the input data in **elementary elaborations** of the above type. The DiMaS acts as orchestrator, it defines 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 that is present on each node, delivers the elaboration to the DiActEng or to the DiAlmEng depending on the elaboration type:

- The DiActEng is in charge of phase A, that is performs an EEB of type A, it operates on the policy portfolio related to the segregate fund, it receives as input the contractual informations, the consistency of policies and the

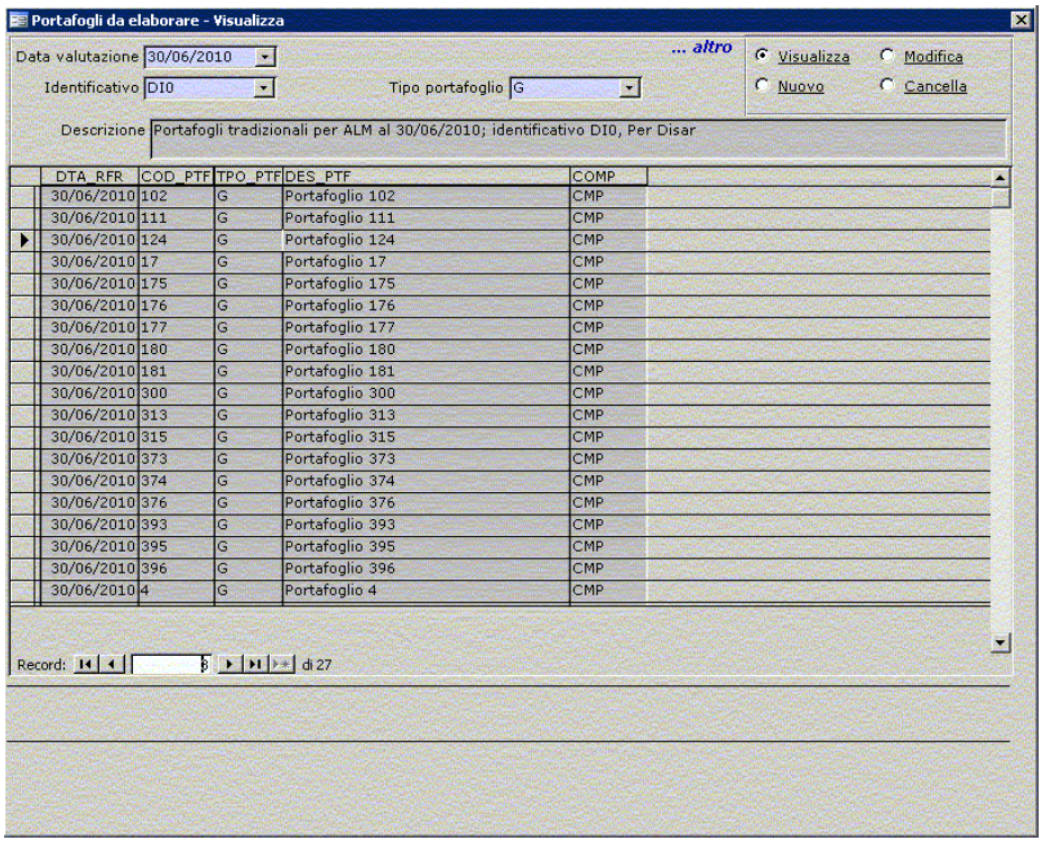


Figure 3.3. Disar®Interface, mask for the selection of portfolios to evaluate.

technical informations and it calculates on the related schedule the aggregate *"probabilized flows"* related to net performances, without loss of information.

- The DialmEng is in charge of phase B, that is performs an EEB of type B. Disar Asset-liability management Engine is the financial engine that operates on the policy portfolio related to the segregated fund, it receives as input the contractual informations, the accounting informations, the probabilized flows computed by the DiActEng related to net performances, the financial hypothesis on the market structure, the features of the management strategy and produces the characteristical quantities useful to evaluate and to manage the risk such the evaluation of market consistent values of policies.

When the engines terminate the elaboration, the DiEng may write the results directly to the Database or return them to the Master Server.

The engine algorithm is realized through the use of Monte Carlo simulation (details in Section: 3.1.3) that performs a large number of trail runs, called simulations, and infers a solution from the collective results of the trial runs. It generates

thousands of probable investment performance outcomes, called scenarios, that might occur in the future. The simulation incorporates economic data such as a range of potential interest rates, inflation rates, tax rates, and so on. The data is combined in random order to account for the uncertainty and performance variation that's always present in financial markets. Disar adopts *risk neutral* and *natural* probability distributions, whose result is the computation of certain quantities that are relevant for *budgeting* and *management*.
The simulations in "risk-neutral" environment, allow to calculate the market-consistent value of some characteristical quantities related to the *policy portfolio*, for example stochastic reserve, value of embedded options in the guaranteed minimum, value of business in force (VBIF). Moreover *"forward - risk-neutral"* expectations of some fund characteristical quantities are computed (budgetary reserve, future profits, return for re-evaluation).
In "natural environment" the simulation produces instead the distributions for the characteristical quantities related to future dates, computing their expectation.
In "mixed environment", using both natural and risk-neutral probabilities, the financial capital risks (interest rate, equity, default) are instead calculated.
In each relevant date Disar, in particular at the end of each year, makes available details about typical sizes of management : equities value, solvency, net income, average balance, management performance, write-downs or write-back value, reserves, profit for the year.
The computation of the market consistent valuation and risk assessment is performed using Monte Carlo simulations for each segregated fund and then aggregating the results. The computations are performed separately for each segregated fund, however if there is only a segregated fund, then evaluations correspond to elementary elaborations. The DiMaS commands the computation process, as we have shown, it receives the request from the clients and organizes the computation grid. Once the elementary elaborations are processed, the server collects their output and writes them on the database.

DiAlmEngParNs: A parallel version of the DiAlmEng

Disar is a high complexity simulation system, its engines are data intensive and CPU intensive. The most time consuming jobs are those relative to the phase B, and so the DiAlmEng that evaluate TP, NAV and SCR. Since the engine relies on Monte Carlo, a further improvement of the system is achievable by parallelizing the

simulations.
The parallelization strategy is based on the distribution of simulations among processors. Consequently a parallel version of the financial engine has been developed that is **DiAlmEngParNs (Disar Asset-liability management Engine Parallel Nested simulations)**.
In this implementation processors operate concurrently in the calculation of the average "local" values, which then are suitably combined to produce the final "global" results. The engine is written in **Fortran**, the parallelization is achieved through the use of the **MPI Library**.

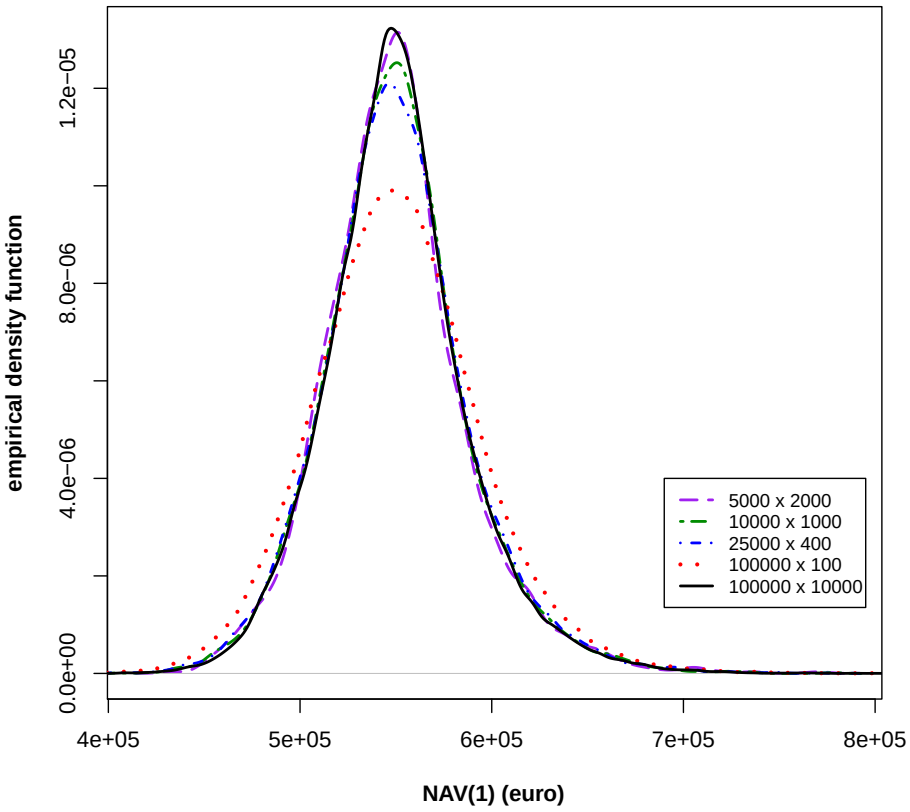
Previously, in reference to multicore architectures, different solutions have been explored. [15] One of these involved **shared memory parallel programming** based on **non-uniform memory access (NUMA)** and **Uniform Memory Access (UMA)** technology. Others implementations involved hybrid paradigms (message passing and shared memory parallel programming), the tested message passing libraries are IntelMPI and OpenMPI.

Let M be an appropriate probability "equivalent martingale" measure[1], denoted with n_M the number of simulations of the trajectories of probability M, and with I the number of MPI ranks, each rank performs $[n_M/I]$ simulations independently and calculates the average values on the basis of local simulations. Subsequently it competes with the others to calculate the average global values. Therefore, inter-process communication is confined to the phase of the final values production. Processing of the remaining $r = n_M \ mod \ I$ simulations is distributed among the ranks numbered from 1 to r, that perform hence $[n_M/I]+1$ simulations.

The parallelized simulations are those related to the trajectories with *"natural"* probability measure P, these simulations are called *"external simulations"* and are distributed with the strategy described above. Denoted with n_P the number of natural simulation, each process performs autonomously $[n_P/I]+1$ simulations if it belongs to processes in range $[1,r]$ with $r = n_P \ mod \ I$, conversely $[n_P/I]$ if it is in range

[1] Also known with the name of risk-neutral measure, it stems from the fact that, under it, all the financial assets of the economy have the same expected return (called risk-free), regardless of their risk level. This happens in contrast to the so-called physical measurement, i.e. the "true" probability distribution yield, according to which in general characterized by an increased risk securities have a yield on average higher (that is, are characterized by a risk premium positive).

$r+1, I$.



[]

Figure 3.4. Empirical density function of $NAV(T)$ with $T = 1$ year for different value of n_P *x* n_M

As previously stated, for each portfolio the computation is divided in several elementary elaboration. Each elementary elaboration required a certain amount of time to complete, depending on two main factors: the complexity of the belonging portfolio, and the quantities it is going to compute, e.g. market all risks, underwriting all risks, underwriting mortality risks, underwriting lapse risks etc. For some of them, for example the evaluation of the empirical probability distribution of $NAV(T)$ with $T = 1$ year, which is needed to calculate the $NAV_\alpha(T)$ and the $SCR(0,T)$, is performed a nested Monte Carlo simulation: for each natural simulation, the process of rank i, runs inside its $[n_P/I]$ simulations, even the elaboration of the n_M *"internal simulations"*.

The consistency of the results of parallel and sequential algorithms is guaranteed by the application of appropriate implementation techniques of procedures for generating pseudo-random numbers. The generation of independent streams is realized by the technique of block-splitting (also known as skipping ahead [24]) which divides the original sequence into k non-overlapping blocks, where k is the number of independent streams. Each flow generates pseudo-random numbers from the corresponding block. Given I processes, wishing to obtain the same results from the parallel version of the sequential algorithm, fixed the same initial seed, if N is the number of random numbers generated, a block of pseudo-random numbers of size $[N/I]+1$ is assigned to the processes from 1 to r with $r = N\ MOD\ I$, and a block of size $[N/I]$ is assigned to the remaining processes. The high performance computing systems identified for the development of the engine are multicore systems. For the engine version used in this work, the **Intel Cluster Studio XE 2013** environment has been used and for the concurrent processes management the **Intel MPI Library**, the library for the message-passing management, compliant with the MPICH ABI Compatibility Initiative, which guarantees both hardware and software product portability. The chosen math library is the **Intel MKL Library**. Applications allow the use of the generators MCG 31, MT2203 and MT 19937 that are based on **Mersenne Twister** algorithm for *Large Scale Monte Carlo simulation* on distributed computing systems. The choice of the generator is determined by a value assigned to an input parameter.

A particular procedure, named **DiIOAlmEng**, has been developed to this engine version aid. It access the database, reads data in it, and taking in account result of phase A previously stored, generates all the input files necessary to the DialmEng-ParNs engine.
This procedure makes parallel nested engines autonomous. Moreover since the engine does not have to interact directly with the database, it can run even in an environment where there are no other Disar components, provided that input data produced by the DiIOAlmEng are transferred on the disk of the master node of the cluster (that with rank 0) or in the cluster shared disk.

Chapter 4

The implemented cloud solution

4.1 Performance analysis of Disar

In this section we show the results obtained from a performance analysis of the DiAlmEngParNs engine. The analysis has been carried out on the AWS infrastructure. For the purpose we have chosen three portfolios of a well-known Italian insurance company. Among all the eebs composing these portfolios, we have selected fifteen of them we consider significant, belonging respectively to a group of five to three different segregated funds. The elementary elaborations are identified by id in E231815-E231829. We have divided the funds into three complexity classes, respectively large, small and medium. For what concerns each segregated fund, each of the five eeb produces different quantities, respectively: market all risks, underwriting mortality risks, underwriting lapse risks, underwriting expense risks, underwriting all risks.
We have run and tested the fifteen elaborations on six different typologies of AWS instances with the respective virtualized feature:

- m4.4xlarge with Intel Xeon E5-2676 v3 (Haswell) 2.4 GHz, 16 vCPUs, 64 GiB of RAM;

- m4.10xlarge with Intel Xeon E5-2676 v3 (Haswell) 2.4 GHz, 40 vCPUs, 160 GiB of RAM;

- c3.4xlarge with Intel Xeon E5-2680 v2 (Ivy Bridge), 16 vCPUs, 30 GiB of RAM;

- c3.8xlarge with Intel Xeon E5-2680 v2 (Ivy Bridge), 32 vCPUs, 60 GiB of RAM;
- c4.4xlarge with Intel Xeon E5-2666 v3 (Haswell), 16 vCPUs, 30 GiB of RAM;
- c4.8xlarge with Intel Xeon E5-2666 v3 (Haswell), 36 vCPUs, 60 GiB of RAM.

The tests have been carried out on more than one instance of the same typology up to a maximum of three instances. The number of risk neutral iterations has been fixed at 50 for all the simulations, a value that as it has often been said previously introduces an acceptable statistical error. Instead, for the natural iterations, in a first testing phase, we have performed multiple simulations by varying this parameter and leaving all the others unchanged. What has came out is a time trend mighty linear varying the number of natural iterations as can be seen in figure 4.1. Taking into account this linear dependence, in the subsequent test phase, all simulations to significantly reduce the overall execution time, have been performed by fixing the number of natural iterations to 1000. To appreciate how the execution time change when varying the instances number, for each elementary elaboration we have plotted and computed the sped-up and the scaling factor.

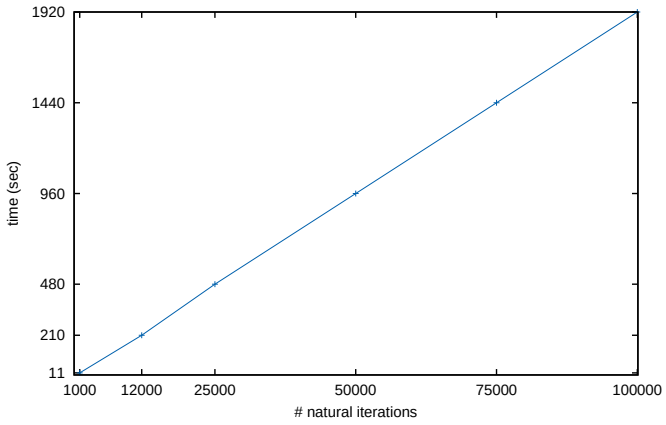


Figure 4.1. Variation of the execution time of the engine on an c3.4xlarge AWS instance with 16 cores increasing the number of natural iteration.

Definition 3. ***Speed-up is a metric for measuring performance improvement when***

executing a task, we can define it as:

$$S = \frac{T_{old}}{T_{new}} \tag{4.1}$$

where T_{old} was the execution time before the improvement and T_{new} is the execution time after the improvement.

The notion of speed-up is a general concept that shows the effect of any performance enhancement. In our case, the improvement consists in the growth of the number of adopted cores. If we address with n the number of the cores adopted in the improved trial execution,

Definition 4. ***following from the speed-up definition, we define the scaling factor ρ as the ratio between the speed-up and the number n:***

$$\rho = \frac{S}{n} \tag{4.2}$$

In figure 4.2 the plots of the speed-up and the scaling factor for the first elaboration belonging to each class of complexity are displayed. We have chosen to show only one elaboration for each class because the behavior is quite analogue within the group, this makes the three cases representative for understanding a general trend. The machine typology is a c3.4xlarge with 16 vCPU. The trend is shown scaling up three machines. However in table 4.1 the full acquired dataset is reported. For each eeb the table shows the belonging fund, the computed quantity and the execution time on the different machine configurations. The time relative to the fund of small complexity are expressed in millisecond to appreciate the scaling at a more fine granularity. What can be observed is that scaling up makes worse performance in accordance with decreasing complexity. Since the inter process communication introduces a certain delay, in case of an elaboration of small complexity, the benefit introduced by the extra instance is exiguous compared to that delay. On the other hand, the situation partially improves when the third instance is added and the the initial disadvantage is partly cushioned. This trend can be observed in plot 4.2(d) where the curve assumes a V behavior. The curve is quite different in 4.2(b) that is relative to the large eeb. In fact after the first hop delay due to the first inter machines communication, the decay goes down more smoothly since the single

process execution time is greater. Three aspects must be taken into account considering these trends. First of all we are in a cloud virtual simulation environment that does not provide us accurate details about the fact that two instances are virtualized within the same rack or not and consequently does not ensure latency constraint except from some specific instance typology. A proof of this is the fact that different trial on the same configuration for the same elaborations (relatively to small ones) returned different values. Secondly, in order to save time we have performed all the computations with 1000 natural iteration that are not enough to ensure an acceptable statistical error and thus a so low value is never used in real computations. Increasing

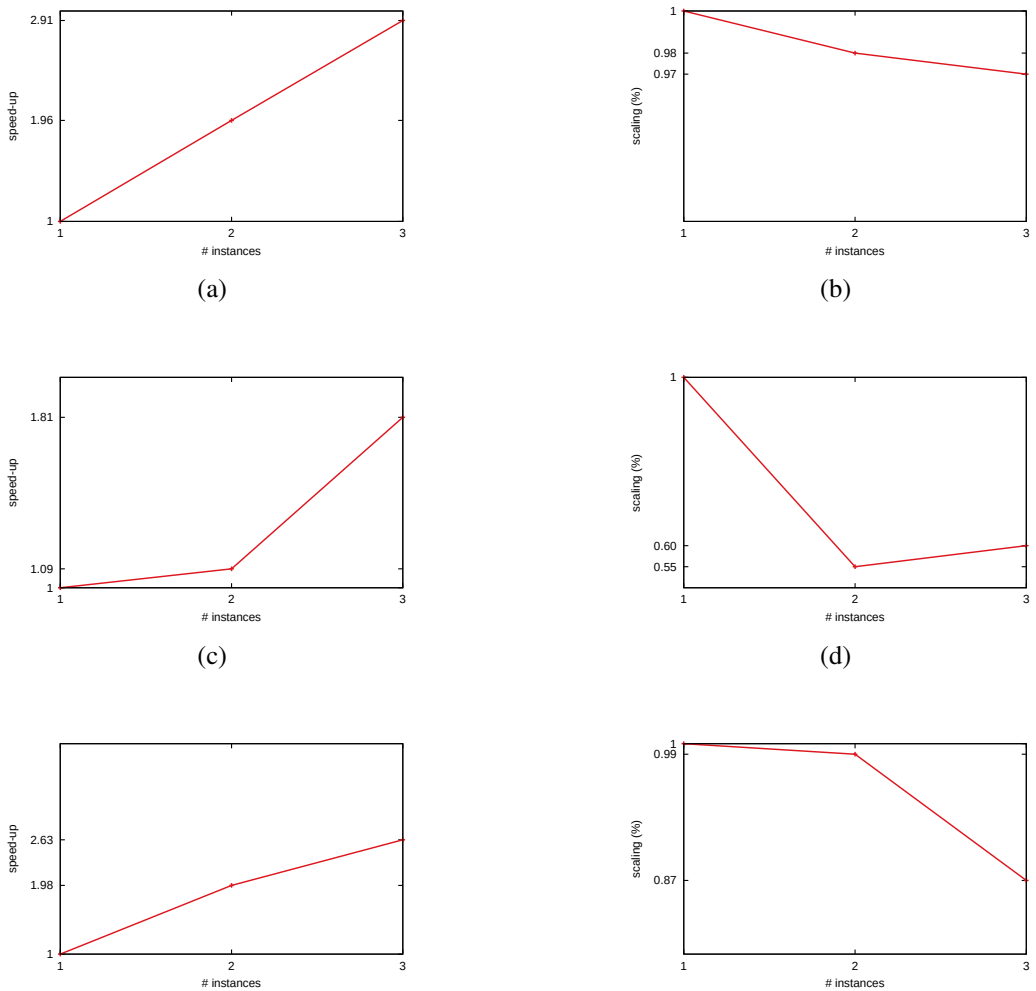


Figure 4.2. Trend of speed-up and scaling factor on c3.4xlarge instances for three elaboration, one for each class of complexity, respectively large, small, medium.

that number would make the complexity highest, making better cushioning for inter process communications. The last and probably the most relevant aspect is that eebs are never elaborated individually but all the elaborations relative to one portfolio are usually treated together, and more than one portfolio at time is evaluated. Usually eeb are elaborated in batch and that will further amortize the waste. A smarter approach to the simulation management will be presented in chapter 6.

elaboration-ID	fund	quantity	# inst.	m4.4.xlarge	# inst.	m4.10.xlarge	# inst.	c3.4.xlarge	# inst.	c3.8.xlarge	# inst.	c4.4.xlarge	# inst.	c4.8.xlarge
E231815	G001	MAR	1	854 sec	1	386 sec	1	823 sec	1	457 sec	1	790 sec	1	387 sec
E231815	G001	MAR	2	431 sec	2	257 sec	2	420 sec	2	232 sec	2	417 sec	2	199 sec
E231815	G001	MAR	3	309 sec	3	147 sec	3	283 sec	3	172 sec	3	281 sec	3	143 sec
E231816	G001	UMR	1	1419 sec	1	660 sec	1	1502 sec	1	798 sec	1	1361 sec	1	681 sec
E231816	G001	UMR	2	751 sec	2	365 sec	2	791 sec	2	420 sec	2	722 sec	2	348 sec
E231816	G001	UMR	3	499 sec	3	250 sec	3	518 sec	3	287 sec	3	481 sec	3	248 sec
E231817	G001	ULR	1	1121 sec	1	475 sec	1	1082 sec	1	583 sec	1	970 sec	1	464 sec
E231817	G001	ULR	2	558 sec	2	248 sec	2	556 sec	2	309 sec	2	523 sec	2	250 sec
E231817	G001	ULR	3	375 sec	3	178 sec	3	367 sec	3	210 sec	3	342 sec	3	172 sec
E231818	G001	UER	1	45 sec	1	390 sec	1	823 sec	1	453 sec	1	819 sec	1	377 sec
E231818	G001	UER	2	434 sec	2	218 sec	2	443 sec	2	231 sec	2	417 sec	2	197 sec
E231818	G001	UER	3	311 sec	3	143 sec	3	286 sec	3	167 sec	3	292 sec	3	142 sec
E231819	G001	UAR	1	2039 sec	1	1180 sec	1	2342 sec	1	1245 sec	1	2115 sec	1	1231 sec
E231819	G001	UAR	2	1025 sec	2	618 sec	2	1190 sec	2	650 sec	2	1114 sec	2	611 sec
E231819	G001	UAR	3	692 sec	3	459 sec	3	810 sec	3	452 sec	3	796 sec	3	427 sec
E231820	G002	MAR	1	11025 msec	1	5082 msec	1	11000 msec	1	3076 msec	1	9095 msec	1	5097 msec
E231820	G002	MAR	2	11010 msec	2	7006 msec	2	10072 msec	2	8024 msec	2	10004 msec	2	3020 msec
E231820	G002	MAR	3	6010 msec	3	5012 msec	3	6063 msec	3	5032 msec	3	7010 msec	3	4081 msec
E231821	G002	UMR	1	11031 msec	1	5057 msec	1	11018 msec	1	3080 msec	1	9091 msec	1	5031 msec
E231821	G002	UMR	2	11015 msec	2	6058 msec	2	6098 msec	2	8009 msec	2	10014 msec	2	5097 msec
E231821	G002	UMR	3	7058 msec	3	2041 msec	3	7005 msec	3	5048 msec	3	5070 msec	3	4082 msec
E231822	G002	ULR	1	10098 msec	1	5062 msec	1	10093 msec	1	3081 msec	1	9093 msec	1	5035 msec
E231822	G002	ULR	2	10087 msec	2	6094 msec	2	7037 msec	2	9002 msec	2	10004 msec	2	5077 msec
E231822	G002	ULR	3	4060 msec	3	4089 msec	3	6029 msec	3	5016 msec	3	4010 msec	3	4056 msec
E231823	G002	UER	1	10098 msec	1	5084 msec	1	10090 msec	1	3077 msec	1	9087 msec	1	5076 msec
E231823	G002	UER	2	10067 msec	2	7032 msec	2	7030 msec	2	8026 msec	2	10014 msec	2	6013 msec
E231823	G002	UER	3	4078 msec	3	5092 msec	3	7091 msec	3	5018 msec	3	4010 msec	3	4064 msec
E231824	G002	UAR	1	11071 msec	1	5084 msec	1	6039 msec	1	3084 msec	1	10031 msec	1	5051 msec
E231824	G002	UAR	2	11026 msec	2	6098 msec	2	6018 msec	2	8056 msec	2	10032 msec	2	6000 msec
E231824	G002	UAR	3	6081 msec	3	5029 msec	3	4077 msec	3	5034 msec	3	6015 msec	3	4068 msec
E231825	G0020	MAR	1	398 sec	1	171 sec	1	379 sec	1	162 sec	1	183 sec	1	165 sec
E231825	G0020	MAR	2	204 sec	2	103 sec	2	191 sec	2	67 sec	2	187 sec	2	89 sec
E231825	G0020	MAR	3	137 sec	3	71 sec	3	144 sec	3	74 sec	3	127 sec	3	67 sec
E231826	G0020	UMR	1	476 sec	1	202 sec	1	464 sec	1	166 sec	1	424 sec	1	196 sec
E231826	G0020	UMR	2	241 sec	2	118 sec	2	229 sec	2	128 sec	2	222 sec	2	106 sec
E231826	G0020	UMR	3	164 sec	3	81 sec	3	160 sec	3	87 sec	3	151 sec	3	188 sec
E231827	G0020	ULR	1	448 sec	1	188 sec	1	432 sec	1	155 sec	1	398 sec	1	184 sec
E231827	G0020	ULR	2	226 sec	2	111 sec	2	216 sec	2	120 sec	2	207 sec	2	158 sec
E231827	G0020	ULR	3	153 sec	3	78 sec	3	150 sec	3	82 sec	3	141 sec	3	73 sec
E231828	G0020	UER	1	399 sec	1	171 sec	1	390 sec	1	159 sec	1	358 sec	1	165 sec
E231828	G0020	UER	2	205 sec	2	102 sec	2	192 sec	2	111 sec	2	188 sec	2	89 sec
E231828	G0020	UER	3	140 sec	3	70 sec	3	149 sec	3	75 sec	3	126 sec	3	64 sec
E231829	G0020	UAR	1	553 sec	1	339 sec	1	530 sec	1	191 sec	1	498 sec	1	233 sec
E231829	G0020	UAR	2	284 sec	2	137 sec	2	269 sec	2	151 sec	2	274 sec	2	126 sec
E231829	G0020	UAR	3	232 sec	3	93 sec	3	203 sec	3	103 sec	3	176 sec	3	91 sec

Table 4.1. DiAlmEngParNs execution time in ms. measured on clusters from 1 up to 3 EC2 instances and on six different instances typology.

4.2 Execution time prediction

In this section we deal with the problem of predict the DiAlmEngParNs engine execution time depending on the underlying IT infrastructure. The adopted approach is that of the machine learning algorithms. In section 4.2.1 we introduce **Weka** [22] [23] an open source machine learning software, while in section 4.2.2 we explain how we have integrated Weka in our project.

4.2.1 Weka

Weka is an open source software developed by the University of Waikato in New Zealand [13] and distributed under GNU General Public License. Weka is the acronym for **Waikato Environment for Knowledge Analysis**. The first original version was released in 1993, the actual version is the 3.7. The software is develop in Java, and is a collection of algorithm and functionality to deal with data mining and machine learning problems. The user can interact with weka through three different modalities: command line, graphical user interface and own code integration. The main provided functionalities are:

- the **Preprocess** that allows to import datasets in the platform using a proper kind of file with *.arg* extension or either a more common CSV file. Moreover the functionality let the user apply some filters to transform the input data and their attributes in order to make them more suitable to scope.
- **Classify** functionality provides the so called classifiers that are classification and regression algorithms. The goal is that of create prediction model and estimate their accuracy.
- **Cluster** section provides instead clustering techniques such as the k-means algorithm.

Multi Layer Perceptron

Multi layer perceptron is a feed forward neural network. Neural network are so called because they emulate the human brain structure. A MLP is applied in those cases in which there is no a linear dependence between the input and output data and so a standard linear perceptron is not applicable [25]. The structure of a MLP is a direct weighed acyclic graph within which vertex are called neurons, in a MLP we distinguish three neuron typology: input, output and hidden neurons. The network consists of two or more layers, at least the input and the output layers, these in the middle are called hidden layers. Each layer is connected to the following, it elaborates data and sends its results to the next one and so on until the output layer is reached. A MLP utilization consists of a first training phase that relies on a supervised learning technique called back-propagation. Once the network is trained and has created its predictive model it processes new data and tries to make

prediction on them. Each layer retrieves a relation between input and output data, creating a function that is able to get that relation, the result is that the final output is a composition of functions.

Random Tree and Random Forest

Random Forest is an ensemble learning algorithm introduced by Leo Breiman. [18] The idea of the algorithm is that of using a combination of classifiers to reduce the error that a single classifier may introduce. The classifiers adopted by the Random Forest are all classification trees. Classification trees suffer from the over fitting problem, that occurs when learning phase is excessive and consequently the model could have been adapted itself to features that are specific only the training set but are not generalized, in the presence of over fitting, performance on the training data will increase, while the performance on new unknown data will be worse. A random forest constructs n different trees, each of them is built by using only a subset of the instance attributes, these subsets are chosen in such a way that they are the more independent as possible, in order to reduce the correlation between the different classifiers. The test instance is evaluated independently on all the classifiers, and then the results are combined together to obtain a final classification based on majority. The single classifiers adopted by the forest is itself a Random Tree that means that the set of attributed on which the model is built is a subset of the original one, these avoid to let the tree complexity grow too much. Weka allow the user to set the number k of trees that will compose the forest. Even if a too large value of k may slow the algorithm, it does not introduce over fitting since there is not correlation among the classifiers.

Out of Bag error. When Random Forests algorithm builds the k trees it creates k datasets of same size as original by randomly re-sampling of data in the training set with replacement, each of these is called a bootstrap data set. Each data set might present duplicate records or several records can be missing with respect to the original datasets and this is what is called Bagging. When the Forest is trained each entry of the original data set is tested on all the trees whose relative data set does not contains it. This is the out-of-bag example. Obviously the classification of such entry is the aggregation of votes only over those tree that does not contain it. Out-of-bag error is the error rate of such tests. Breiman's study on the topic [17] proves that the out-of-bag estimate is as accurate as using a test set of the same size as the training set.

IBk

Instance **B**ased **L**earner is the name with which Weka identify its implementation of the **K-Nearest-Neighbor Algorithm**. It does not relies on the construction of a model but instead try to predict the result of the instance we have to test *just in time*. The idea is that of transform all the instances of the training set in vectors in a multidimensional space. To make the prediction the algorithm search for the k nearest instances to the test instance, and it assigns to the test one, the class that appears more frequently among the nearest instances. To compute the distances, IBk adopts the Euclidean distance, however we should choose also Chebyshev Distance, Filtered Distance, Manhattan Distance, Minkowski Distance, Normalizable Distance or write our own function. Weka permits to the user to set the value of k. The choose of k is very important and depends mainly on data. If it is too small, the method is susceptible to noise in the data. but if it is too large, the decision is smeared out, covering too great an area of instance space.

KStar

K-Star is as the IBk algorithm an instance based classifier and it is very similar to it. The algorithm is quite the same except that the way to find distance between the new entry x and the already present y_i instances is the entropic distance computed as $K^*(y_i,x) = -\log P^*(y_i,x)$ where P^* is the probability of all transformational paths from instance x to y_i by randomly choosing between all possible transformations. [29].

Decision Tables

Decision Tables algorithms create some associations between some conditions and actions to perform in case that set of conditions occur. The table is filled creating a row for each possible condition and it has a number of columns equal to all the possible combinations of the conditions. Then the table is filled by inserting a boolean value for each cell. The result is that each column represents a possible scenario, in which some conditions hold (true) and some others no (false). At each of these set of conditions corresponds an action to perform. Decision Tables should became no more feasible in case there are so many attributes that make hard to manage all possible combinations.

4.2.2 Disar prediction

The approach we have adopted to estimate Disar eebs execution time take advantage of Weka functionality, both for its efficiency and for the ease of integration with the own code. To avoid data set to became mess we have decided to build a training set for each instance typology (the same utilized in the performance analysis), and let the classifiers make prediction separately for each of them. The six training sets are stored in *.arff* files, the file extension utilized by Weka. We have chosen some financial parameters that relative to each elaboration that from several previous studies within the company it has been proved to be significant for the purpose of forecasting. For their inherent complexity but also because it is beyond the scope of this work, we will avoid to describe what each all them represents, but we give in figure 4.3 the header of a single training set to give the reader an idea. A previous test phase has been conducted using the Weka gui to identify the most suitable machine learning algorithms and how to tune them in order to obtain the lowest relative absolute error. The tests have adopted a split percentage of 50% that means that half of the entries where used to train the classifier while the remaining to test it. The most suitable algorithms for our aim results to be those introduced in section 4.2.1. the criteria that we will use to evaluate the goodness of the algorithms, are the average value of the difference between the predicted values and the real values, let call it $\bar{\delta}$ and the study of the distribution of such differences on the tails. Let us denote the real values of interest as Θ and the estimated values using the algorithms as $\hat{\Theta}$, it can be calculated as:

$$\bar{\delta} = \frac{\sum_{i=1}^{n} \hat{\Theta}_i - \Theta_i}{N} \tag{4.3}$$

with N = number of instances in the sample. This average value is important to understand how much on average, the estimated values deviate from the real ones, the more this value is small less error is made by the algorithm. Moreover it allows us to understand if on average the algorithm tends to overestimate or underestimate. In fact if this value is positive it means that in most cases the predicted value is bigger than the real one and viceversa. On the other hand, this value alone is not enough, in fact an algorithm might present a a very small $\bar{\delta}$, but in turn the points that have the bigger absolute value of δ_i, could make the difference, even though they are few. In fact if these points represents an over estimation, never mind, we

will perhaps lose some euros, but in the opposite case we might wait for the result much more than we have expected. that is why it is necessary to study the tails. This study also allows us to see what are the situations in which this problem above occurs, in fact a only case on all the sample in which the δ is for example of one hour is surely most important that 10 cases that make a prediction mistake of 5 minutes.

A summary of the $\bar{\delta}$ reported by each algorithm in this preliminary phase can be see in table 4.2. However the implemented version of the classifiers we have developed adopts the strategy of separate training sets described above, in the last coloumn of the table, the error committed by the algorithms on an unique database, is also reported since the analysis might result to be not accurate on a so small data set. Due to this, only for the analysis aim we will rely on the complete data set. The resulting errors obtained till here are not so significative if we consider that the sample is very small, in addition it has been in turn splitted in half. However, this analysis has been useful to determine which algorithms to discard, if we consider that those not listed in the table yield value of $\bar{\delta}$ very high in any analysis context. The sample used in this phase is the same that has been used in the performance analysis in section 4.1. A more accurate estimate of the algorithms goodness will be given further in chapter 5.1.

```
1  %% Execution time of DiAlmEngParNs
2  %% mal = 1,umr=2, ulr=3,uer=4,uar =5
3  @relation diAlmEngParNs_execution_time
4  @attribute DimPAnaFasce numeric
5  @attribute  DimPAnaFLux numeric
6  @attribute DimPFlu numeric
7  @attribute PolMan_Ntimes numeric
8  @attribute NTIMES numeric
9  @attribute NASSET0 numeric
10 @attribute NAZ0 numeric
11 @attribute NOBB0 numeric
12 @attribute NRISK numeric
13 @attribute quantities numeric
14 @attribute instance_num numeric
15 @attribute execution_time numeric
16
17 @data
```

Figure 4.3. Header of a training set where it is possible to read the parameters utilized for the classification.

	m4.4.xlarge	m4.10.xlarge	c3.4.xlarge	c3.8.xlarge	c4.4.xlarge	c4.8.xlarge	all
IBk	44.8 (s)	24.6 (s)	-19.5 (s)	-4.6 (s)	44.4 (s)	24.1(s)	16.7 (s)
KStar	125 (s)	59.2 (s)	59.0 (s)	36.6 (s)	120.5 (s)	60.5 (s)	12.3 (s)
Random Tree	77.4 (s)	34.4 (s)	-10.2 (s)	0.4 (s)	102.9 (s)	35.4 (s)	1.1 (s)
Random Forest	78.1 (s)	40.2 (s)	14.6 (s)	15.0 (s)	77.8 (s)	42.3 (s)	13. 4 (s)
Multy Layer Perceptron	2.6 (s)	-6.7 (s)	-72.8 (s)	-33.6 (s)	-12.2 (s)	-0.1 (s)	-9,9 (s)
Decision Table	1.3 (s)	3.8 (s)	-70.5 (s)	-28.2 (s)	-1.2 (s)	2.5 (s)	14.5 (s)

Table 4.2. $\bar{\delta}$ reported by each classifier on each of the six training set with a 50% splitting percentage.

4.3 User interface and software tools

4.3.1 GUI

To manage all the functionalities of the DiAlmEngParNs in the cloud environment a java user interface has been realized. The choice of the language has been also driven by the possibility to integrate the in the source code the interaction with WEKA. In figure 4.4 the main view of the interface can be seen. It allows the user to select the number and the typology of instance to launch and start the cluster. The operation takes on average no more that two/three minutes. A scrolling menu permits to select the single **eeb** to elaborate, or eventually all the eebs related to a chosen segregated fund. Once the grid is ready, the number of MPI ranks on which the simulation should be distribute and the number of natural simulations we want to perform, must be selected. Three buttons make possible the transfer of the input/output data toward/from the cluster and the simulation start up.

Before launching the simulations, the user has the possibility to have an idea of

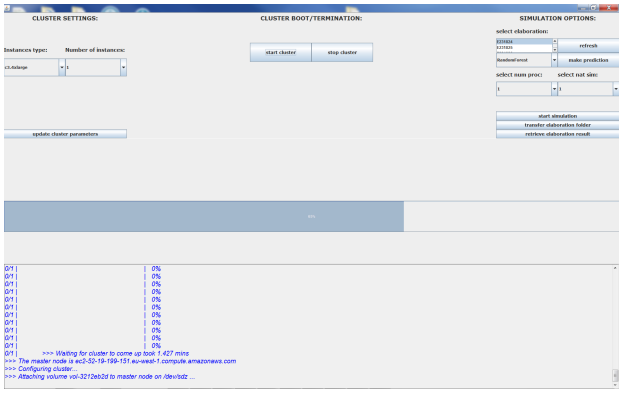


Figure 4.4. User interface of the system.

the execution time they will take. From the same menu designed for select the elaborations, the eeb whose we want to predict the execution time, can be chose,

while in the menu immediately below, the desired machine learning algorithm is selectable. Pressing the prediction button the user can have the estimated time for that elaboration on different type and number of instances.

Auxiliary scripts

Quite all the functions activated by buttons pressure, are performed by python scripts, this make their implementation quite simple since many of these operations contain file manipulation. Moreover python adoption does not introduce an extra dependence since it is already required by Starcluster. The first script is called when the cluster parameters are set, it modify some fields of the starcluster *config* file on the user machine, including the AMI id. These parameters are afterward used during the boot phase. An other script recall the starcluster executable and launch the cluster with the given configuration. After this, a series of scripts start, one after the completion of the previous one, once the instances have booted, the *"mpd.hosts"* file is generated and transferred on the master node of the cluster. The file is used by the mpi library to known all the machines on which the MPI ranks should be distributed together with each machine's relative available cores. This last parameter is retrieved during the file creation from a map contained in the java code, that keep note for each instance type the number of maximum cores available. The same class that contains the map that is named Utility.java, wraps several information about AWS that must be periodically update, including instances prices. SSH infra-nodes password-less communication is set on the nodes through the generation and exchange of rsa keys, this allows inter process communication for the MPI library. A virtual volume is shared among all the cluster nodes. A progress bar and a log panel show the succession of all these phases. When the instances are booted, it is notified to the user through the log view. After segregated funds are selected, a script compresses all the input data necessary to the simulation, that have been previously generated by the DiIOAlmEng procedure, transfers them to the shared device and unzip their content. All those operation are managed through ssh connection between the local machine and the master node. Immediately after the same script generate the *run* file that contains the *mpirun* command with the relatives parameters set until now from the user through the UI. Once that file is generated, it is made executable and it is send on the master node. At this point, it is possible to start the engine, the button *start simulation* with its relative script executes the run file, showing the simulation progress on the log windows and the progress bar. After its completion, the output

data, together with log and error files are taken back to the user workstation. The user may at this point start new elaborations or chose to stop the cluster, in this case every component is shut down stopping paying for it.

Prediction implementation

The prediction function is achieved taking advantage of Java Weka classes. Every time a new elaboration is successfully completed, a new entry is inserted in the six training sets. The system supports prediction with all the algorithms introduced in section 4.2.1. When the prediction is started six classifiers are build from the six training sets, while new instances relative to the elaboration to test are appended in others arff file containing the test sets. A script reads the parameters in figure 4.3 from the input files and inserts n new entries in each test set. The last n instances of the single test set are tested on the relative classifier, and the results are communicated to the user on the log area of the GUI. Regarding the IBk algorithm, the parameter k that represents the number of neighbors on which the classification is given, is set to 5, such value after different trials is the one that returns the lowest error. The number of trees in the random forest s set to 500, while the number of attributes on which each tree is build in equal to 500. The evaluation measure adopted for the decision table is the mean absolute error (MAE). The MAE measures the average magnitude of the errors in a set of forecasts, without considering their direction. It measures accuracy for continuous variables. The MAE is the average over the verification sample of the absolute values of the differences between forecast and the corresponding observation and it is a linear score which means that all the individual differences are weighted equally in the average. The training time for the MultiLayerPerceptron is set to $1000ms$. All these features tunings have been calibrated taking advantages of several estimation trials.

4.3.2 AMI content

The **AMI (Amazon Machine Images)** id stored in the configuration file we mentioned previously, is relative to a system image stored on the AWS account. Once this image is saved in the own account, the user we has the possibility to run instances with the desired (virtualized) hardware features, starting from the AMI snapshot. To realize the AMI there are some different way. One consist in starting an instance based on a public AMI in the marketplace, custom it and do a snapshot

```
Random forest of 500 trees, each constructed while considering 4 random features.
Out of bag error: 51875.7837

All the base classifiers:


RandomTree
==========

DimPAnaFasce < 104.5
|   DimPFlu < 1191 : 5326.47 (15/461313.85)
|   DimPFlu >= 1191
|   |   quantities < 1.5
|   |   |   instance_num < 2.5 : 103000 (1/0)
|   |   |   instance_num >= 2.5 : 71000 (1/0)
|   |   quantities >= 1.5
|   |   |   DimPAnaFasce < 72 : 102000 (1/0)
|   |   |   DimPAnaFasce >= 72
|   |   |   |   instance_num < 1.5 : 202000 (1/0)
|   |   |   |   instance_num >= 1.5 : 118000 (1/0)
DimPAnaFasce >= 104.5
|   PolMan_Ntimes < 918
|   |   DimPAnaFasce < 193.5
|   |   |   instance_num < 1.5 : 339000 (3/0)
|   |   |   instance_num >= 1.5 : 137000 (3/0)
|   |   DimPAnaFasce >= 193.5 : 188000 (1/0)
|   PolMan_Ntimes >= 918
|   |   DimPAnaFLux < 19134
|   |   |   DimPAnaFLux < 2321
|   |   |   |   instance_num < 1.5 : 388000 (6/4000000)
|   |   |   |   instance_num >= 1.5
|   |   |   |   |   quantities < 2.5 : 257000 (1/0)
|   |   |   |   |   quantities >= 2.5
|   |   |   |   |   |   instance_num < 2.5 : 218000 (1/0)
|   |   |   |   |   |   instance_num >= 2.5 : 143000 (1/0)
|   |   |   DimPAnaFLux >= 2321
|   |   |   |   DimPAnaFasce < 1468
|   |   |   |   |   instance_num < 1.5 : 475000 (2/0)
|   |   |   |   |   instance_num >= 1.5
|   |   |   |   |   |   instance_num < 2.5 : 248000 (2/0)
|   |   |   |   |   |   instance_num >= 2.5 : 178000 (2/0)
|   |   |   |   DimPAnaFasce >= 1468 : 250000 (1/0)
|   |   DimPAnaFLux >= 19134
|   |   |   instance_num < 2.5 : 618000 (1/0)
|   |   |   instance_num >= 2.5 : 459000 (1/0)

Size of the tree : 37
```

(a)

```
Decision Table:

Number of training instances: 45
Number of Rules : 18
Non matches covered by Majority class.
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 64
        Merit of best subset found: 70518.964
Evaluation (for feature selection): CV (leave one out)
Feature set: 1,4,11,12

Rules:
=============================================================================
DimPAnaFasce      PolMan_Ntimes     instance_num      execution_time
=============================================================================
'(184.2-359.4]'   '(739.2-832.8]'   '(1.8-2]'         111000.0
'(-inf-184.2]'    '(739.2-832.8]'   '(1.8-2]'         115000.0
'(1585.8-inf)'    '(926.4-inf)'     '(2.8-inf)'       459000.0
'(1410.6-1585.8]' '(926.4-inf)'     '(2.8-inf)'       214000.0
'(709.8-885]'     '(926.4-inf)'     '(2.8-inf)'       145000.0
'(-inf-184.2]'    '(-inf-177.6]'    '(-inf-1.2]'      5073.8
'(184.2-359.4]'   '(739.2-832.8]'   '(2.8-inf)'       78000.0
'(-inf-184.2]'    '(739.2-832.8]'   '(2.8-inf)'       78750.0
'(1585.8-inf)'    '(926.4-inf)'     '(-inf-1.2]'      1180000.0
'(1410.6-1585.8]' '(926.4-inf)'     '(-inf-1.2]'      567500.0
'(709.8-885]'     '(926.4-inf)'     '(-inf-1.2]'      388000.0
'(-inf-184.2]'    '(-inf-177.6]'    '(1.8-2]'         6457.6
'(184.2-359.4]'   '(739.2-832.8]'   '(-inf-1.2]'      188000.0
'(-inf-184.2]'    '(739.2-832.8]'   '(-inf-1.2]'      220750.0
'(-inf-184.2]'    '(-inf-177.6]'    '(2.8-inf)'       4252.6
'(1585.8-inf)'    '(926.4-inf)'     '(1.8-2]'         618000.0
'(1410.6-1585.8]' '(926.4-inf)'     '(1.8-2]'         306500.0
'(709.8-885]'     '(926.4-inf)'     '(1.8-2]'         237500.0
=============================================================================
```

(b)

```
Linear Node 0
    Inputs    Weights
    Threshold    2.15164597135131
    Node 1    -2.3871620828549496
    Node 2    0.15092043429502552
    Node 3    -0.43378164424729065
    Node 4    -0.7284281789614525
    Node 5    0.6776246713680432
    Node 6    -0.27256193664858136
Sigmoid Node 1
    Inputs    Weights
    Threshold    3.693728569310362
    Attrib DimPAnaFasce    -0.08877368856144535
    Attrib DimPAnaFLux    -0.7207529622027992
    Attrib DimPFlu    -0.35151496510280855
    Attrib PolMan_Ntimes    0.21324633465641035
    Attrib NTIMES    0.14544066947148482
    Attrib NASSET0    -0.12321642909475483
    Attrib NAZ0    -0.6948158481938373
    Attrib NOBB0    0.062465014890821076
    Attrib NRISK    -0.8951598784374721
    Attrib quantities    -0.2918500347827356
    Attrib instance_num    1.9123522217398663
```

(c)

Figure 4.5. Examples how how some machine learning algorithms build its own inference rules, respectively: 5.2(b): Random Forest, 4.5(b): Decision Tables, 4.5(c): MultyLayer-Perceptron.

of the instance creating an own private AMI. An other possibility is to import into AWS an image obtained doing a snapshot of an on premise virtual machine already

running on the top of a **VMware vSphere system**. [1] The AMI used for Disar has been created through the first option, we start from a 64 bit version of **CentOS 6.5**. The CentOS Linux distribution is a stable platform derived from the sources of Red Hat Enterprise Linux (RHEL). Due to its stability it is suitable for servers in Linux environment. On the OS we have installed the **Intel Cluster Studio XE** environment that even if it embeds more software modules than those we need, it contains all necessary libraries. For our purpose it was important to set the environment variables related to the following runtimes :

- Intel MKL math library
- Intel MPI library (impi)
- Intel Fortran (ifort)

On the OS we have created a work directory that contains the last version of the **DiAlmEngParNs** engine. Since the engine is written in Fortran, we need ifort runtime to run it. the mkl library is used for the generation of pseudo random numbers necessary to the Monte Carlo.

[1] VMware vSphere is the brand name for VMware's suite of virtualization products. Before 2009, VMware vSphere was known as VMware Infrastructure.

Chapter 5

Experimental results

5.1 Experimental results

In this section we will report the results obtained by the application of the algorithms introduced in section 4.2.1. They have been tested on 25 more elaborations and in different contexts. For each algorithm we will discuss pros and cons and we will analyze the reasons that make them perform well or bad depending on the prediction we want to obtain. According to what stated in section 4.1, we have worked moving into two direction: for the first we have make 190 predictions using a training set of 126 instances on different instance typology, for the second we have make 27 predictions with a training set of 41 elaborations, always on the same single instance typology. Despite the software implemented solution relies on the second methodology, we have decided to show both results because the tests on the single instance are too few and they could sometimes show significant errors since they could be not enough. Subsequently we have actually performed the simulations, comparing then the predicted and real values in both situations. We have plotted for each elaboration a point having coordinates $x = \Theta_i$ and $y = \hat{\Theta}_i$, i.e. real and estimated execution time. In figure 5.7 we have instead plotted the distribution of δ to study the behavior in correspondence of the tails. The red line represents the bisecting line. If the point relative to an elaboration is placed up the red line it means that the prediction for that elaboration has been over estimated, in the contrary if the point lies under the line the prediction has been under estimated. Moreover, observing the plot is possible to understand in which situation the classifier takes a certain behavior. For example if the plot takes the shape similar to a triangle with the point to left and the base to the right, it means that it estimates well for small

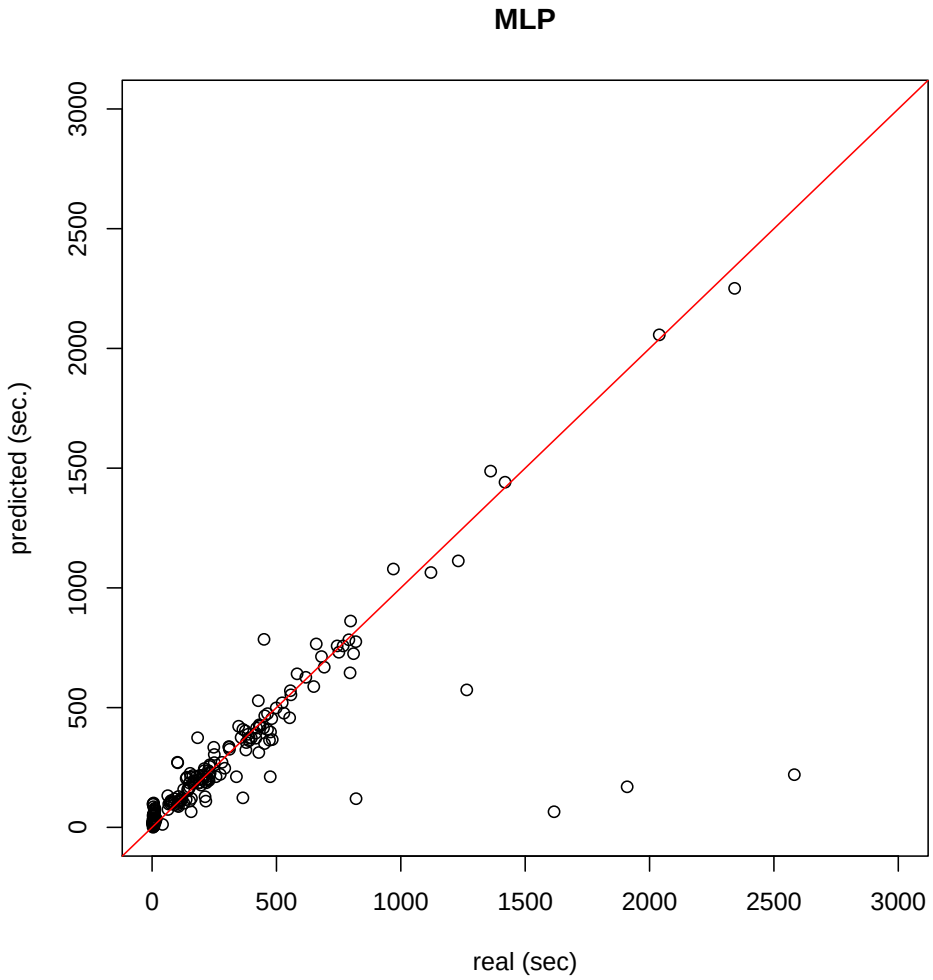


Figure 5.1. Plot of real and estimated execution time using Multi Layer Perceptron algorithm on the unified data set.

time values and fails for big ones. Viceversa the base to left and the point to right means the contrary. The resulting shape could be even irregular.

5.1.1 Multi Layer Perceptron

The initial seed for random number generation has been set to 7 with a number of epochs equal to 1000 that through different trials results to be a good value to avoid network over fitting. Reading table 5.1 we can see that the neural network yield a $\bar{\delta} = -33sec.$, meaning that it slightly under estimates. According to this, what we can observe in figures 5.1 is that most of the misplaced points lies in the

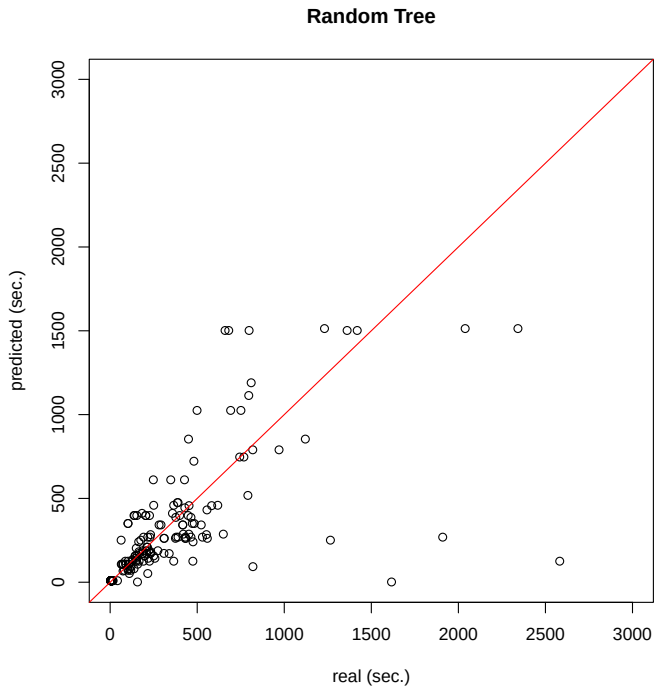
bottom-right side of the plot. This means that the algorithm tends to under estimate especially when Θ_i grows more that about 850. We can also note a vertical points disposition in the axis origin, it represents those estimation errors related to all those elaborations whose Θ_i is of the order of seconds. This highlights that the algorithm fails in estimating this class of elaborations. The most plausible cause of this is the fact that the samples representing this type of elaborations in the training set are very few, as also those whose time is closer to 50 minutes, causing that the network is unable to catch the patterns related to them. If we observe the tails distribution in figure 5.7(a) we see that it is quite populated in the left side. The standard deviation is equal to 6.5 minutes, this implies that the 68% of the sample fall into an error of about 7 minutes.

5.1.2 Random Tree and Random Forest

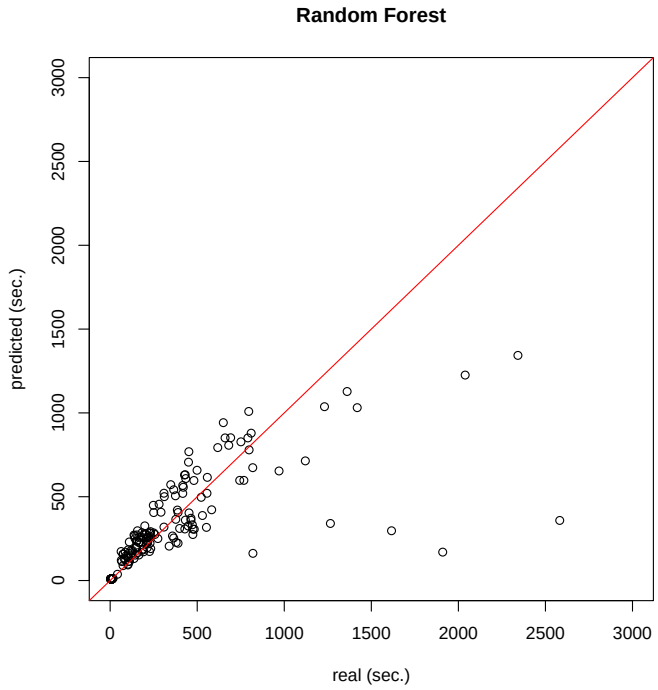
For both Random Tree and Random Forest we have used the value $k = 8$ for the number of randomly chosen attributes on which to construct the trees. The forest is made of 500 trees. What is immediately clear to the observer eye comparing figures 5.2(a) and 5.2(b) is the better accuracy of Random Forest in the left side of the plot, this is predictable since Random Forest apply something more with respect to Random Tree that is the final voting among the classifications obtained by all the trees composing the forest. In fact with respect to Random Tree, Random Forest is able to recover some wrong over estimations that can be see as on horizontal line in figure 5.2(a) in corrispondence of the value of about 1500 on the y axis. Both the algorithms fail on the same samples Multi Layer Perceptron do for big value, while they behave slightly better on very small values with respect to MLP, since they fail only on Θ_i with value lower than $6\backslash 7$ seconds, while MLP under approximately 1 minutes. The mean committed error is of 38 seconds. The distribution in figure 5.7(b) presents a standard deviation of 6.10 minutes.

5.1.3 Ibk

Ibk algorithm shows the second biggest vale for δ that is equal to -66.7 seconds. It is difficult to tune the parameter $k =$ number of nearest neighbors once for all since it depends not only from the general density of sample, but in particular from the local density we are working around. If in that zone the sample is dense enough, an high, but limited value of k results in a good approximation since it gives an average



(a)



(b)

Figure 5.2. Plot of real and estimated execution time using Random Tree 5.2(a) and Random Forest 5.2(b) algorithm on the unified data set.

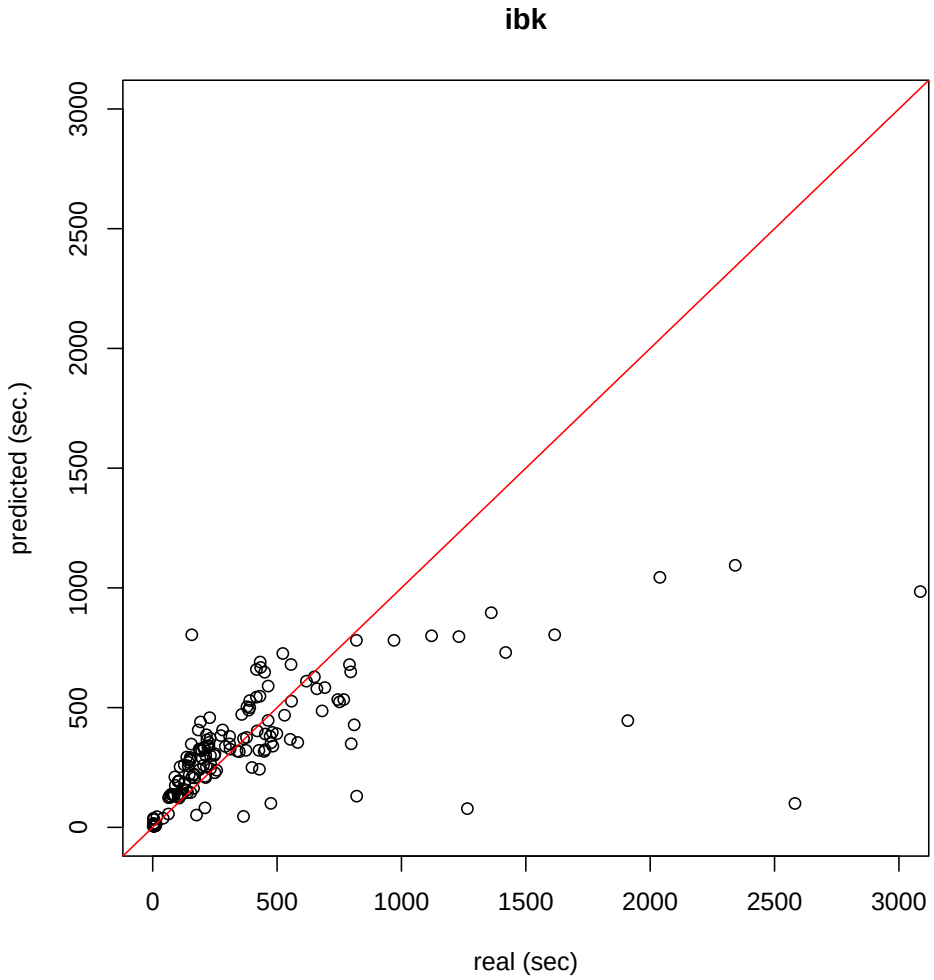


Figure 5.3. Plot of real and estimated execution time using ibk algorithm on the unified data set

value among the neighbors, but if the zone is not enough populated we risk to embed neighbors that are too far from the zone our test will be located. A good value for the both the tests results to be $k = 8$. Looking at plot in figures 5.3 we can observe that generally the algorithm tends to under estimate especially after elaborations whose real value is bigger than 15 minutes. In fact for how the algorithm is designed, the instances in the right side are too distant among each others. The dense distribution of the left tail can be observed in figure 5.7(d). The standard deviation is of 6.7 minutes, meaning that the 68% of the sample fall into an error of about 8 minutes.

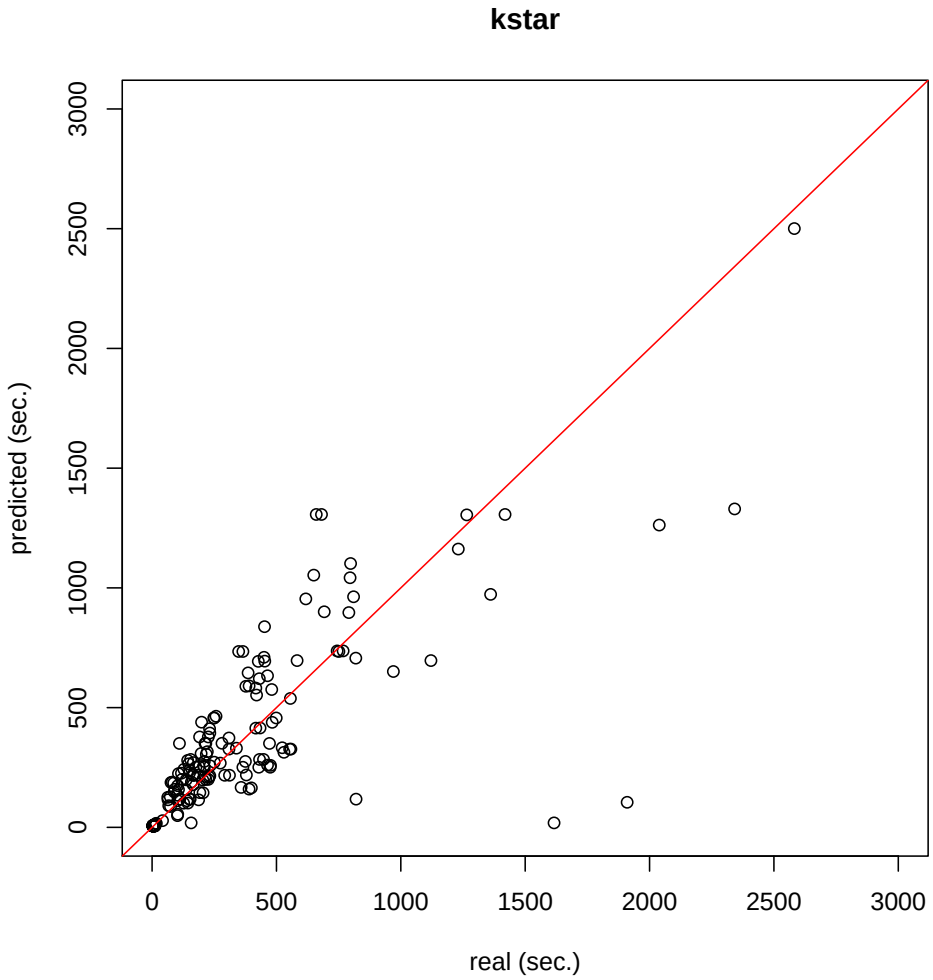


Figure 5.4. Plot of real and estimated execution time using kstar algorithm on the unified data set.

5.1.4 kStar

Kstar achieve the smaller value for $\delta = -1.6sec.$, but what is interesting observing figure 5.4 is that this is due to a balancing between underestimations and overestimations. In fact it fails giving an underestimation, in predicting some values that are the same for which MLP fails, even if, in that zone kstar is a little bit more accurate. While in the upper part of the plot we can observe that it suffer of overestimation for some points that instead MLP guesses. The two algorithms seem to be quite complementary from this point of view. Its standard deviation is equal to 6.7 minutes.

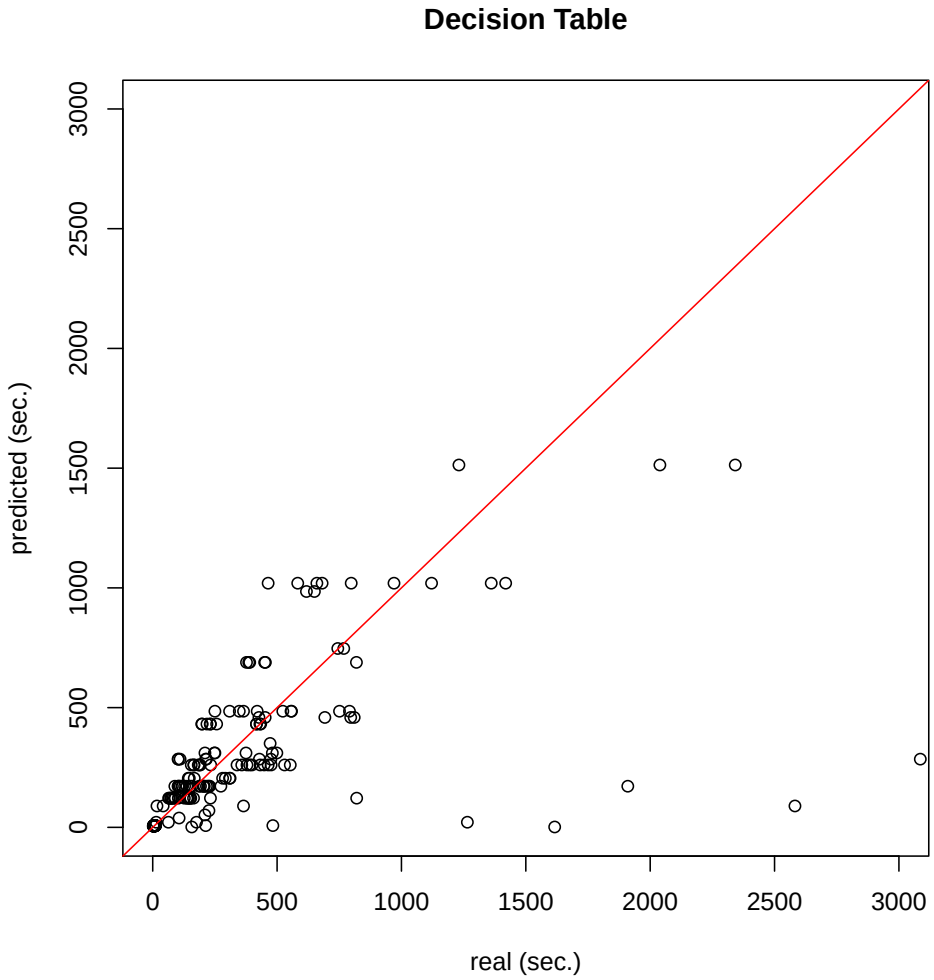


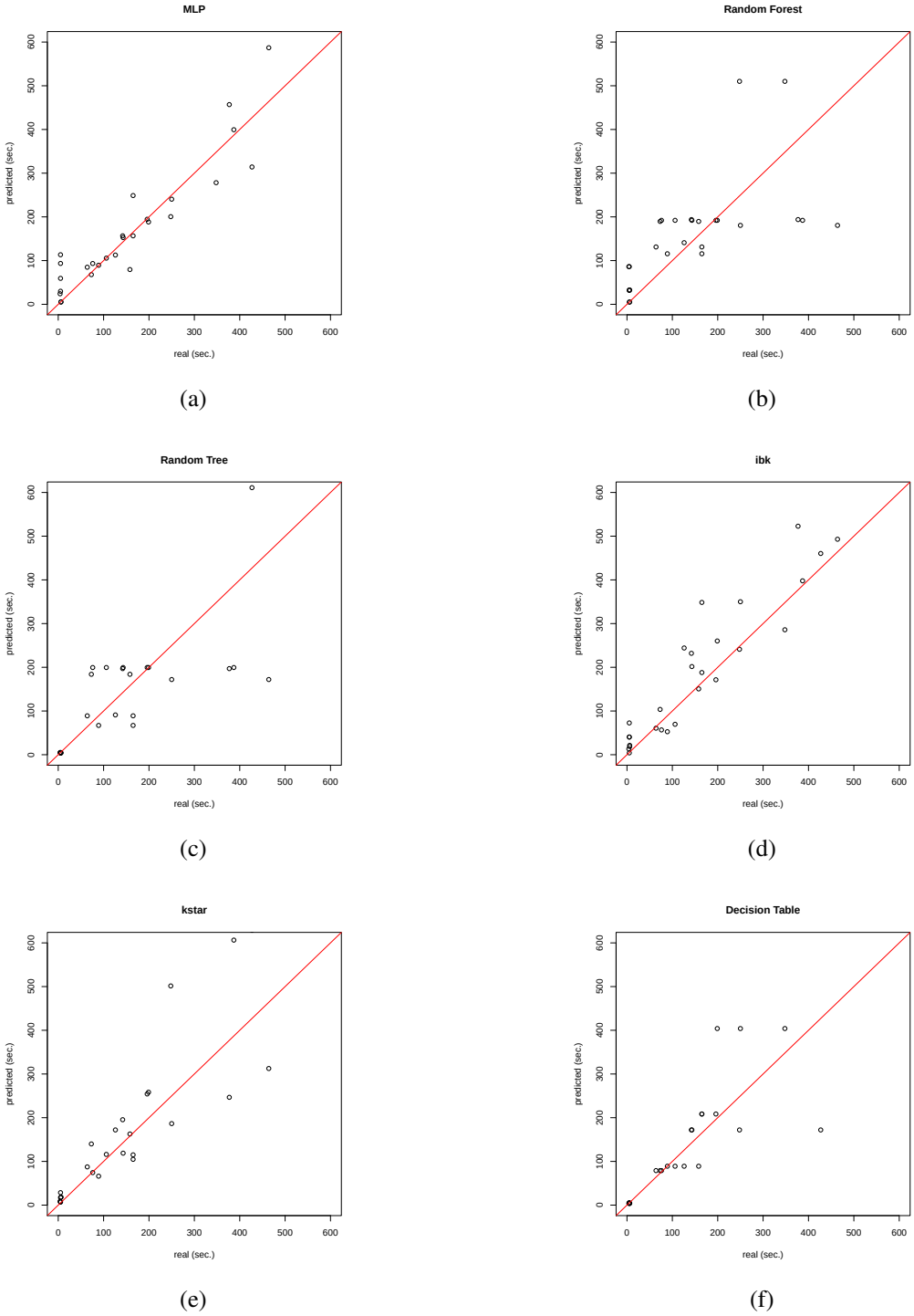
Figure 5.5. Plot of real and estimated execution time using Decision Table algorithm on the unified data set.

5.1.5 Decision Table

Decision Table shows the higher values for δ respect to other classifiers. Moreover the behave in completely different on the two data sets figure 4.5(b) and figure 5.6(f). The tails in figure 5.7(f) are very wide in both direction, with a standard deviation of 7.3 minutes with a resulting $\delta + \sigma = 8.5$ minutes. Due to its implementation, Decision Table's output are only some discrete values, this is observable looking at figure 4.5(b), in which we can see some horizontal lines.

5.1.6 Comparison

An important and main limitation encountered in carrying out this work is surely related to time and costs. In fact, the used sample contains a few instances for an exhaustive accurate estimate, but enrich it during the evolution of the work would involve huge expenses for testing, having to repeatedly turn on different instances, of which the most performing have a considerable cost, considering that already for tests carried out up to here the expense was not indifferent. Also make the sample homogeneous would require many elaborations with fairly distributed execution times in order to avoid the problem encountered for those elaborations of short duration. This would result in many hours of testing. However the problem is not limiting as the system is designed to come up to steady state and make training more and more accurate as new elaborations are performed. Despite this we have shown how quite satisfactory results can be achieved applying some algorithms. In figures 5.8 we have compared the three classifiers that better achieve the goal. For each real value three corresponding points, red, green, blue represent the time predicted by respectively :MLP, kstar and Random Forest. To have a better view, we have divided the image into two, in order to have a zoom-in in the region where points are too close to each other. In figure 5.8(a) it is possible to observe elaborations whose real value is less than 800 seconds. In the first left side we can better observe what has been previously discussed. MLP fails in predicting values below about 50 seconds by over estimation. In the same region Random Forest gives the same values for all the elaborations, this is represented by the blue horizontal line. A better results is achieved instead by kstar. In the region between 100 and 800 MLP is the best accurate, while the others two classifiers fluctuate around the bisector line, especially kstar, that is why its mean error results so low. In figure 5.8(b) we can observe the remaining region of the plot. Excluding some common mistakes, MLP is still the most accurate even in this region. What is interesting to note is that the error committed by MLP for small elaborations, even if completely wrong, all represent overestimation with a positive gap of in average 100 seconds. This implies that such mistakes are not evident on the tails of figure 5.7(a), since it is a value closer to 0, and so they simply broaden the base of the distribution. Summing up we can derive that MLP, is the most suitable classifier for our purpose, despite some inaccuracies, that could be probably correct by enriching the sample in some less dense regions.



(a)

(b)

(c)

(d)

(e)

(f)

Figure 5.6. Plot of real and estimated execution time using: 5.6(a) MLP, 5.6(b) Random Forest, 5.6(c) Random Tree, 5.6(d) Ibk, 5.6(e) kStar, 5.6(f) Decision Table algorithms on the c4.8xlarge instances.

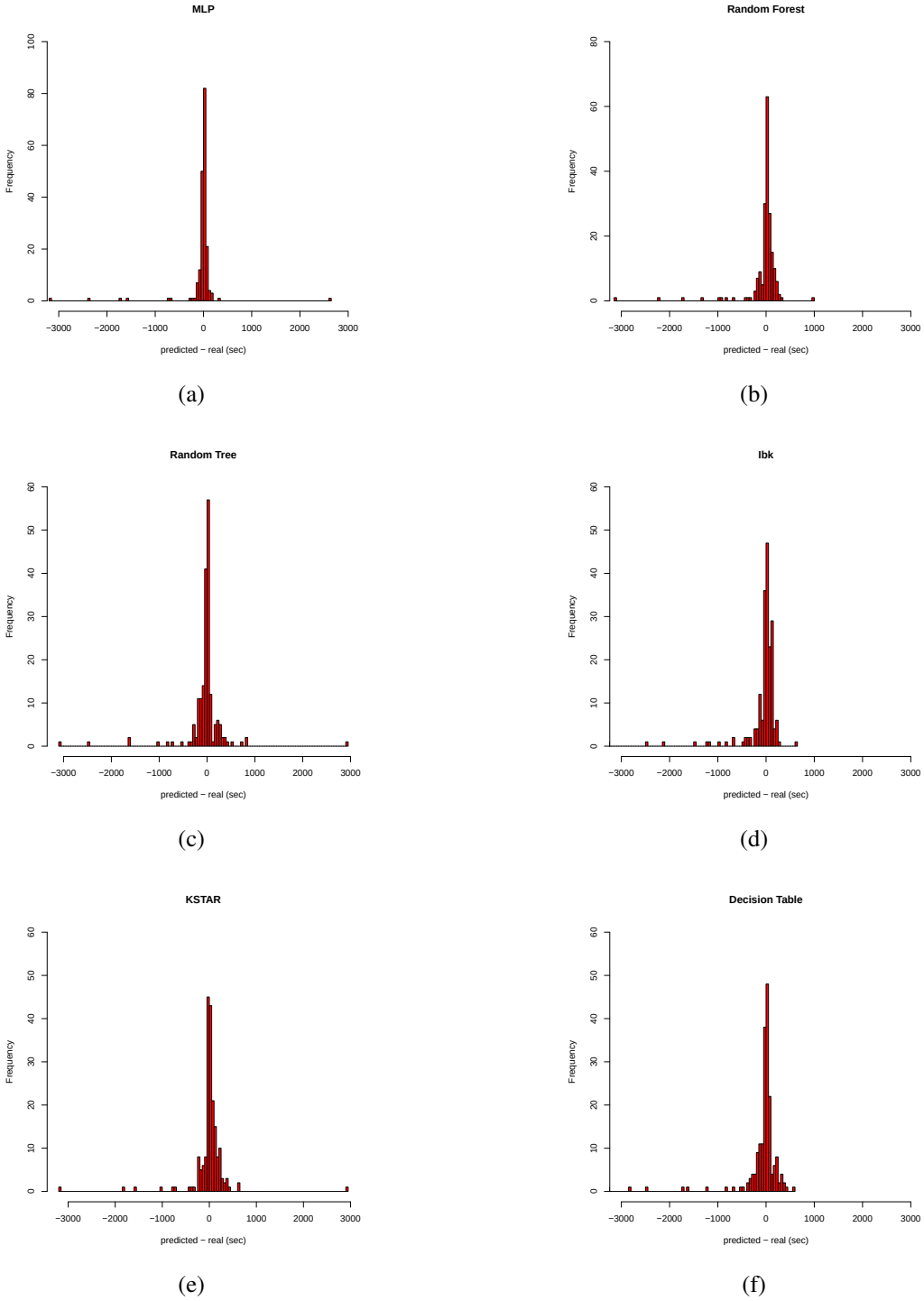
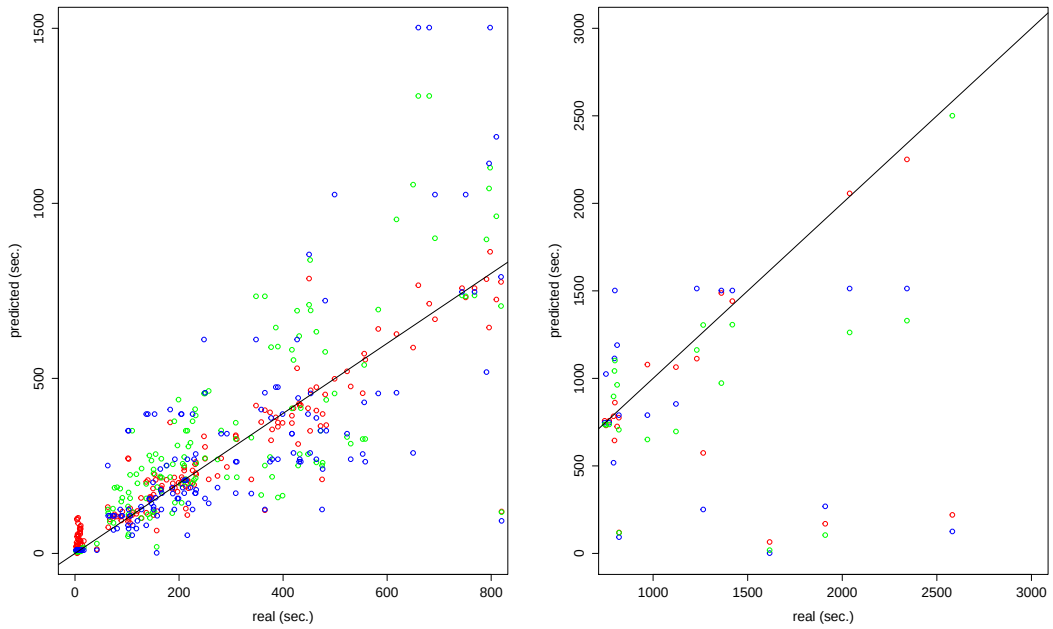


Figure 5.7. Histogram of δ distribution using: 5.7(a) MLP, 5.7(b) Random Forest, 5.7(c) Random Tree, 5.7(d) Ibk, 5.7(e) kStar, 5.7(f) Decision Table.



(a) (b)

Figure 5.8. Plot of real and estimated execution time achieved by MLP (red), kstar (green) and Random Forest (blue).

	c4.8.xlarge	all
IBk	31.9 (s)	-66.7 (s)
KStar	31.9 (s)	-1.6 (s)
Random Tree	17.4 (s)	-33.5 (s)
Random Forest	24.3 (s)	-38.37 (s)
Multy Layer Perceptron	11.0 (s)	-33.0 (s)
Decision Table	65.8 (s)	-75.8 (s)

Table 5.1. Values of $\bar{\delta}$ reported by each classifier on the test sets.

real	ibk	kstar	MLP	Random Tree	Random Forest	Decision Table
650000.0	628563.1	1053398.92	588015.18	287000.0	941883.978	985000.0
2342000.0	1094333.333	1329790.228	2250939.379	1513333.333	1343113.495	1513333.333
768000.0	533763.1	736916.991	757703.123	747000.0	597616.308	747000.0
250000.0	228000.0	456875.675	304310.45	459000.0	405461.709	310666.667
1121000.0	800000.0	696509.475	1064248.33	854000.0	713842.881	1019666.667
7037.0	8152.2	7535.491	4885.696	9219.961	9156.241	7301.6
390000.0	500222.222	590798.881	361895.171	475000.0	405912.57	689000.0
819000.0	781125.0	706663.41	775510.859	790000.0	673266.889	689000.0
5048.0	7024.875	6606.434	52621.184	9219.961	9156.241	5206.75
660000.0	578777.778	1306746.704	766014.446	1502000.0	850976.132	1019666.667
63000.0	55896.75	125350.498	132372.91	251000.0	173244.257	21500.0
143000.0	256250.0	279577.987	152838.239	157000.0	249960.545	205200.0
10004.0	7160.0	7340.244	26806.897	9219.961	9156.241	7301.6
5070.0	6590.222	7449.17	50338.34	9219.961	9156.241	5206.75
205000.0	330250.0	144380.13	217960.893	398000.0	273317.681	171875.0
192000.0	440375.0	145046.246	187248.929	269000.0	171384.584	171875.0
475000.0	100261.375	251000.0	211485.359	125500.0	274463.997	284832.135
118000.0	138000.0	226275.613	113307.539	71000.0	148316.838	171875.0
798000.0	349078.875	1101873.187	861651.051	1502000.0	779461.776	1019666.667
339000.0	318400.0	330695.411	211448.626	171000.0	205218.231	260666.667
6013.0	10291.5	7354.336	7780.196	9219.961	9156.241	7301.6
377000.0	375666.667	589171.931	323133.238	387000.0	365170.241	689000.0
348000.0	316000.0	734486.118	422313.187	611000.0	571966.663	485000.0
464000.0	446555.556	633122.996	474936.26	387000.0	371578.463	1019666.667
81000.0	138111.111	188939.396	105439.074	71000.0	129452.554	122000.0
157000.0	804203.875	18671.003	65316.709	1631.0	296954.233	1631.0
137000.0	155625.0	157049.429	204360.089	398000.0	271471.726	122000.0
365000.0	45764.375	251000.0	123365.13	125500.0	253927.952	89333.333
64000.0	123777.778	115138.0	74695.291	107500.0	120571.347	122000.0
4077.0	6046.444	5790.847	47609.13	9219.961	9156.241	5206.75
481000.0	396700.0	575662.859	454068.937	722000.0	596887.154	310666.667
309000.0	348545.455	373329.25	337509.67	261923.077	520969.464	205200.0
2041.0	5251.3	6261.044	19405.938	9219.961	9156.241	5206.75
176000.0	51399.25	251000.0	210410.827	251000.0	258135.916	21500.0
796000.0	649375.0	1042449.454	645188.95	1114000.0	1008690.152	459000.0
390000.0	529222.222	159776.579	373975.726	269000.0	222551.154	260666.667
472000.0	380703.875	350500.0	364043.092	350500.0	332643.229	350500.0
76000.0	138000.0	187495.832	111699.757	107500.0	157664.975	122000.0
188000.0	240875.0	254640.909	200087.635	171000.0	182098.717	260666.667
229000.0	458000.0	209663.997	195986.952	269000.0	223053.336	171875.0
5018.0	7024.875	6381.009	55007.354	9219.961	9156.241	5206.75
309000.0	380000.0	326922.285	334572.012	172000.0	318860.146	485000.0
106000.0	195000.0	224435.405	128831.071	107500.0	162316.529	171875.0
10067.0	8601.667	8372.082	63467.214	9219.961	9156.241	7301.6
386000.0	489666.667	645137.496	388649.199	475000.0	420464.045	689000.0
147000.0	219777.778	265341.734	160527.33	143000.0	197131.768	205200.0
4078.0	6673.75	5958.365	99223.339	9219.961	9156.241	5206.75
137000.0	293777.778	198596.763	116302.766	80333.333	131238.085	171875.0
1419000.0	730363.636	1306746.704	1441394.134	1502000.0	1031403.464	1019666.667
167000.0	206818.182	216520.199	193479.215	172000.0	225324.042	205200.0
434000.0	667777.778	415010.174	423761.39	261923.077	608244.65	431500.0
3076.0	38017.625	7204.345	10513.183	9219.961	9156.241	7416.412
213000.0	208000.0	350499.99	128005.092	209000.0	232001.284	284832.135
4068.0	8859.727	5645.385	22512.53	9219.961	9156.241	5206.75
14000.0	13162.125	15027.054	24370.664	9219.961	11973.541	21500.0
73000.0	126416.667	127930.769	100255.533	107500.0	157019.83	122000.0
274000.0	383125.0	268382.517	221330.236	188000.0	250307.581	171875.0
4089.0	5171.625	5530.336	14060.833	9219.961	9156.241	5206.75
3081.0	7782.375	7755.82	12767.054	9219.961	9156.241	7416.412
144000.0	271375.0	101412.332	161326.054	154500.0	151773.064	122000.0
618000.0	610545.455	954114.604	626679.019	459000.0	793219.336	985000.0
164000.0	163000.0	219792.288	216427.846	241000.0	258413.086	122000.0
399000.0	250000.0	164956.529	372899.461	398000.0	310769.342	260666.667
4064.0	13041.0	5793.885	17438.716	9219.961	9156.241	5206.75
692000.0	583777.778	900294.251	668876.193	1025000.0	851848.0	459000.0
432000.0	690500.0	283712.199	422555.765	269000.0	360359.109	260666.667
102000.0	191500.0	49000.0	271913.536	350500.0	137819.231	284832.135
103000.0	129666.667	111465.8	104551.704	71000.0	97335.099	171875.0
375000.0	321777.778	275395.466	402922.373	261923.077	505517.093	310666.667
751000.0	524000.0	734486.118	731303.058	1025000.0	827514.387	485000.0
810000.0	428125.0	963031.932	725324.803	1190000.0	879209.639	459000.0
151000.0	292000.0	228572.668	109474.77	125500.0	196698.922	171875.0
91000.0	175666.667	158860.457	119463.043	107500.0	147548.285	122000.0
233000.0	263750.0	257255.164	225576.265	181666.667	190467.154	260666.667
1910000.0	445666.667	104711.637	169442.983	269000.0	169602.563	171875.0
191000.0	323250.0	377565.897	216898.51	125500.0	255402.185	260666.667
5084.0	6989.182	7325.359	21958.224	9219.961	9156.241	7416.412
11071.0	7848.2	6051.358	79629.011	9219.961	9156.241	7416.412
7005.0	6798.25	6931.803	36464.558	9219.961	9156.241	5206.75
420000.0	402875.0	552476.471	414724.23	287000.0	556565.485	485000.0
6098.0	8461.833	8301.598	4494.346	9219.961	9156.241	7301.6
1265000.0	78259.0	200250.0	574080.596	251000.0	340518.81	21500.0
216000.0	386625.0	134178.663	209540.182	269000.0	238148.471	171875.0
102000.0	138000.0	178375.252	91245.946	80333.333	93219.784	171875.0
165000.0	219000.0	200250.0	184041.18	181666.667	152282.109	260666.667
120000.0	161250.0	242464.303	121247.032	93000.0	179339.534	171875.0
128000.0	186250.0	185149.858	100568.453	125500.0	178954.015	171875.0
87000.0	136375.0	200250.0	104749.272	125500.0	169073.455	122000.0
226000.0	354727.273	8901.584	237064.198	398000.0	285326.784	171875.0
9087.0	7874.0	326922.285	70586.878	9219.961	9156.241	7416.412
558000.0	526875.0	4591.719	553476.025	261923.077	615600.568	485000.0

Table 5.2. Real and predicted execution time provided by each algorithm in ms.

real	ibk	kstar	MLP	Random Tree	Random Forest	Decision Table
558000.0	526875.0	4591.719	553476.025	261923.077	615600.568	485000.0
4081.0	32641.7	394097.737	382.961	9219.961	9156.241	5206.75
232000.0	299750.0	115031.021	261415.609	172000.0	291725.638	431500.0
187000.0	327222.222	838123.608	184584.201	188000.0	187118.162	171875.0
452000.0	324444.444	117863.018	349977.371	287000.0	768749.605	459000.0
820000.0	130454.545	7135.614	119773.716	93000.0	162286.381	122000.0
10004.0	7242.4	6565.024	34324.761	9219.961	9156.241	7301.6
10032.0	7025.556	696509.475	37835.324	9219.961	9156.241	7301.6
583000.0	354578.875	7088.297	641191.693	457000.0	422137.543	1019666.667
5057.0	6393.333	270451.886	21811.579	9219.961	9156.241	7416.412
166000.0	222250.0	256153.142	186192.944	125500.0	229514.323	260666.667
207000.0	334125.0	1162264.931	211557.437	188000.0	219830.456	171875.0
1231000.0	796625.0	710197.599	1112757.047	1513333.333	1037107.367	1513333.333
450000.0	647750.0	5723.513	785094.725	854000.0	706300.02	689000.0
6081.0	6293.0	28000.0	101733.534	9219.961	9156.241	5206.75
42000.0	38017.625	305327.835	12277.596	9219.961	37418.048	89333.333
218000.0	365500.0	6187.591	186359.589	143000.0	255062.739	431500.0
6063.0	6815.222	6010000.0	30242.712	9219.961	9156.241	5206.75
3086000.0	984759.625	316037.381	5704678.407	6010000.0	4068870.819	284832.135
222000.0	334125.0	736916.991	203082.086	188000.0	259565.142	171875.0
744000.0	533763.1	16954.277	757703.123	747000.0	597616.308	747000.0
17000.0	44778.0	218795.967	35557.276	9219.961	12781.017	89333.333
379000.0	503636.364	6258.262	354190.042	269000.0	227242.232	260666.667
3084.0	15529.375	620842.135	31057.952	9219.961	9156.241	7416.412
431000.0	547666.667	272253.329	428370.841	261923.077	631381.64	431500.0
250000.0	307250.0	734486.118	270393.186	157000.0	284083.004	485000.0
365000.0	370375.0	581742.878	409162.192	459000.0	541612.001	485000.0
417000.0	543875.0	7379.349	393426.324	342000.0	567254.433	431500.0
10072.0	8294.25	148193.696	15629.52	9219.961	9156.241	7301.6
89000.0	210625.0	259991.194	94292.603	107500.0	123622.906	171875.0
464000.0	590100.0	350499.999	408148.414	269000.0	360065.053	260666.667
110000.0	253000.0	464051.623	109597.152	52000.0	230087.744	284832.135
257000.0	238500.0	5203.661	211583.785	143000.0	274718.536	431500.0
4056.0	8543.5	217613.516	11578.726	9219.961	9156.241	5206.75
311000.0	326555.556	7141.922	326074.286	261923.077	500427.096	205200.0
7032.0	5314.125	5838.888	6342.14	9219.961	9156.241	7301.6
10031.0	7550.625	7083.522	75779.226	9219.961	9156.241	7416.412
11010.0	7942.4	197652.028	35856.746	9219.961	9156.241	7301.6
204000.0	259666.667	7473.568	200780.748	398000.0	277847.624	171875.0
5076.0	6423.5	414342.312	20142.948	9219.961	9156.241	7416.412
417000.0	659555.556	212178.972	372293.776	342000.0	519761.245	431500.0
213000.0	300625.0	438627.626	190798.082	209000.0	195671.283	7301.6
483000.0	339078.875	117863.018	366580.816	350500.0	306132.303	7416.412
153000.0	145500.0	411960.871	225625.058	398000.0	272100.035	122000.0
231000.0	371828.875	6704.366	254631.582	172000.0	281918.448	431500.0
11026.0	8590.667	5931.838	70746.196	9219.961	9156.241	7301.6
6010.0	6784.25	456875.675	86211.25	9219.961	9156.241	5206.75
248000.0	300666.667	1306746.704	334410.908	611000.0	448904.245	310666.667
681000.0	486300.0	350499.999	713278.711	1502000.0	807028.692	1019666.667
216000.0	253000.0	6100.737	109597.152	52000.0	230087.744	284832.135
6029.0	6427.5	189511.352	41050.254	9219.961	9156.241	5206.75
158000.0	210625.0	8413.388	120637.245	107500.0	165918.317	171875.0
3077.0	15529.375	218023.172	22233.077	9219.961	9156.241	7416.412
183000.0	406625.0	8100.273	374233.98	411000.0	285286.108	260666.667
11025.0	8055.0	693298.587	29266.874	9219.961	9156.241	7416.412
427000.0	321666.667	694101.993	528921.489	611000.0	629601.328	459000.0
453000.0	389828.875	8360.673	465918.252	457000.0	403433.269	689000.0
7030.0	8821.0	456875.675	13085.634	9219.961	9156.241	7301.6
499000.0	391250.0	538372.694	498756.005	1025000.0	657873.115	310666.667
556000.0	680125.0	7772.383	570652.826	431500.0	520660.164	485000.0
10087.0	8601.667	217458.904	55632.698	9219.961	9156.241	7301.6
292000.0	338000.0	326667.987	246932.129	342000.0	407658.417	205200.0
553000.0	367625.0	166872.979	457657.915	284000.0	317172.456	260666.667
358000.0	471555.556	650874.514	375265.065	411000.0	265974.81	260666.667
970000.0	781125.0	174041.934	1078894.219	790000.0	653640.487	1019666.667
103000.0	129111.111	258969.066	98890.305	125500.0	182967.49	122000.0
476000.0	353444.444	9268.437	398327.788	241000.0	306773.597	260666.667
10098.0	8939.625	18671.003	68285.192	9219.961	9156.241	7416.412
1616000.0	804203.875	308380.847	65316.709	1631.0	296954.233	1631.0
197000.0	319875.0	377565.897	176261.212	157000.0	278418.125	431500.0
226000.0	339882.0	972960.185	200827.93	125500.0	173210.754	70000.0
1361000.0	896400.0	313499.015	1487731.861	1502000.0	1127777.283	1019666.667
530000.0	468363.636	55000.0	476901.888	269000.0	387992.072	260666.667
103000.0	192250.0	6144.632	269676.911	350500.0	138816.474	284832.135
4060.0	6673.75	118090.075	96186.082	9219.961	9156.241	5206.75
111000.0	138000.0	111307.885	115558.978	80333.333	113944.094	171875.0
140000.0	145500.0	251000.0	209407.465	398000.0	265877.751	122000.0
210000.0	81387.125	87174.03	236729.961	209000.0	252558.486	52000.0
74000.0	138125.0	1262357.866	106885.118	67000.0	88928.536	122000.0
2039000.0	1044100.0	90600.182	2056828.3	1513333.333	1225761.048	1513333.333
67000.0	130000.0	215329.569	97229.111	107500.0	113160.251	122000.0
232000.0	244200.0	8232.924	229466.946	284000.0	262526.262	122000.0
9091.0	8547.0	351165.053	59010.265	9219.961	9156.241	7416.412
281000.0	406875.0	5631.745	271656.008	342000.0	455339.295	205200.0
7010.0	6650.25	439289.432	50161.226	9219.961	9156.241	5206.75
199000.0	289000.0	283712.199	201724.723	157000.0	326232.956	431500.0
155000.0	347875.0	239773.355	210048.798	159000.0	237164.631	260666.667
151000.0	284875.0	6006.63	167871.564	133500.0	234939.843	122000.0
6098.0	5832.556	125649.502	3306.982	9219.961	9156.241	7301.6
3310000.0	55896.75	251000.0	132375.575	251000.0	173244.257	21500.0
429000.0	242500.0	100982.881	312585.104	444000.0	307599.584	284832.135
127000.0	261111.111	5826.159	158546.832	133500.0	166683.617	122000.0
7006.0	5430.5	896985.295	13418.485	9219.961	9156.241	7301.6
791000.0	679750.0	283712.199	783483.076	518000.0	850791.594	485000.0
448000.0	317111.111	117863.018	414715.807	398000.0	325504.611	260666.667
150000.0	279375.0	332493.892	185480.856	203000.0	198068.11	122000.0
523000.0	726250.0	157676.901	520118.865	342000.0	496454.075	485000.0
2582000.0	99893.0	275395.466	220060.676	125500.0	358539.267	89333.333

Table 5.3. Real and predicted execution time provided by each algorithm in ms.

Chapter 6

Conclusions and Future Work

Conclusion

In this work a study of a specific and particular distributed application has been conducted. The purpose of the work is to take the most time and resources consuming part of a more complex IT system dedicated to the insurance field in a cloud environment. In particular the idea is to fully exploit the formula *pay as you go* trying to minimize as much as possible the costs. Since the context we are dealing with might in some cases require some urgency, or in any case a bounded time to produce some results, the idea is to be able to estimate the IT infrastructure necessary to achieve that goal in the desired time with the minimal expense. The porting of this portion of the system in the cloud, may represent a significant savings for the company, which may choose to vary the processing time according to need by finding the right compromise between time and costs. Since these procedures do not require daily use, this choice might prevent the company purchasing very high-performance computing infrastructure together with their maintenance. First we have given a general overview of the laws that require this kind of computation, its aim, and the context in which the problem arises to contextualize it. We have then analyzed the performance of the system, underlining its behavior when it is scaled up/down on several cloud machines and giving a trend of its scaling factor and speed-up. The computation depends on so many aleatory variables and conditions such that it is hard to establish a mathematical model that describe the algorithm complexity. The observed behavior is almost linear varying the number of natural iterations of the Monte Carlo simulation, thus a naive approach could be a trial of the elaboration on a very small number of iterations and then multiply the result to obtain an estimation on a bigger number of iteration. However this approach requires to a priori turn on

the machines to carry out the test, disregarding the fact that this approach may not capture what would be adding machines to the cluster. To deal with the situation in a more efficient way we have tried to exploit some machine learning algorithms that training themselves on all previous successfully completed elaborations till the current one, they try to grasp as much as possible dependencies and patterns that affect the execution time, and give a prediction on that one that must be elaborated.

Future work

Within the entire work, we based all our reasoning using as reference unit the individual elementary elaborations (eeb). This is to analyze more closely the problem, since each eeb has its own characteristics that has said in section 4.1 could be completely different from the others belonging to the same portfolio. On the other hand an eeb is never treat independently from the other, considering even that it would be a waste purchasing an instance for a single eeb. Therefore it would make sense to reason on an total complexity considering someway an aggregation of all the predictions regarding the entire portfolio. Due to this substantial difference among the elaborations, the cluster could be result underestimated for some and less suitable for others. Taking into account this, a further optimization might consist in partitioning the grid, in order to launch concurrently the less complex elaboration on a small partition of the cluster and dedicate an higher number of cores to the more complex ones. The implementation of this strategy would require more effort as well as greater granularity approach. In view of a service that can be convenient, competitive and optimal, we consider the possibility to integrate the software with a multi cloud support. That means that the system should be able not only to find the best configuration, but even the best configuration on the most convenient provider at that moment, considering that always more frequently the providers offer the possibility of instances auctions, as it is already provided by AWS with the bidding service on the **spot instances**. [1]

[1] Amazon EC2 Spot instances are spare EC2 instances that you can bid on to run your cloud computing applications. Since Spot instances are often available at a lower price, you can significantly reduce the cost of running your applications, grow your application's compute capacity and throughput for the same budget, and enable new types of cloud computing applications.

Bibliography

[1] Alef srl. Available from: http://www.alef.it/.

[2] Amazon web sevices. Available from: https://aws.amazon.com/it/.

[3] Ansible. Available from: http://www.ansible.com/home.

[4] Goolgle compute engine. Available from: https://cloud.google.com/compute/.

[5] Ibm cloud manager with openstack. Available from: http://www.ibm.com/developerworks/servicemanagement/cvm/sce.

[6] Ibm smart cloud. Available from: http://www.ibm.com/cloud-computing/us/en/.

[7] Microsoft azure. Available from: http://azure.microsoft.com/it-it/.

[8] Nimbus. Available from: http://cloud.verizon.com/.

[9] Nimbus. Available from: http://www.nimbusproject.org/.

[10] Open nebula. Available from: http://opennebula.org/.

[11] Open stack. Available from: http://www.openstack.org/.

[12] Rackspace cloud. Available from: http://www.rackspace.com/cloud.

[13] Waikato university. Available from: http://www.waikato.ac.nz/.

[14] Directive 2009/138/EC of the European Parliament and of the Council of 25 november 2009 on the taking-up and pursuit of business of Insurance and Reinsurance (Solvency II). *Official Journal of the European Union*, **L351/1** (17.12.2009).

[15] ANGELIS, P. L. D., PERLA, F., AND ZANETTI, P. Hybrid mpi/openmp application on multicore architectures: The case of profit-sharing life insurance policies valuation. *Applied Mathematical Sciences*, **7** (2013), 051 .

[16] BAUER, D., REUSS, A., AND SINGER, D. On the calculation of the solvency ii capital requirement based on nested simulations. *Astin Bulletin*, **42** (2012), 453.

[17] BREIMAN, L. Out-of-bag estimation. (1996).

[18] BREIMAN, L. Random forests–random features. (1999).

[19] CASARANO, G., CASTELLANI, G., PASSALACQUA, L., PERLA, F., AND ZANETTI, P. Relevant applications of monte carlo simulation in solvency ii. *Soft Computing*, (2015).

[20] DE FELICE, M. AND MORICONI, F. *Una nuova finanza d'impresa. Le imprese di assicurazione, Solvency II, le Autorità di vigilanza*. Itinerari. Il Mulino (2011).

[21] EVOY, G. M. AND SCHULZE, B. Understanding scheduling implications for scientific applications in clouds. MGC (2011).

[22] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: An update. *SIGKDD Explorations*, **11** (2009).

[23] HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P., AND WITTEN, I. H. The weka data mining software: An update. *SIGKDD Explorations*, **Volume 11** (2009).

[24] HAROMOTO, H., MATSUMOTO, M., NISHIMURA, T., PANNETON, F., AND P., L. Efficient jump ahead for $f_2 - linear$ random number generators. *INFORMS Journal on Computing*, **20** (2008), 385.

[25] HAYKIN, S. AND NETWORK, N. A comprehensive foundation. *Neural Networks*, **2** (2004).

[26] IAS. *A Global Framework for Insurer Solvency Assessment* (2004). Research Report of the Insurer Solvency Assessment Working Party.

[27] MIT. Starcluster. Available from: `http://star.mit.edu/cluster/docs/latest/overview.html`.

[28] POLITO, S. G. AND ADRIANO COSTANZO. Demo paper: Automatic provisioning, deploy and monitoring of virtual machines based on security sevice level agreement in the cloud. *IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, **14** (2014), 536.

[29] RAINER SCHMIDT, G. F., HEIKE WEISS. *Advances in Data Mining. Applications and Theoretical Aspects.* (2012).

[30] W.R., L., ET AL. *Professional Actuarial Speciality Guide - Asset-Liability Management*. Society of Actuaries (2002).

[31] YAO, J. On-demand optimal cloud service provisioning composition across multi-cloud. *International Conference of Computational and Information Sciences*, (2013), 1566.

Ringraziamenti

This page has been written in Italian, so that all the people to whom it is addressed can read it easily ...
Questa pagina é stata scritta in italiano, in modo tale che tutti coloro alla quale é dedicata possano leggerla facilmente...

Il primo grande grazie va ai miei genitori, che mi hanno dato la possibilitá materiale ma ancor prima l'appoggio morale per arrivare fino a qui, oltre al grande dono della vita. Ai miei fratelli Francesco e Stefano, anche lui ha dato il suo grande contributo rinunciando a una cosa per lui molto importante, la mia presenza.
Un grandissimo ringraziamento va al dott. Giuseppe Casarano, che mi ha dato la grande opportunitá non solo di svolgere questo lavoro di tesi in un ambito piú che interessante, ma soprattutto di lavorare in un ambiente eccezionale, formato da persone altrettanto speciali che mi hanno fatto sentire fin dal primo giorno parte del gruppo, senza mai farmi sentire l'ultimo arrivato. Grazie a Enzo, Gabriele, Giuseppe C., Giuseppe F., Luana, Leonardo, Paolo, Pietro, Stephane, Vincenzo, grazie al prof. Passalacqua, al prof. Castellani e al prof. De Felice. Grazie anche a tutti gli altri componenti di Alef, ognuno di loro ha contribuita alla buona riuscita di questo lavoro, anche se solo con un sorriso.
Grazie ai ragazzi del coffee group, con i quali abbiamo condiviso questi anni di universitá e i discorsi assurdi e spesso metafisici che hanno caratterizzato i nostri pranzi, a Daniele S., Romolo, Francesca, Daniele U., Ion, Nick, Luigi, Eleonora, Davide, Giuseppe e Alessio, con alcuni di loro abbiamo affrontato scogli insormontabili, ma che assieme si sono rivelati molto meno spigolosi e alti di quel che sembravano.
Grazie al mio prof. Bruno Ciciani, esempio di serietá, disponibilitá ma sopra ogni cosa di umiltá, virtú spesso difficile da trovare in persone che ricoprono il suo ruolo.
Grazie all'Ing. Alessandro Pellegrini, una presenza costante che ha sempre creduto nelle mie capacitá anche quando io stesso non ci credevo, e che ha sempre saputo darmi lo stimolo per progredire oltre che la sua estrema disponibilitá.
In ultimo solo per ordine di arrivo nella mia vita, ma non per importanza grazie a Susanna, che mi ha conosciuto nel momento in cui non potevo dedicarle il tempo che avrei voluto, ha sempre ascoltato le mie ansie, accolto le mie difficoltá, sopportato i miei momenti no, eppure ha deciso di restare.

Grazie a tutti coloro che ci sono o ci sono stati, perché una piccola parte di questo successo é anche opera vostra.

Andrea