Submitted to
Journal of Parallel and Distributed Computing
"Special Issue on Parallel/Distributed Computing and Optimization"

Configurable Cost-Quality Optimization of Cloud-based VoIP

Andrei Tchernykh [a,e,*], Jorge M. Cortés-Mendoza [a], Igor Bychkov [b], Alexander Feoktistov [b], Loic Didelot [c], Pascal Bouvry [d], Gleb Radchenko [e], Kirill Borodulin [e]

[a] *CICESE Research Center, Carretera Ensenada-Tijuana 3918, 22860, Ensenada, Baja California, Mexico. chernykh@cicese.mx, jcortes@cicese.edu.mx*
[b] *Matrosov Institute for System Dynamics and Control Theory of SB RAS, 134 Lermontov St., Irkutsk, 664033, Russia. bychkov@icc.ru, agf65@yandex.ru*
[c] *MIXvoip S.A., 70 Rue des Prés, 7333, Steinsel, Luxembourg. ldidelot@mixvoip.com*
[d] *University of Luxembourg, Maison du Nombre 6, Avenue de la Fonte, L-4364 Esch-sur-Alzette, Luxembourg. Pascal.Bouvry@uni.lu*
[e] *South Ural State University, Chelyabinsk, 76 Lenina St., Chelyabinsk, 454080, Russia. gleb.radchenko@susu.ru, borodulinkv@susu.ru*

Abstract

In this paper, we formulate configurable cloud-based VoIP call allocation problem as a special case of dynamic multi-objective bin-packing. We consider voice quality influenced by CPU stress, cost contributed by the number of billing hours for Virtual Machines (VMs) provisioning, and calls placed on hold due to under-provisioning resources. We distinguish call allocation strategies by the type and amount of information used for allocation: knowledge-free, utilization-aware, rental-aware, and load-aware. We propose and study a variety of strategies with static and dynamic policies of VM provisioning. To study realistic scenarios, we consider startup delays for VM provisioning, and three configurable parameters: utilization threshold, rental threshold, and prediction interval. They can be configured and dynamically adapted to cope with different objective preferences, workloads, and cloud properties. We conduct comprehensive simulation on the real workload of the MIXvoip company and show that the proposed strategies outperform ones currently in-use.

Keywords: Call allocation; Scheduling; Cloud computing; Cloud Voice over IP; Quality of Service; Bin packing

*Corresponding author
Email address: chernykh@cicese.mx (A. Tchernykh)
URL: http://usuario.cicese.mx/~chernykh/
Phone +52 (646) 175-0595 ext. 23426
FAX +52 (646) 175-0593, 175-0549,2

1. Introduction

Cloud computing is growing as feasible business model adopted as an innovative and profitable solution (AWS, 2017) [1], (GCP, 2017) [2]. Cost savings and scalability are the most important reasons due to business is moving to the cloud. Both factors allow increasing own profit and provide an adequate quality of services to the users. Some examples of business migration are: TV (AWS, 2017) [1], radio (GCP, 2017) [2], telephone (MIXvoip, 2017) [3], etc.

Voice over IP (VoIP) is a fast growing technology in cloud computing considered as a long-term service roadmap, offering more features than traditional telephony (PSTN) infrastructure. The main benefits of this technology, over conventional telephones and VoIP, are cost-effectiveness, higher flexibility, scalability, extensive service variety, etc. It can cope with different workloads, objective preferences, and cloud properties.

Asterisk (Madsen et al., 2011) [4] is the backbone of Cloud VoIP (CVoIP) solution. This software powers IP Private Branch Exchange (PBX) systems. Virtual Machines (VMs) execute Asterisk instances and provide calls, voice mails, video/audio conferences, interactive phone menus, call distribution, etc. Additionally, users can transfer images and texts, and they can create new functionalities, opening up an entirely new experience in telephonic communication.

The success of the business depends significantly on the price and Quality of Service (QoS) factors. CVoIP providers always look to offer an adequate service considering various parameters: the quality of voice, transit time of packets across the Internet, queuing delays at the routers, signaling overhead, end-to-end delay, jitter, call set-up and tear-down time, codec compression technique, processing capability, etc. (Singh et al., 2014) [5].

In our previous works (Cortés-Mendoza et al., 2015) [6], (Cortés-Mendoza et al., 2016a) [7], (Cortés-Mendoza et al., 2016b) [8], we formulated the problem of scheduling of VoIP services in cloud environments, and proposed a new model for bi-objective optimization. We considered the particular case of the on-line non-clairvoyant dynamic bin packing problem and discussed solutions for provider cost, quality of service, and robustness optimization.

We proposed and evaluated several call allocation strategies that use information about VMs utilization, VM provisioning time, and VM startup time delays. These factors impact on resource under- or over-provisioning, call quality, and cost (Cortés-Mendoza et al., 2015) [6].

In our most recent study (Cortés-Mendoza et al., 2017) [9], we analyzed strategies focusing on estimation of the amount of VMs needed to provide VoIP services without quality degradation and on techniques to reduce the number of active VMs.

In this paper, we analyze multi-objective scheduling strategies with billing hours, calls placed to hold, and voice quality criteria. We classify call allocation strategies by the type and amount of information used for allocation: (1) Knowledge-Free (KF), with no information about calls and needed resources; (2) Utilization-Aware (UA), with VM utilization information; (3) Rental-Aware (RA) with known VM start time; and (4) Load-Aware (LA) with VM load information.

We introduce three configurable parameters: Utilization Threshold (UT), Rental Threshold (RT), and Prediction Interval (PI) to cope with different objective preferences, workloads, and cloud properties. We propose and study thirty-two on-line non-clairvoyant scheduling strategies with static policy of VM provisioning and ninety-six allocation strategies with dynamic load prediction and dynamic criteria for VM provisioning.

We conduct comprehensive simulation analysis on the real workload of the MIXvoip company [3] and show that our strategies improve the performance of known ones. They increase voice quality, reduce VoIP provider cost, and amount of calls placed on hold.

The paper is structured as follows. The next section presents VoIP service considering underlined infrastructure and software. Section 3 briefly reviews related works on bin packing, call load balancing, and load estimation. Section 4 provides the problem definition and proposed model. Section 5 describes VoIP call allocation strategies. Section 6 presents three prediction algorithms used to estimate new VM startup time. Section 7 shows the configuration and training setup for neural network predictor. Section 8 discusses our experimental setup, workload and studied scenario. Section 9 presents experimental analysis. Section 10 highlights the conclusions of the paper and future work.

2. Internet telephone

The Internet telephony VoIP refers to making calls according to the IP standard. It achieves significant call rate reduction decreasing the infrastructure and communication costs.

VoIP providers have to deal with several problems of traditional VoIP systems. For instance, availability of the service for any number of users is fundamental. With the increasing number of clients, providers need to invest in a large infrastructure to avoid loss of calls (hence, users). This situation leads to two major problems: over-provisioning and over-running cost. Even with a high number of resources, the system cannot be able to deliver services during peak hours or abnormal system behavior.

A cloud-based VoIP reduces the costs and increases availability without fall into over-provisioning. The deployment of the virtual infrastructure is easy to implement, faster to provision, and integrates services that are dynamically scalable. Further, it adds new features and capabilities for users (data transfer availability, integrity, and security).

Voice Nodes (VNs) are the core part of the CVoIP telephony system (Figure 1) (Cortés-Mendoza et al., 2016a) [7]. They execute specialized software to emulate a telephone exchange, gateways, interconnection switches, session controllers, firewall, etc. VNs verify the credentials of the user and try to reach end-devices. After the call is finished, call details are stored in the Call-Detail-Record (e.g., client id, destination, prefix, duration call).

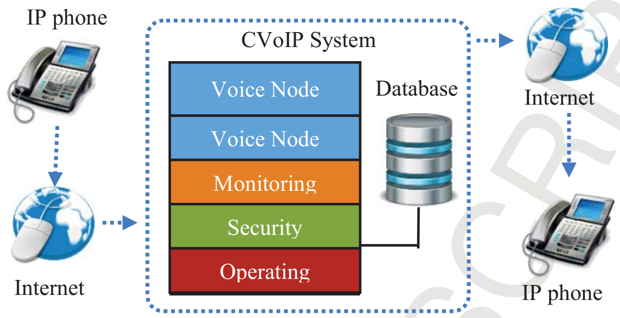

Figure 1: VoIP architecture.

2.1 Infrastructure

MIXvoip [3], a company that hosts and delivers VoIP services, developed the concept of Super-Node (SN) and Super Nodes Cluster (SNC) to enrich features for telephone exchanges. SN model combines cloud service with smart business telephony, VoIP, and other telephony services. It brings redundancy in communication in a given geographical area to ensure a high voice quality between SNs through the public Internet and provides short paths between two local users.

The most known telephone software for processing calls and providing a powerful control over call activity is Asterisk (Madsen et al., 2011) [4]. It is a framework, under a free license, for building multi-protocol, real-time communication solutions providing a powerful control over call activity. It processes calls and connects to other telephone services, such as the Public Switched Telephone Network (PSTN) and VoIP services.

The VoIP system consists of multiple VNs that run and handle calls. Each node has Asterisk running process with a unique IP address that is used by end users to connect inside and outside the network.

2.2 Quality of service

The VoIP services have stricter constraints and sensitive factors. Call processing and call delivery are two key issues which determine the quality of calls. Call processing focuses on the time to set-up and tear-down the call, and on converting the voice portion of the calls into packets transported over the network. Adequate quality of voice is the most important aspect of call processing.

The quality of voice is subjectively perceived by the listener. A standard benchmark used to determine the quality of voice is the Mean Opinion Score (MOS). It evaluates the quality of speech provided by a codec. Each codec offers a certain quality of speech only if processor utilization is low enough. Theoretically, processor utilization of 100% provides the best-expected performance. However, Eleftheriou (2017)[10] showed that CPU cannot handle the stress when utilization is up to 85%, then jitters and broken audio symptoms appear.

Additionally, the author did not report any influence of memory on the voice quality reduction.

During a call processing, the codec increases the used bandwidth, but it is less significant than the same codec adds to CPU utilization.

Montazerolghaem et al. (2015) [11] reported that the consumed bandwidth of 6,500 calls per second do not exceed 100 Mbps, and 10,000 calls do not reach 400 Mbps.

Table 1 shows the bandwidth used by different codecs considering that VoIP calls use audio streams for endpoints (a call between two parties will use double of bandwidth). The number of calls supported with 100 Mbps connection is between 6,000 and 25,000 depending on codec of the calls.

Table 1. Codec bandwidth.

Codec	Bit Rate (kbps)	MOS	Bandwidth (kbps)
G.711	64	4.1	87.2
G.729	8	3.92	31.2
G.723.1	6.3	3.9	21.9
G.723.1	5.3	3.8	20.8
G.726	32	3.85	55.2
G.726	24	-	47.2
G.728	16	3.61	31.5
G722_64k	64	4.13	87.2
ilbc_mode_20	15.2	NA	38.4
ilbc_mode_30	13.33	NA	28.8

However, the amount of calls handled by CPU is less than supported by 100 Mbps connection. Hence, the call processing is the key feature to guarantee the QoS. In (Cortés-Mendoza et al., 2015) [6], the authors proposed to limit processor utilization in order to ensure QoS.

2.3 CPU utilization

Calls have a different impact on the processor utilization (Cortés-Mendoza et al., 2015) [6] depending on the operations performed by Asterisk. If transcoding operations are performed, the utilization is higher than when transcoding is not used. In the latter case, Asterisk is in charge of only routing the call. However, depending on the codec, the processor load is influenced as well.

Table 2 shows processor utilization for call without transcoding presented by Montoro et al. (2009) [12].

Table 2. CPU utilization for calls without transcoding.

Protocol	Codec	10 Calls (%)	1 Call (%)
SIP/RTP	G.711	2.36	0.236
SIP/RTP	G.726	2.13	0.213
SIP/RTP	GSM	2.58	0.258
SIP/RTP	LPC10	1.92	0.192

The performance of ATOM processors is analyzed for VoIP considering calls number, utilization, power consumption, database messaging, registration and call performance (Eleftheriou, 2017) [10]. The author concludes that CPU can process from 70 to 500 calls with 100% of utilization.

2.4 VoIP provider optimization criteria

Inefficient resource utilization directly leads to higher costs. VoIP providers should use the resources efficiently to offer competitive prices to customers. Virtualization technologies allow creating VoIP virtual servers hosted in clouds and provisioned (leased) on a subscription basis to any scale.

In a typical cloud scenario, a VoIP provider can select different resources that are available on demand from cloud providers. They have certain service guarantees distinguished by the amount of computing power received within a requested time, and a cost per unit of execution time. In this paper, three criteria are considered: the billing hours for VMs to provide a service, number of calls on hold, and voice quality reduction.

3. Related work

The migration of VoIP businesses to clouds triggers intensive research on several fields: call allocation, load balancing, quality of service, load prediction, load estimation, etc. The next section highlights recent works on load balancing, bin-packing, and load estimation related to VoIP management.

3.1 Call load balancing

The main objective of call load balancing is to reduce the infrastructure cost and guarantee that service will be delivered in the best possible way. Several algorithms have been proposed to improve the performance of CVoIP system.

The Virtual Load-Balanced Call Admission Controller - VLB-CAC (Montazerolghaem et al., 2016) [13] is a strategy to balance calls and provide admission control for SIP servers. It has mechanisms to predict the call number, required resources, and select the most appropriate VM instances considering CPU, memory, and bandwidth. The authors propose a model to maximize the resource usage and system throughput.

Mobicents SIP Load Balancer (Mobicents, 2017) [14] is a SIP-based proxy for VoIP infrastructure with multiple ingress proxy servers. It allows avoiding congestion, bad resource utilization, and overload. Mobicents uses Round Robin algorithm to distribute the traffic between servers, and it contemplates requests and system parameters, like CPU.

3.2 Bin packing

Bin packing heuristics are used widely in cloud management. Their primary function is to allocate applications into the VMs and/or VMs on the resources.

Li et al. (2016) [15] consider a variant of dynamic bin packing problem to solve the request dispatching problem arising from cloud gaming. In cloud gaming systems, computer games are managed on cloud servers, where each game instance demands a certain amount of resources. To provide a good user experience, requests must be dispatched with enough resources of CPU and GPU. The main objective is to minimize the number of servers (bins) to process the gaming requests (items) due to each resource adds a proportional cost to the duration of its usage. The authors analyze the competition ratio for original and modified versions of Best Fit and First Fit algorithms.

Song et al. (2014) [16] propose a relaxed on-line bin packing algorithm called Variable Item Size Bin Packing (VIS-BP). It allocates data center resources using live VM migration. The main goal is to restrict the combinations of tasks in a bin to minimize the amount of wasted space. The authors evaluate the effectiveness of the algorithm and provide a theoretical proof for the number of used servers and VM migrations. The multi-dimensional version considers a mix of CPU and network.

Wolke et al. (2015) [17] analyze a wide variety of controllers for VMs placement and reallocation using the resources availability (CPU, memory, etc.). Incoming VMs are placed on servers as long as their residual capacity allows. Bin packing algorithms are used at this stage. The reallocation controller triggers VM migration when underloaded servers are emptied, and overloaded ones are relieved. The authors found that combinations of placement controllers and periodic reallocations achieve the highest energy efficiency subject to predefined service levels.

3.3 Load estimation

Some researchers are being conducted with the motivation of load prediction for an efficient distribution of the load in the system. The main idea is to anticipate the incoming traffic (calls for VoIP problem) to minimize the infrastructure costs and improve the QoS to the end user.

Simionovici et al. (2013) [18] study and compare two prediction models for real VoIP environment. Interactive Particle System (IPS) and Neural Network (NN) are used to predict the incoming voice traffic during time frames. In (Simionovici et al., 2015) [19], the authors extend previous work and use IPS, Gaussian Mixture Model (GMM), and Gaussian Process (GP) to estimate the incoming voice traffic. The authors provide flexible modeling approaches, traffic shaping determined by clients, and scalable solutions with good prediction precision. All algorithms were trained and tested under different scenarios.

Lu et al. (2016) [20] investigate the dynamic characteristics of cloud workloads and analyze three algorithms as workload predictors to improve the energy efficiency of cloud systems. The authors evaluate the performance of Rand Variable Learning Rate Backpropagation Neural Network (RVLBPNN), Hidden Markov Modelling (HMM), and Naive Bayes Classifier (NBC) for predicting of future workloads in cloud datacenters. All models are evaluated for their efficiencies in predicting memory and CPU intensive workloads. The main goal is to maintain a high level of Quality of Service (QoS) and reduce the usage of the data center resources.

Rate of Change – RoC (Campos and Scherson, 2000) [21] is a strategy for load balancing tasks in the distributed system. It allows to trigger a dynamic, distributed, and implicit load balancing mechanism. The balancer (*Bal*) makes job distribution decisions at run-time, locally and asynchronously. Each *Bal* considers its load; migration does not depend on the load of other Bals. The migration decision depends on current load, load changes in the time interval (rate of load change), and current load balancing parameters. The difference in the load is used as an estimation of load, its prediction for the next time slot, and detection of the need for load balancing process.

4. Model

The model follows our previous works (Cortés-Mendoza et al., 2015) [6], (Cortés-Mendoza et al., 2016a) [7], (Cortés-Mendoza et al., 2016b) [8], where cloud VoIP infrastructure consists of m heterogeneous Super Node Clusters $SNC_1, SNC_2, \dots, SNC_m$ with relative speeds $s_1, s_2, \dots, s_m$. Each SNC_i, for all $i = 1, \dots, m$, consists of m_i SNs. Each SN_k^i, for all $k = 1, \dots, m_i$, runs $k_i(t)$ VMs at time t. We assume that VMs of one SN are identical and have the same processing capacity. The virtual machine VM_j is described by a tuple $\{st_j, size_j, StUp_j\}$ that consists of its start of rental period $st_j \geq 0$, startup delay $StUp_j$, the processing capacity $size_j$ in MIPS.

The SNC contains a set of routers and switches that transport traffic between the SNs and to the outside world. A switch connects a redistribution point or computational nodes. The connections of the processors are static but their utilization is changed. The SNC interconnection network is local. The interconnection between $SNCs$ is provided through public Internet.

We consider n independent calls $J_1, J_2, \dots, J_n$ that must be scheduled on a set of SNCs. The call J_j is described by a tuple $\{r_j, p_j, u_j\}$ that consists of its release time $r_j \geq 0$, duration p_j (lifespan), and contribution to the processor utilization u_j due to the used codec. The release time of a call is not available before the call is submitted, and its duration is unknown until the call has been completed. The utilization is a constant for a given call that depends on the used codec and VM processing capacity.

We define the provider cost model by considering a function that depends on the number of provisioned VMs and their rental time. We denote the number of billing hours in SNC_i by:

$$\bar{b}_i = \sum_{r=0}^{C_{max}/rp} k_i(r * rp), \tag{1}$$

where rp defines the minimum rental period of a VM, typically $rp = 60$ min. Total billing hours in all $SNCs$ is defined by:

$$\bar{b} = \sum_{i=1}^{m} \bar{b}_i \tag{2}$$

In addition to $st_j, StUp_j, size_j$, the VM is characterized by $vmu_j(t)$ the utilization (load) of the VM_j at time t. We introduce a Quality Reduction as a function of the VMs utilization by:

$$\bar{q}_i = \sum_{j=1}^{\bar{b}_i} \sum_{t=st_j}^{ft_j} \gamma(vmu_j(t)), \tag{3}$$

where st_j and ft_j define the rental time of the VM_j and the penalization function. $\gamma(\alpha)$ models the behavior of the quality reduction as a function of the VM utilization, see Figure 2. $\gamma(\alpha) = ((\alpha - 0.7) * \overline{3.33})^2$, when α>0.7, and 0, otherwise. Total Quality Reduction is defined by:

$$\bar{q} = \sum_{i=1}^{m} \bar{q}_i \tag{4}$$

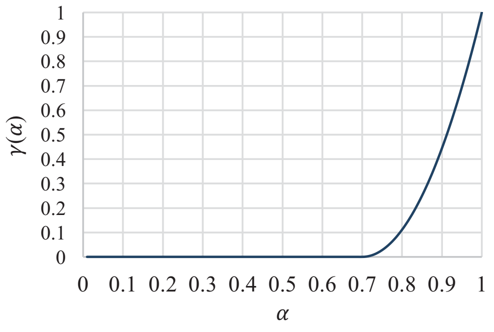

Figure 2: Voice quality reduction versus processor load (utilization).

Moreover, we analyze the number of calls on hold. It occurs when the VMs cannot process the arriving calls, so the calls wait in the queue for an available VM. The amount of calls placed on hold is defined by:

$$\bar{c} = \sum_{j=1}^{n} \delta(s_j - r_j), \tag{5}$$

where s_j is the initiation time of J_j, and $\delta(\beta)$ is 1, if $\beta > 0$ and 0, otherwise.

We consider this problem as a special case of dynamic bin packing (on-line and non-clairvoyant). Bins represent VMs, and the items height define the call contribution to the VM utilization.

To compare strategies, we perform an analysis based on the degradation methodology proposed in (Tsafrir et al., 2007) [22], and applied for scheduling in (Cortés-Mendoza et al., 2015) [6], (Cortés-Mendoza et al., 2016a) [7], (Tchernykh et al., 2016a) [23]. It shows how the metric generated by our algorithms gets closer to the best-found solution as:

$$\left(\frac{strategy\ metric\ value}{best\ found\ metric\ value} - 1\right) \cdot 100 \tag{6}$$

In multi-objective optimization, one solution can represent the best solution concerning provider cost, while another solution could be the best one concerning the QoS. The goal is to choose the most adequate solution and obtain a set of compromise solutions that represents a good approximation to the Pareto front.

A solution is Pareto optimal if no other solution improves it in terms of all objective functions. Any solution not belonging to the front can be considered of inferior quality to those that are included. The selection between the solutions included in the Pareto front depends on the system preference.

Often, results from multi-objectives problems are compared via visual observation of the solution space. One of the formal and statistical approaches uses Dominance Rank (DR) and Dominance Count (DC) [28]. A set of solutions, provided by different strategies, can be compared using both metrics. DR defines the number of solutions by which a solution is dominated, and DC considers the number of solutions dominated by a certain solution. On both metrics, the fitness of a solution is related to the whole set of solutions.

For a given set of solution S and a solution $i \in S$, $DR(i)$ defines the number of solutions that dominate i:

$$DR(i) = |\{j|\, j \in S \wedge i \geq j\}| + 1 \tag{7}$$

Similarly, $DC(i)$ represents the number of solutions that i dominates by:

$$DC(i) = |\{j|\, j \in S \wedge i \leq j\}| \tag{8}$$

where $|\cdot|$ denotes the cardinality of a set and the symbol $\leq$ corresponds to the Pareto dominance relation. Figure 3 shows an example of DR and DC for a minimization problem of f_1 and f_2. Solution c (red dot) dominates solution e (green dot) and it is dominated by solutions a (black dot) and b (orange dot), then $DR(c) = 3$ and $DC(c) = 1$.

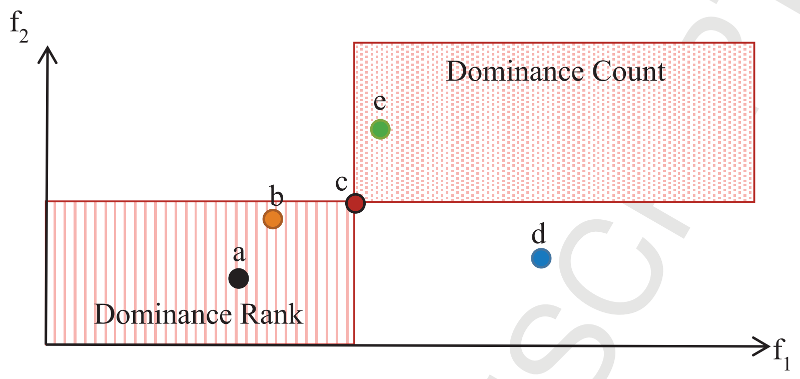

Figure 3: Example of dominance rank and dominance count.

The cardinality of S is the same for each scenario because strategies provide a single solution, so we can consider to use $|S|$ to normalize and average all scenarios. In Figure 3, solution a (black dot) dominates all other solution then $DC(a) = 4$, the mean dominance count (MDC) for solution a is $\frac{DC(a)}{|S|-1} = 1$. A metric value $MDC(i) = 1$ means that all solutions of S are dominated by solution i, whereas $MDC(i) = 0$ means that no member of S is dominated by i. This way, the larger the value of $MDC(i)$, the better the solution i with respect to S. In a similar way, we can define $MDR(i)$ to normalize the values and consider all scenarios.

We can generalize the terms $MDC(A)$ and $MDR(A)$ for the solution of strategies A under $|A|$ scenarios, see Equation (9) and Equation (10).

$$MDC(A) = \frac{1}{|A|}\sum_{i=1}^{|A|} MDC(a_i) \quad (9) \qquad MDR(A) = \frac{1}{|A|}\sum_{i=1}^{|A|} MDR(a_i) \quad (10)$$

5. Call allocation

The call allocation problem is similar to a well-known dynamic bin-packing problem, a variation of the classical NP-hard optimization problem with high theoretical relevance and practical importance. Bin-packing is a very active area of research in the algorithms and operations research communities.

In VoIP, the scheduler decides whether the call is placed into one of the currently available VMs or new VM should be run. The scheduler only knows the contribution of the call to the VM utilization u_j. All decisions have to be made without knowledge of duration of the call, call arrival and completion rates, etc.

Temporal existence of the items is the principal novelty of this problem. Call lifespan and allocations determine the state of the VMs. Unlike the standard formulation, bins are always open and dynamic, even wholly packed. Items in bins can be terminated (call termination) and utilization can be changed at any moments, then VMs can use free space to process more calls.

5.1 Configurable parameters

To study realistic scenarios, with gran variety of heterogeneous environments, we define three configurable parameters: Utilization Threshold (UT), Rental Threshold (RT), and Prediction Interval (PI). They can be configured and dynamically adapted to cope with different objective preferences, workloads, and cloud properties.

This approach allows adapting to cloud uncertainties such as dynamic elasticity, performance changing, virtualization, loosely coupling application to the infrastructure, parameters such as an effective processor speed, number of running virtual machines and actual bandwidth, among many others (Tchernykh et al., 2016b) [24].

UT is a parameter that is used as a trigger to request a new VM, when current utilization is reached the threshold. It is known that CPU cannot handle the stress when utilization is up to a particular level. Jitters and broken audio symptoms can appear. It depends on the CPU characteristics and allows to avoid voice quality reduction.

RT is a time interval before the VM finishes its renting period. It restricts calls allocation to VMs before rental time is finished. It avoids the case when a call arrives just before finishing of renting time and terminate after renting period causing continue VM renting for one hour more.

PI is a time interval for which a prediction is performed. Predictor performs an estimation of utilization each Time Step (TS). A smaller TS provides a better forecast but increases the overhead of the systems. Oppositely, a larger TS reduces the system overhead but decreases prediction accuracy.

VM StartUp time delay (StUp) is a time required to initialize a VM. It depends on several factors, but the most important is the Operative System and Software running in the VM (Mao and Humphreyet, 2012) [25], (Hoffman, 2017) [26].

Table 3 shows the StUp for several OS and providers. Table 4 presents the average times of VM creation and the time of accessing the SSH key among cloud vendors.

Table 3. Average VM startup time

Cloud	O.S.	StUp (sec.)
EC2	Linux	96.9
	Windows	810.2
Azure	WebRole	374.8
	WorkedRole	406.2
	VMRole	356.6
Rackspace	Linux	44.2
	Windows	429.2

Table 4. Average VM startup time with SSH access

Cloud	StUp (sec.)
Google Cloud Platform	31
Amazon Web Services	47
Vexxhost	47
Linode	57
DigitalOcean	89
Rackspace	128
Azure	138

In our case, we study how CVoIP can be affected by the time of the infrastructure deployment (StUp). We consider StUp = 45, 90, 135, 180, 225, 270 and 315 sec.

First, we consider a scenario, where the bin size equals to 0.7 that corresponds to UT=0.7. When utilization reaches this limit, a new VM is requested and provided after StUp time (see Table 5).

In the second scenario, each given TS, the system predicts VM utilization after PI=10, 20, 30, and StUp time intervals. If it is reached a UT limit, new VM is requested, so that after StUp time, new VM will be ready, exactly when it is needed by prediction.

Table 5. Configurable parameters.

Name		Values
UT	Utilization threshold	0.7
RT	Rental threshold	5, 10, and 15 min. (Cortés-Mendoza et al., 2016a) [7]
PI	Prediction interval	10, 20, 30, and StUp sec.
TS	Time step	10, 20, 30, and StUp sec.
StUp	Startup time delay	0, 45, 90, 135, 180, 225, 270 and 315 sec.

The system has no information of the calls arrival rate, and it takes decisions depending on the current system state. To analyze the behavior of the VoIP system under different circumstances, we configure parameters to consider several cloud and CVoIP providers.

5.2 Call allocation strategies

In the previous section, we describe how StUp time delay affects the call allocation. Figure 4 shows an example of call allocation with VM startup time delay.

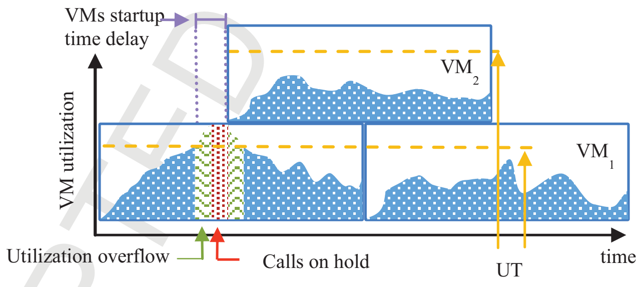

Figure 4: Calls allocation with startup time delay.

VM_2 is requested when the utilization of VM_1 is over UT. During VM_2 StUp, VM_1 continues call processing with possible utilization more than UT reducing QoS. The worst case appears when VM_1 does not have enough resources to process arriving calls. In this case, the system places call on hold, waiting for available resources.

Table 6 summarizes the call allocation strategies that request VM, when arriving call exceeds UT. Additionally, we incorporate a mechanism to reduce the number of active VMs. Rental-Threshold strategies limit the allocation of calls RT minutes before the VM finishes its renting.

Table 6. Call allocation strategies.

Type	Strategy	Description

KF	Rand	Allocates job *j* to VM randomly using a uniform distribution.
	RR	Allocates job *j* to VM using a Round Robin algorithm.
RA	MaxFTFit	Allocates job *j* to VM with farthest finish time.
	MidFTFit	Allocates job *j* to VM with shortest time to the half of its rental time.
	MinFTFit	Allocates job *j* to VM with closest finish time.
UA	FFit	Allocates job *j* to the first VM capable to execute it.
	BFit	Allocates job *j* to VM with smallest utilization left.
	WFit	Allocates job *j* to VM with largest utilization left.
KF + RT	Rand_05, 10, 15	Allocates job *j* to VM with RT equal to 5, 10 and 15 minutes using the Rand, and RR strategies.
	RR_05, 10, 15	
RA + RT	MaxFTFit_05, 10, 15	Allocates job *j* to VM with RT equal to 5, 10 and 15 minutes using the MaxFTFit, MidFTFit, and MinFTFit strategies.
	MidFTFit_05, 10, 15	
	MinFTFit_05, 10, 15	
UA + RT	BFit_05, 10, 15	Allocates job *j* to VM with RT equal to 5, 10 and 15 minutes using the BFit, FFit, and WFit strategies.
	FFit_05, 10, 15	
	WFit_05, 10, 15	

Table 7 shows the call allocation strategies with load prediction. Each given TS, the strategy estimates utilization for next PI, and, if it exceeds the UT of current VMs, it requests an additional VM.

Algorithm 1 describes the BFit_RT strategy, where two lists are supported: Admissible Voice Node List (AVNL) and no AVNL list (nAVNL). AVNL list contains the VNs with renting time that completes not less than in RT minutes. Both lists are sorted in not decreasing order of their utilization. We use the term VN instead of VM to have coherence with the call allocation terminology.

Table 7. Call allocation strategies with prediction.

Type	Strategy	Description
LA	Rand_StUp Rand_s10, 20, 30	Allocates job *j* to VM using the Rand, and RR strategies. They use PI equal to 10, 20, 30 and StUp seconds to estimate future load.
	RR_StUp RR_s10, 20, 30	
RA + LA	MaxFTFit_StUp MaxFTFit_s10, 20, 30	Allocates job *j* to VM using the MaxFTFit, MidFTFit, and MinFTFit strategies. They use PI equal to 10, 20, 30 and StUp seconds to estimate future load.
	MidFTFit_StUp MidFTFit_s10, 20, 30	
	MinFTFit_StUp MinFTFit_s10, 201, 30	
UA + LA	BFit_StUp BFit_s10, 20, 30	Allocates job *j* to VM using BFit, FFit, and WFit strategies. They use PI equal to 10, 20, 30 and StUp seconds to estimate future load.
	FFit_StUp FFit_s10, 20, 30	
	WFit_StUp WFit_s10, 20, 30	

Algorithm 1. Best Fit with Rental-Threshold (BFit_RT)

```
Input: Voice node list (VNlist), UT, RT and call.
Output: Allocation of call in one voice node.
```

```
vnIndex ← -1
Create AVNL and nAVNL lists with RT time.
Sort AVNL by utilization on decreasing order.
Sort nAVNL by utilization on decreasing order.
vnIndex ← Best_Fit(AVNL, UT, call)
    if vnIndex < 0 then
        vnIndex ←Best_Fit(nAVNL, UT, call)
        if vnIndex < 0 then
            vnIndex ←Best_Fit(nAVNL, 1.0, call)
            if vnIndex < 0 then
                vnIndex ←Best_Fit(AVNL, 1.0, call)
if vnIndex < 0 then
    Place call on hold
    Start a new node voice
else
    Insert call into VN with index vnIndex
Endif
```

When the VNs from AVNL cannot process the arriving calls due to exceeding the UT, the strategy searches on nAVNL. If a VN is available to process calls without exceeding UT, then the call is placed to it, the VN is moved to AVNL, and it schedules one hour more of rental time.

When the call cannot be assigned to VNs without QoS degradation, the strategy attempts to allocate the call on the nAVNL without QoS guarantee. If not, the call is placed on hold waiting for an available VM. The main goal of the algorithm is to use running VNs, even if they in nAVNL list, instead to start new VNs instances. The StUp reduces the QoS, so algorithm looks first on nAVNL list. In the worst case, algorithms start a new VN.

For all no RT strategies, AVNL list holds all the active VNs on the SNC. For example, RA strategies sort AVNL list by time and Rand does not sort AVNL list (Cortés-Mendoza et al., 2016a) [7].

An adequate prediction of calls arriving can reduce the quality degradation and number of calls on hold. Our allocation strategies use several prediction model to estimate the workload.

6. Load-aware strategies

An adequate load estimation could help to increase the efficiency of calls allocation. In the best case, the system can predict necessity of additional VNs exactly StUp time in advance, so that it has VM just in time when it needed to avoid reduction quality of service and increasing cost.

Figure 5 illustrates an example of load prediction. Assume that the predictor estimates the utilization for each TS. If the prediction exceeds UT, then the system requests a new VM.

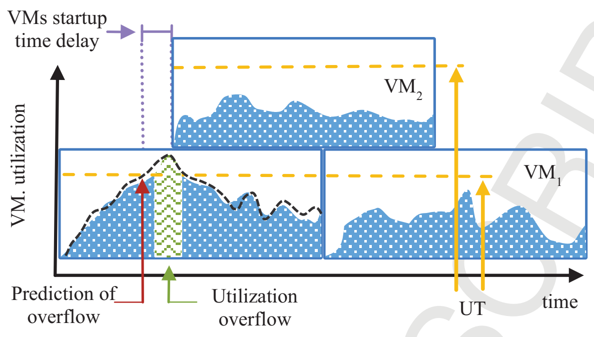

Figure 5: Calls allocation with prediction and startup time delay.

6.1 Rate of Change

Rate of Change – RoC (Campos and Scherson, 2000) [21] is a dynamic distributed load balancing algorithm; it achieves the goal of minimizing processor idling times without incurring into unacceptably high load overheads. It calculates the change in the load between two TSs.

The TS is an adaptive parameter, and its length may vary. RoC calculates the difference in load (Δ), and use it as estimation on load for the PI.

This estimation allows allocating calls more efficiently. A finer sampling allows to improve the balance of the system, but it increases the overhead.

We use the concept of Δ as a mechanism to predict requests for new VMs, which can be provided after StUp time. Δ is used to estimate the number of VMs each PI. It permits to initialize VMs before the arriving calls degrade the QoS.

Let $u_i(t)$ be the utilization of SNC_i at time t, and $k_i(t)$ the number of VMs running, then, the rate of load change during the TS is defined by:

$$\Delta_i(t) = (u_i(t) - u_i(t - TS))/k_i(t) \tag{11}$$

A positive value of $\Delta_i(t)$ means that utilization on the system increased in the period $[t - Ts, t]$ and a negative value implies that the utilization decreased. We estimate future load of the system by:

$$u_i(t + PI) = u_i(t) + \Delta_i(t) \tag{12}$$

Figure 6 shows this scenario. The solid lines represent real utilization and dashed lines are estimated utilization. If the utilization is larger than UT then the system requests for a new VM. UT is an adjustable parameter than depend on the number of running VMs and the utilization threshold. For instance, at time T_4, the system initiates a new VM based on predicted utilization for T_5. This is an example of incorrect over-provisioning. At time T_5, the system predicts utilization at T_6 less that UT, and does not request new VM. However, the real utilization is more than UT causing QoS degradation (under-provisioning). Finally, at time T_9, the system predicts utilization at T_{10} more that UT and requests new VM. This is an example of adequate VM provisioning.

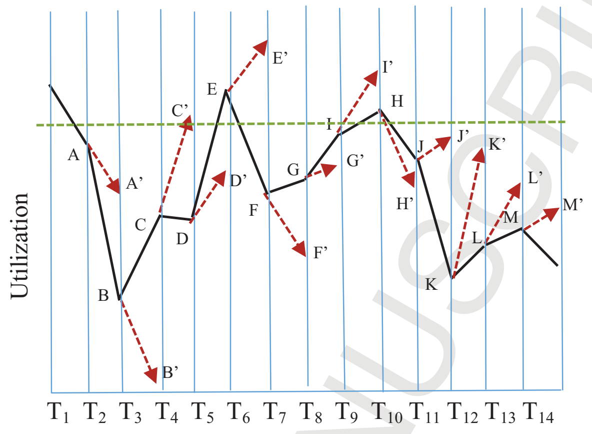

Figure 6: RoC prediction.

6.2 Neural Networks

Neural Networks (NNs) are widely used for prediction, classification or control, object recognition, etc.

Figure 7 shows the NN architecture for call prediction. Nodes represent artificial neurons, and arrows characterize connections between them. An interconnected group of nodes (the output of one neuron can be the input of another) forms the network.

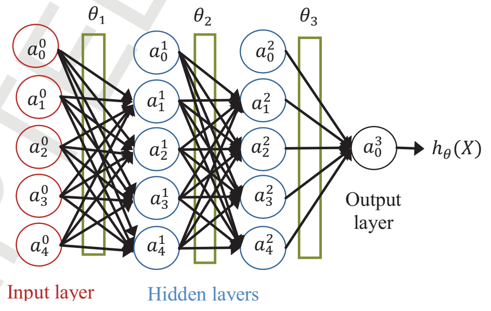
θ_1
θ_2
θ_3
a_0^0 a_1^0 a_2^0 a_3^0 a_4^0
a_0^1 a_1^1 a_2^1 a_3^1 a_4^1
a_0^2 a_1^2 a_2^2 a_3^2 a_4^2
a_0^3
$h_\theta(X)$
Output layer
Input layer
Hidden layers

Figure 7: Neural network architecture.

The input data to the NN is $X = [a_0^0, a_1^0, a_2^0, a_3^0, a_4^0]$. θ_1, θ_2, and θ_3 are the weight factors between layers 1, 2, and 3, respectively, and $h_\theta(X) = a_0^3$ is the output. In our case, $a_0^3 = u_i(t + PI)$. The inputs a_1^0, a_2^0, a_3^0 and a_4^0 are defined by; $a_1^0 = u_i(t - 2 * TS)$, $a_2^0 = u_i(t - TS)$, $a_3^0 = u_i(t)$ and $a_4^0 = t$.

The elements a_0^0, a_0^1, and a_0^2 define the bias at each layer. The NN considers a sigmoid (logistic) activation function defined by Equation (13). The activation of unit i (for i >0) in layer j (for j >0) is defined by Equation (14), where $s(\theta_i)$ is the number of weights factors (size) on θ_i.

$$g(x) = \frac{1}{1 + e^{-z}} \tag{13}$$

$$a_j^i = g\left(\sum_{j=0}^{s(\theta_i)} a_j^{i-1} * \theta_{i,j}\right) \tag{14}$$

Equation (15) defines the cost function, a generalization of logistic regression (Zhang, 2000) [29]. For a training set, $T = \{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$, where m is the number of elements on T, $y^{(i)}$ is the result value of the i-esim element, and $\mathrm{h}_\theta(x^{(i)})$ is the predicted value of $x^{(i)}$ inputs. The first term is the sum-of-logarithmic error between $\mathrm{h}_\theta(x^{(i)})$ and $y^{(i)}$.The second term is a regularization term, where λ controls the relative importance of the two terms and L is the number of layers in the NN, see Equation (16).

$$J(T) = \frac{-1}{m}\left[\sum_{i=1}^{m} \begin{matrix} y^{(i)} \log\left(\mathrm{h}_\theta(x^{(i)})\right) + \\ \left(1 - y^{(i)}\right) \log\left(1 - \mathrm{h}_\theta(x^{(i)})\right) \end{matrix}\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1} \theta_l^2 \tag{15}$$

$$\theta_l^2 = \sum_{j=1}^{s(\theta_l)} (\theta_{i,j})^2 \tag{16}$$

Our goal is to train a NN and minimize $J(T)$ in order to get the best values for θ_1, θ_2, and θ_3. Then, we use them to predict the calls arriving. Algorithm 2 shows the steps of training a NN, where we use descend gradient to perform backpropagation.

Algorithm 2. Training Neural Network

```
Input: Training set T, testing set R, termination (ter).
Output: Values of $\theta_1$, $\theta_2$, and $\theta_3$ that minimize $J(R)$
1   Normalization of T and R.
2   Initialization of $\delta_1$, $\delta_2$, $\delta_3$ and $\delta$
3   For j ← 1 to 30
4      Random initilization of $\theta_1$, $\theta_2$, and $\theta_3$
5      While ter ≠ true do
6         For each $x_i$ into T
7            Compute $h_\theta(x_i)$
8         Compute $J(T)$
9         Apply backpropagation
10        Update $\theta_1$, $\theta_2$, and $\theta_3$
11     If $\delta > J(R)$
12        $\delta \leftarrow J(R)$
13        $\delta_1 \leftarrow \theta_1$, $\delta_2 \leftarrow \theta_2$, $\delta_3 \leftarrow \theta_3$
14  Return $\delta_1$, $\delta_2$, and $\delta_3$
```

6.3 History based prediction

To compare the efficiency of calls predictors previously described, we developed a historical predictor that consider the actual utilization and historical information about the VM utilization.

At time t, the historical predictor estimates $u_i\ (t + PI)$ based on: $u_i\ (t)$ and $\widehat{u_i}(t + PI)$, where $\widehat{u_i}(t)$ defines the historical standard deviation of utilization at time t.

Figure 8 shows an example of the historical utilization by day with a TS of 315 seconds, the average utilization and deviation standard consider 85 days of workload. We estimate future load of the system by:

$$u_i\ (t + PI) = u_i(t) \pm \left(x * \widehat{u_i}(t + PI)\right), \tag{17}$$

where x is a single uniformly distributed random number in the interval [0, 1]. Additionally, we generate a random value to define if the utilization is increased or decreased ($\pm$).

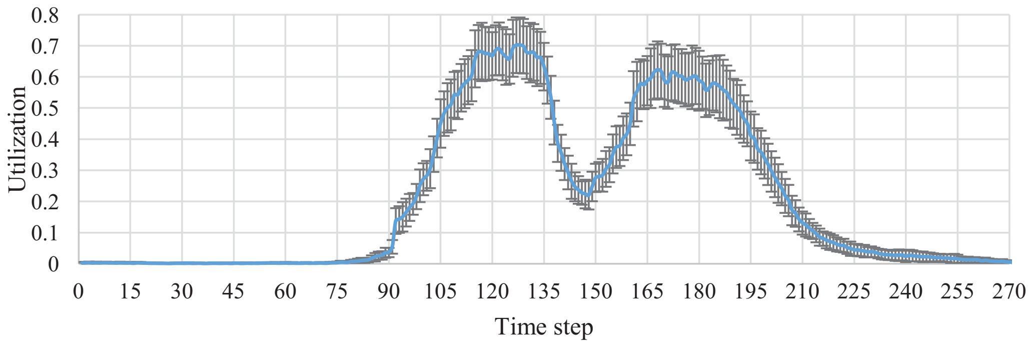

Figure 8: History based prediction with 315 sec. of TS.

7. Neural network setup

To find the best version of the Neural Network (NN), several parameters have to be set up. First, we search for an adequate λ value. Then, we consider utilization of the system within three intervals $(t - TS, t]$, $(t - 2 * TS, t - TS]$ and $(t - 3 * TS, t - 2 * TS]$ to estimate $u_i(t + TS)$. Finally, we vary the number of elements in the training sets to avoid overfitting.

We evaluate the algorithms with $PI = TS = stUp = 315$ seconds. As we state before, in the best case, additional VM starts accepting calls exactly when it needed.

The values of θ_1, θ_2, and θ_3 are computed for each model with the following configuration: λ ={10.0, 1.0, 0.5, 0.1, 0.01, 0.001, 0.0001, 0.00001}. All configurations are trained with 15 days of workload, and validate with other 15 days. To obtain their best values, the training process are conducted 30 times.

We use Root Mean Square Deviation (RMSD) and Symmetric Mean Absolute Percentage Error (SMAPE) to evaluate the performance of the algorithm. They are defined by equations (18) and (19), respectively, and applied to measure the accuracy of a method.

$$\text{RMSD} = \sqrt{\frac{\sum_{i=1}^{n}(F(i) - A(i))^2}{n}} \tag{18}$$

$$SMAPE = \frac{\sum_{i=1}^{n}|F(i) - A(i)|}{\sum_{i=1}^{n}|A(i) + F(i)|}, \quad (19)$$

where n is the size of the set, $A(i)$ defines the actual value and $F(i)$ defines the forecast value. Smaller values of RMSD and SMAPE are better.

Table 8 shows the calculated values of RMSD and SMAPE for different λ values. Note that the best algorithms have smallest values of λ for both metrics.

The NN uses utilization of three TSs to predict PI. To train the NN, we analysis three possible values for $u_i(t)$, $u_i(t)$ - the utilization at time t, $u_i(t) = \sum_{k=t-TS}^{t}\frac{u_i(k)}{TS}$ the average utilization in the interval $(t - TS, t\,]$, and $u_i(t) = \max\,(u_i(t - TS), u_i(t)]$ the maximum utilization in the interval $(t - TS, t\,]$.

Table 8. RMSD and SMAPE for $\boldsymbol{\lambda}$.

λ	RMSD	SMAPE
0.00001	**106.739660**	**0.055020**
0.0001	107.239957	0.055468
0.001	108.844112	0.056072
0.01	118.981582	0.060927
0.1	120.277697	0.062772
0.5	130.424748	0.072682
1.0	138.546018	0.079169
10.0	167.208843	0.10274

We evaluate three versions of NNs with the previous configurations and smallest λ value ($\lambda = 0.00001$). Finally, we select the best model to use as call predictor.

Table 9 shows the value of RMSD and SMAPE for different values of VM utilization for prediction: current, average, and maximum. We see that the best version is the NN that considers the current utilization at time t.

Table 9. RMSD and SMAPE for utilization values.

Utilization value	RMSD	SMAPE
Actual	**106.739660**	**0.055020**
Average	111.681136	0.056846
Maximum	110.478730	0.057947

Finally, we analyze different training sets. We vary the number of days in the training set and tune the resulting values of θ_1, θ_2, and θ_3 in the predictor.

We consider four sets with 30, 45, 60, and 75 workloads to conform the training set. The training sets contain 15 workloads from the testing set, five workloads from validation set and the rest are new workloads.

Validation set shares five workloads from the training set and ten new workloads. Testing set contains 30 workloads, sharing 15 workloads with the training set and 15 new workloads.

Table 10 shows the value of RMSD and SMAPE for different training set versions. We see that the best version of NN is training with 75 days of workload and $\lambda = 0.00001$.

Table 10. RMSD and SMAPE for different length training sets.

Workloads per training set	RMSD	SMAPE
30	115.679943	0.054781
45	116.057055	0.055013
60	115.563836	0.054721
75	**115.026791**	**0.054598**

8. Experimental setup

We perform experiments using standard trace based simulator CloudSim (Garg and Buyya, 2011) [27] extended by our algorithms, supporting dynamic calls arrival, VM startup delays, statistical analysis, and predictions models.

We evaluate eight strategies: BFit, FFit, MaxFTFit, MidFTFit, MinFTFit, Rand, RR and WFit, with PI and TS equals to 10, 20, 30, and StUp (time interval is equal to startup time delay). In total, thirty-two strategies per prediction model (rate of change, neural network and history based) are evaluated.

8.1 Workload

We use traces of the real VoIP service provider (Simionovici et al., 2015) [19]. They include phone calls with the following information: Index of the call; ID of the user who makes the call; IP of the phone where the call is placed from; IP of the local phone; Destination of the call; Destination country code; Destination country name; Telecommunications service provider; Beginning of the call (timestamp); Duration of the call (in seconds); Duration of a paid call; Cost per minute; etc.

A number of calls per day and their duration are presented in Table 11 and Table 12. The histogram of the number of calls per hour during a day shows that the load is typical for business clients with two peaks in 10-12 and 14-16 hours (Simionovici et al., 2015) [19].

Table 11. Number of calls per day.

Day	Total	Average
Monday	131,443	21,906
Tuesday	129,379	21,563
Wednesday	131,460	21,910
Thursday	130,439	21,739
Friday	120,999	20,166

Table 12. Call duration.

Time (min.)	Number of calls
0 - 1	310,602
1 - 2	136,211

2 - 3	68,988
3 - 4	39,392
4 - 5	23,397
5 - 6	15,075
6 - 7	10,009
7 - 8	7,256
8 - 9	5,536
9 - 10	4,202
…	…
19 - 20	721

9. Experimental analysis

The VoIP providers rent VMs on an hourly base. When the VM rental time is finished, the VM can be turned off only if VM is not processing calls. In any other case, this VM continue running for one hour more.

First, we analyze the benefits of Rental-Threshold versions, when strategies take into account the exact time of each VM provisioning. Second, we study the advantages of load-aware versions with three prediction mechanisms. Finally, we perform a comparative study of the best-found strategies in more details.

In order to evaluate their robustness, we consider eight StUp delays that taken from real cloud providers: 0, 45, 90, 135, 180, 225, 270, and 315 seconds (Mao and Humphreyet, 2012) [25], (K. Hoffman, 2017) [26]. StUp=0 represents the perfect predictor because the system starts VMs when it needs. It is used as a reference point to evaluate the performance of other strategies under several StUps. Table 13 presents the simulation setup for experimental analysis.

Table 13. Simulation setup.

Name	Value
Total call allocation strategies:	128
Without prediction	32
With prediction	96
Number of Workloads	30
Workload length	24 hours
Number of codecs	4
Number of calls	643,718
VMs StUp	8

In next section, we do not present voice quality reduction ($\bar{q}$) due to the difference between no quality reduction degradation and the worst degradation is about 0.92 %. We analyze number of calls placed on hold $\bar{c}$ as a representation of QoS reduction.

9.1 Rental-Threshold strategies

In this section, we evaluate eight strategies: BFit, FFit, MaxFTFit, MidFTFit, MinFTFit, Rand, RR and WFit, and three RT versions of each of them.

Figure 9 shows the billing hours degradation ($\bar{b}$) versus StUps. We see that the scheduling strategies tend to be robust with respect to $\bar{b}$. The StUp does not change $\bar{b}$, considerably. We also observe that strategies with better performance are BFit and FFit. The average hold calls ($\bar{c}$) is about 6 per day during 30 days for MaxFTFit_15 (the worst strategy) with StUp equals to 270 sec (Figure 10).

Table 14 shows the average reduction of $\bar{b}$. RT technique benefits strategies with high fluctuation of $\bar{b}$ (MinFTFit, Rand, RR, and WFit), and affects other with low fluctuation of $\bar{b}$ (BFit and FFit). Table 15 presents the average $\bar{c}$ of the strategies. We see that BFit_RT and FFit_RT increase $\bar{b}$ and $\bar{c}$, while another RT strategies reduce $\bar{b}$.

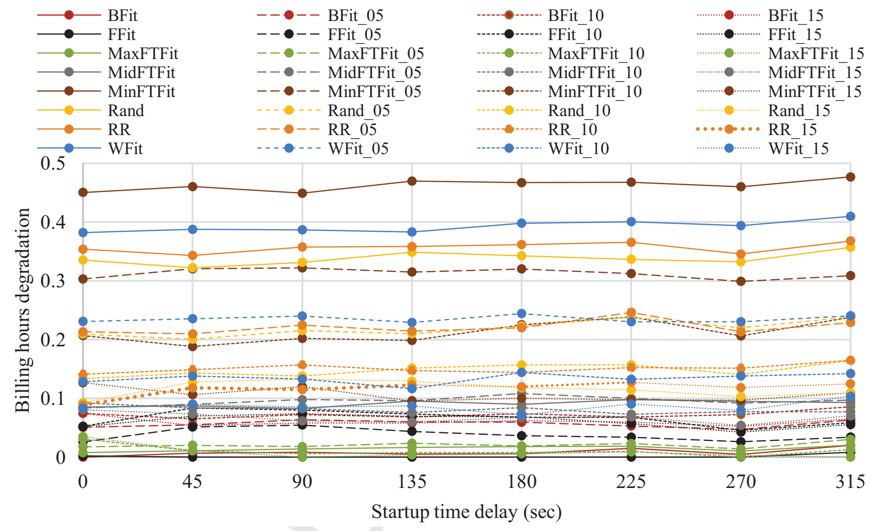

Figure 9: Billing hour degradation for strategies with RT.

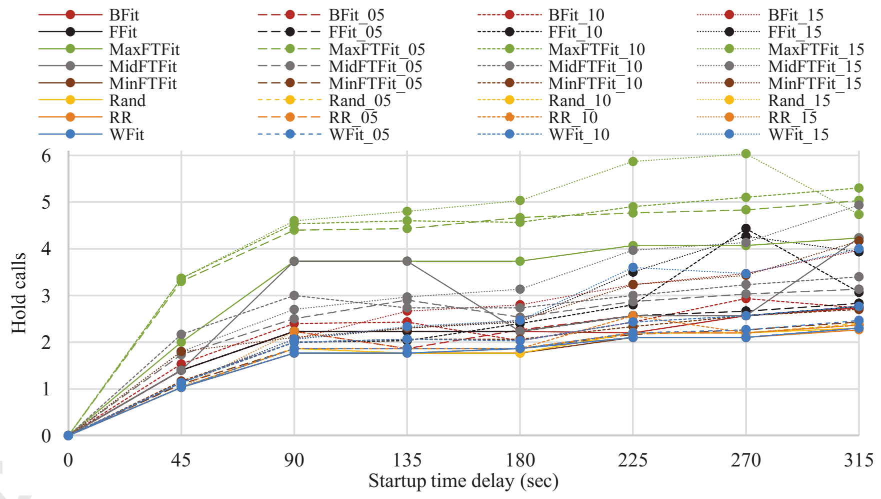

Figure 10: Average hold calls per day for strategies with RT.

Table 14. Average billing hours reduction (%).

RT / Strategy	05 min	10 min	15 min.

Table 15. Average hold calls per day.

RT / Strategy	No RT	05 min	10 min	15 min.

BFit	- 4.69	- 6.27	- 4.8
FFit	- 4.79	- 6.4	- 5.18
MaxFTFit	- 0.61	0.38	0.59
MidFTFit	- 0.44	0.91	2.19
MinFTFit	10.25	17.07	24.34
Rand	8.88	14.21	16.61
RR	9.97	15.17	17.67
WFit	11.3	18.56	21.98

BFit	1.95	1.95	2.06	2.42
FFit	2.00	2.02	2.23	2.45
MaxFTFit	3.20	3.93	4.05	4.30
MidFTFit	2.56	2.34	2.53	2.95
MinFTFit	1.60	1.70	1.86	2.44
Rand	1.64	1.68	1.68	1.68
RR	1.61	1.68	1.72	1.72
WFit	1.62	1.70	1.88	2.38

9.2 Load-aware strategies with RoC

Figure 11 shows $\bar{b}$ degradation versus StUps. We observe that strategies with better

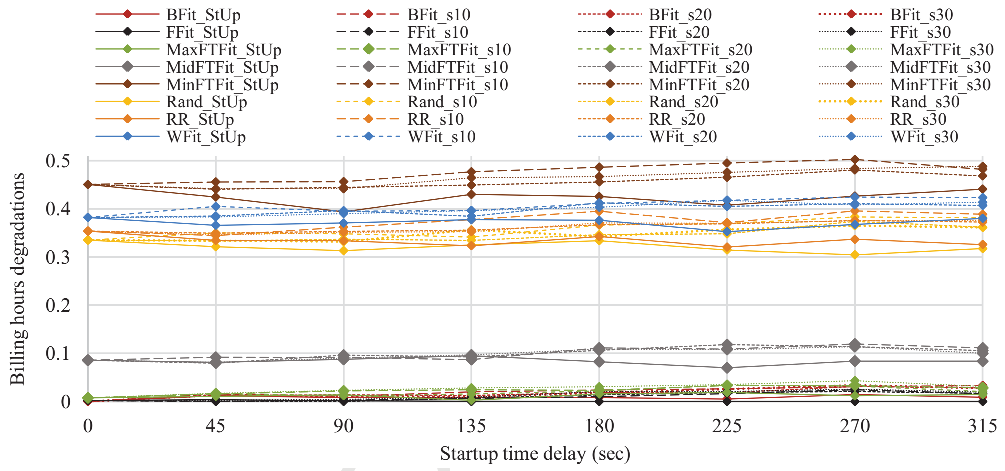

Figure 11: Billing hours degradation for strategies with RoC estimation.

performance with respect to the billing hours are BFit_StUp, FFit_StUp and MaxFTFit_StUp, and the worst strategies are different versions of MinFTFit.

Similar to strategies with RT, load-aware strategies tend to be robust with respect to $\bar{b}$. The StUp does not change it considerably. The average hold calls is about 6 per day for the worst strategy MaxFTFit_StUp with StUp equals to 315 sec. (Figure 12). The best strategies are MinFTFit_s30, Rand_s30, RR_s30, and WFit_s30, $\bar{c}$ varies between 0.9 and 2.26 calls per day.

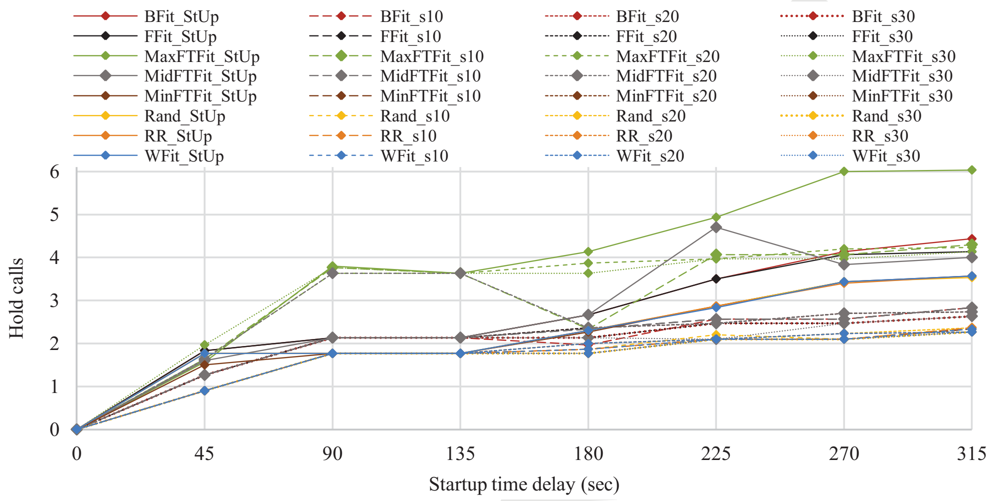

Figure 12: Average hold calls per day for strategies with RoC estimation.

9.3 *Load-aware strategies with NN*

For neural network prediction, the strategies with the best performance on $\bar{b}$ are FFit_StUp, Bfit_StUp, and MaxFTFit_StUp. They use StUp time delay as prediction time, and have the worst performance on $\bar{c}$ (Figure 13, Figure 14).

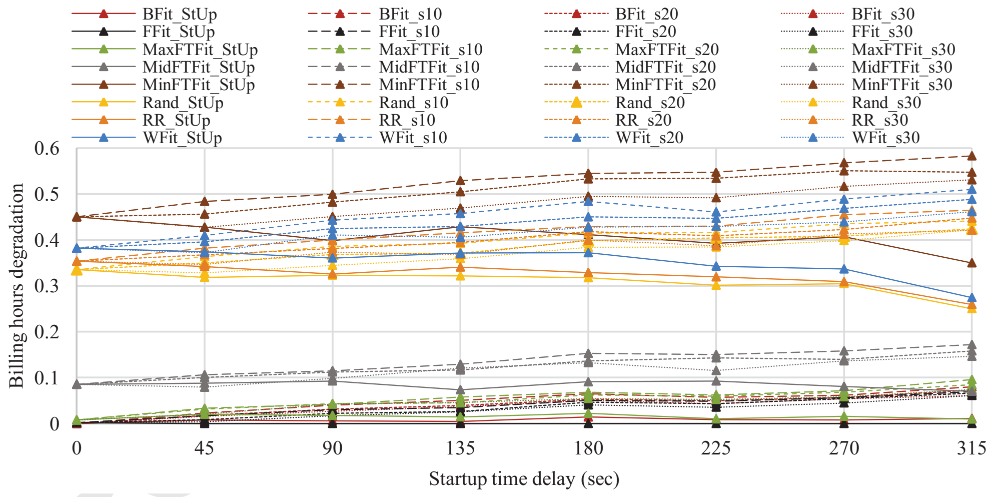

Figure 13: Billing hours degradation for strategies with NN estimation.

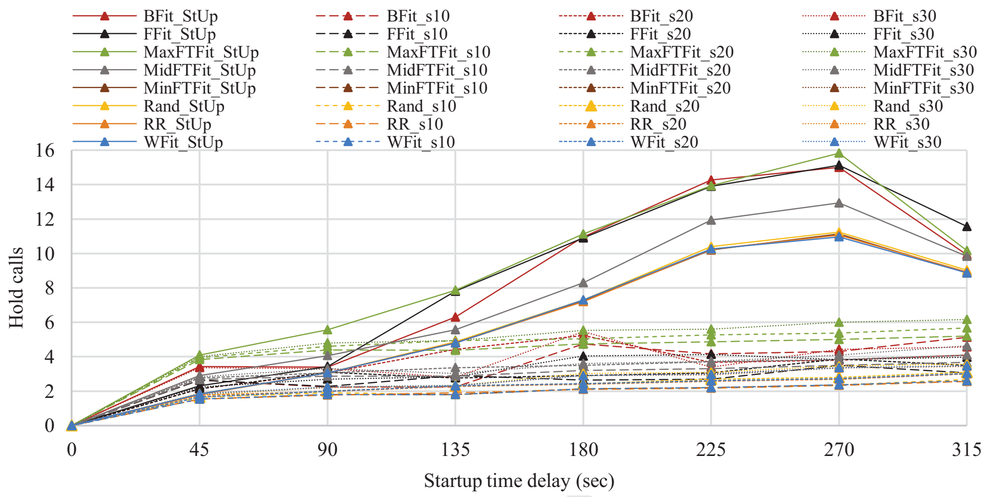

Figure 14: Average hold calls per day for strategies with NN estimation.

MaxFTFit_StUp is the worst strategy. It uses NN as prediction model with StUp equals to 270 sec. Its average $\bar{c}$ is about 16 calls per day, hence, it places on hold the 0.073% of calls during one month (see Figure 14).

9.4 Load-aware strategies with History based prediction

Similar to RoC and NN, the best strategies on $\bar{b}$ that use History based prediction have StUp time delay as prediction time. They have the worst performance on $\bar{c}$ (see Figure 15 and Figure 16). The average $\bar{c}$ is about 8 calls per day for the worst strategy MaxFTFit_StUp with StUp equals to 270 sec. (see Figure 14).

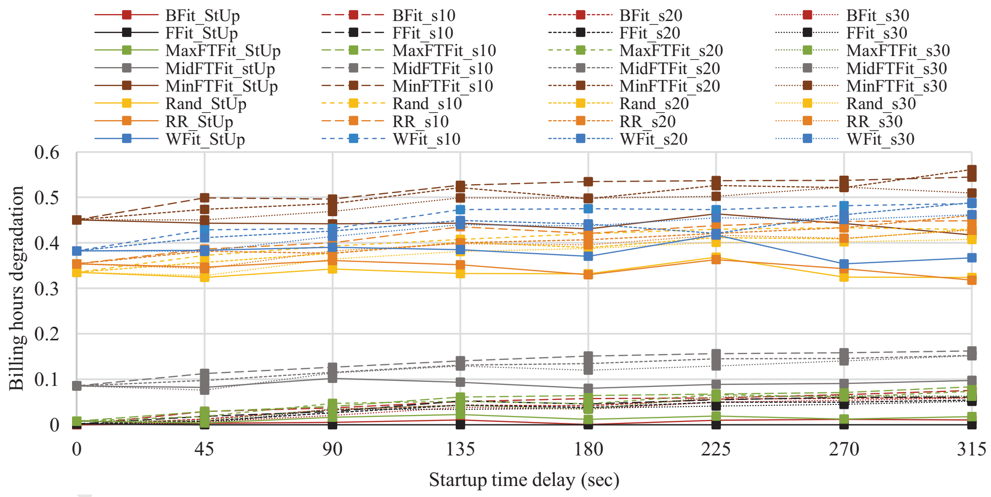

Figure 15: Billing hours degradation for strategies with history based estimation.

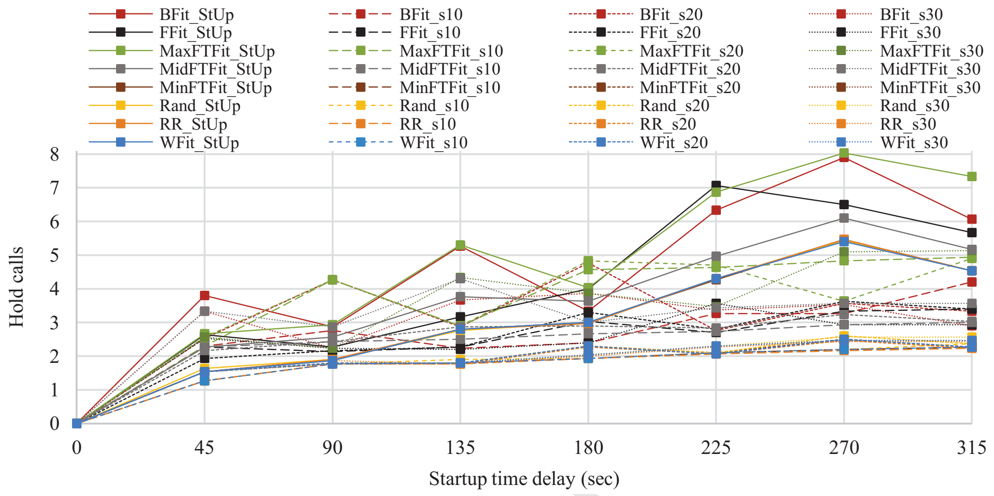

Figure 16: Average hold calls per days for strategies with history based estimation.

Figure 17 shows an example of load estimation, 12 hours with PI of 315 seconds, for three strategies and the real load values. Figure 18 presents the same values for half hour with PI of 10 seconds. On both figures, RoC prediction fits better, closer to the real values.

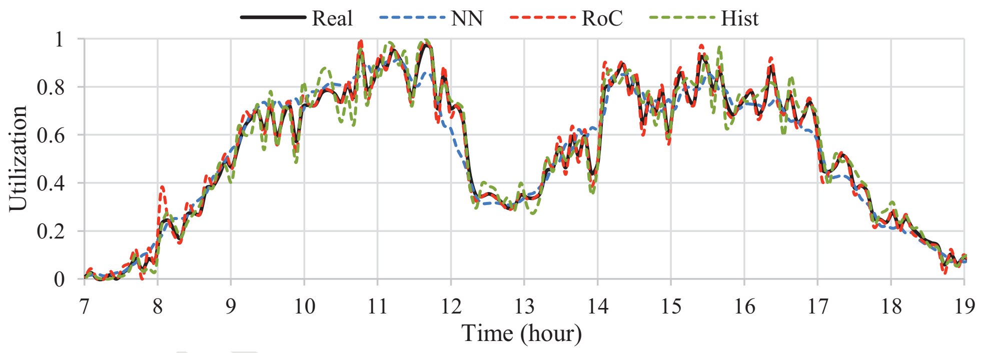

Figure 17: Utilization estimation with 315 sec. of PI.

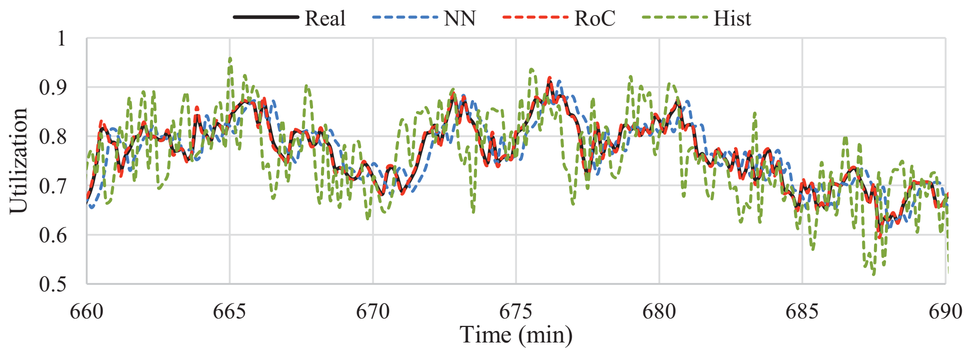

Figure 18: Utilization estimation with 10 sec. of PI.

9.5 Bi-objective analysis

In multi-objective analysis, the problem can be simplified to a single objective one through different methods of objective weighted aggregation. There are various ways to model preferences. For instance, they can be given explicitly to specify the importance of every criterion or a relative importance between criteria. This can be done by a definition of criteria weights or criteria ranking by their importance.

In this section, we perform a joint analysis of two metrics according to the mean degradation methodology (see Formula 6). First, we present the analysis of $\bar{b}$ and $\bar{c}$ separately. Then, we find the strategy that generates the best compromise between them.

Tables 16 presents the average degradation of $\bar{b}$, $\bar{c}$, and their means. The last four columns contain the ranking of each strategy regarding to the provider cost, quality, means, and final ranking. Rank $\bar{b}$ is based on the billing hours degradation. Rank $\bar{c}$ refers to the position in relation to the calls on hold. Rank mean is the position based on the averaging two ranking. Rank is the relative position to all strategies, it considers the mean of Rank $\bar{b}$ and Rank $\bar{c}$. Tables 19 – 22 contain all information about mean degradation and ranking for all strategies, see appendix A.

Table 16. Degradations and ranking for the best allocation strategies.

Strategy	Deg $\bar{b}$	Deg $\bar{c}$	Mean	Rank $\bar{b}$	Rank $\bar{c}$	Rank mean	Rank
NN_FFit_StUp	0.0000	4.1342	2.0671	**1**	100	127	32
NN_BFit_StUp	0.0084	3.9974	2.0029	**2**	99	126	32
NN_MaxFTFit_StUp	0.0157	4.4158	2.2158	**3**	101	128	34
RoC_WFit_s30	0.4455	0.0000	0.2227	105	**1**	30	36
RoC_MinFTFit_s30	0.5118	0.0000	0.2559	118	**1**	36	49
RoC_Rand_s30	0.3918	0.0000	0.1959	85	**1**	21	18
RoC_RR_s30	0.4056	0.0000	0.2028	91	**1**	22	23
RoC_MinFTFit_s10	0.5253	0.0132	0.2692	121	**2**	41	53
RoC_WFit_s10	0.4543	0.0132	0.2338	106	**2**	33	38
RoC_RR_s10	0.4192	0.0158	0.2175	95	**3**	27	29
Rand_15	0.1731	0.0579	0.1155	64	11	**1**	12
RoC_FFit_s30	0.0464	0.2026	0.1245	15	28	**2**	**2**
RoC_BFit_s30	0.0517	0.2026	0.1272	20	28	**3**	**3**
RoC_FFit_s10	0.0425	0.2500	0.1462	9	33	9	**1**
RoC_FFit_s20	0.0427	0.2474	0.1451	10	32	8	**1**
RR	0.4222	0.0184	0.2203	**97**	**4**	**29**	**32**

We see that the best strategy for $\bar{b}$ is NN-FFit_StUp that allocates calls based on NN prediction and First Fit strategy, where we put the call into the first VM. The PI is equal to VM startup time StUp. NN-BFit_StUp and NN-MaxFTFit_StUp are the second and third best strategies (with respect to $\bar{b}$). These strategies use neural networks to predict future calls arrival. However, they are no efficient strategies for $\bar{c}$ criterion. They tend to increase utilization, and reduce quality.

The best strategies for $\bar{c}$ are RoC-MinFTFit_s30 (where we put calls into the VM, with rental time closest to completion), RoC-Rand_s30, RoC-RR_s30, and RoC-WFit_s30. They use Rate of Change to predict the workload and prediction interval is 30 seconds. It tends to under-utilize VMs reducing $\bar{c}$ and increasing $\bar{b}$.

Strategies with good compromise between $\bar{b}$ and $\bar{c}$ are: RT-Rand_15, where we allocate the calls randomly into the VMs, but try to reduce $\bar{b}$ by limit the allocation of calls 15 minutes before VMs achieve their rental time. RoC-FFit_s30 and RoC-BFit_s30 strategies that allocate calls to VMs using FFit and BFit, respectably, and they predict the load with Rate of Change each 30 seconds.

Finally, 28 strategies improve the performance of RR considering the average degradation of $\bar{b}$ and $\bar{c}$. The rank of RR is 32, with respect to the relative position of all the strategies, it means that at least 52 strategies have a performance equal or better than this strategy.

Using the dominance rank and dominance count metric, a set of solutions can be compared. Table 17 presents the MDR values for the best strategies. The rows of the table show the mean dominance rank values of strategy A under different StUps (columns). The last two columns show the mean $MDR(A)$, and ranking based on the average dominance. The lower ranking implies that the solution is better. According to the dominance rank, the strategy that has the best compromise between the number of billing hours and quality reduction is NN_FFit_StUp, followed by NN_BFit_StUp, NN_MaxFTFit_StUp and RoC_FFit_s10.

We see that NN_FFit_StUp is dominated by other strategies in the range 11-19%, with 15.1% in average occupying the first rank. Meanwhile, NN_BFit_StUp and NN_MaxFTFit_StUp with the second and third ranks are dominated by other strategies on 19.3% and 22.9%, on average, respectively.

Table 17. Mean dominance rank values for the best strategies.

Strategy	45	90	135	180	225	270	315	mean	rank
NN_BFit_StUp	0.263	0.208	0.208	0.188	0.160	0.154	0.169	0.193	2
NN_FFit_StUp	0.196	0.183	0.171	0.110	0.129	0.133	0.133	0.151	1
NN_MaxFTFit_StUp	0.335	0.275	0.246	0.225	0.183	0.177	0.158	0.229	3
Rand_15	0.458	0.442	0.381	0.344	0.331	0.321	0.363	0.377	8
RoC_BFit_s30	0.360	0.327	0.265	0.240	0.242	0.229	0.275	0.277	7
RoC_FFit_s10	0.288	0.292	0.252	0.215	0.206	0.202	0.233	0.241	4
RoC_FFit_s20	0.294	0.275	0.263	0.229	0.227	0.200	0.244	0.247	5
RoC_FFit_s30	0.310	0.317	0.271	0.260	0.238	0.200	0.252	0.264	6
RoC_MinFTFit_s10	0.900	0.881	0.827	0.802	0.746	0.771	0.792	0.817	16
RoC_MinFTFit_s30	0.885	0.875	0.823	0.754	0.750	0.752	0.810	0.807	15
RoC_Rand_s30	0.631	0.594	0.571	0.500	0.463	0.515	0.558	0.547	9
RoC_RR_s10	0.669	0.673	0.638	0.615	0.490	0.606	0.592	0.612	10
RoC_RR_s30	0.692	0.654	0.573	0.571	0.775	0.538	0.592	0.628	12
RoC_WFit_s10	0.823	0.769	0.700	0.663	0.610	0.658	0.702	0.704	14
RoC_WFit_s30	0.775	0.763	0.713	0.654	0.621	0.615	0.685	0.689	13
RR	0.681	0.692	0.627	0.585	0.552	0.546	0.606	0.613	11

Table 18 presents the MDC values for the best strategies. The rows of the table show the mean dominance count values of strategy A under different StUps (columns). The last two columns show the mean $MDC(A)$, and ranking based on the average dominance. The higher ranking implies that the solution is better. According to the dominance count, the strategy that has the best compromise between the number of billing hours and quality reduction is RoC_FFit_s20, followed by RoC_FFit_s10, RoC_FFit_s30, and RoC_BFit_s30.

Table 18. Mean dominance count values for the best strategies.

Strategy	45	90	135	180	225	270	315	mean	rank
NN_BFit_StUp	0.780	0.698	0.427	0.249	0.184	0.169	0.320	0.404	7
NN_FFit_StUp	0.813	0.727	0.444	0.313	0.200	0.176	0.358	0.433	6
NN_MaxFTFit_StUp	0.644	0.593	0.367	0.244	0.178	0.138	0.351	0.359	9
Rand_15	0.524	0.516	0.516	0.518	0.518	0.529	0.518	0.520	5
RoC_BFit_s30	0.738	0.731	0.758	0.736	0.711	0.676	0.700	0.721	4
RoC_FFit_s10	0.851	0.802	0.798	0.751	0.756	0.718	0.713	0.770	2
RoC_FFit_s20	0.851	0.818	0.800	0.756	0.764	0.720	0.736	0.778	1
RoC_FFit_s30	0.831	0.789	0.769	0.724	0.747	0.751	0.744	0.765	3
RoC_MinFTFit_s10	0.089	0.087	0.087	0.073	0.100	0.082	0.118	0.091	16
RoC_MinFTFit_s30	0.116	0.102	0.102	0.113	0.133	0.093	0.084	0.106	15
RoC_Rand_s30	0.376	0.391	0.376	0.402	0.409	0.400	0.387	0.391	8
RoC_RR_s10	0.344	0.327	0.316	0.293	0.391	0.307	0.333	0.330	11
RoC_RR_s30	0.349	0.344	0.393	0.342	0.096	0.360	0.371	0.322	12
RoC_WFit_s10	0.269	0.286	0.264	0.252	0.263	0.230	0.254	0.260	13
RoC_WFit_s30	0.242	0.233	0.244	0.253	0.273	0.276	0.260	0.255	14
RR	0.331	0.316	0.324	0.329	0.347	0.358	0.342	0.335	10

We see that RoC_FFit_s20 dominates the other strategies in all scenarios in 72-85% range (77.8% in average) occupying the first rank. Meanwhile, RoC_FFit_s10 and RoC_FFit_s30 with the second and third ranks dominate other strategies on 77% and 76.5%, on average, respectively.

Finally, $MDR(RR)$ implies that RR is dominated by other strategies in 54-69% range with 61.28%, in average. $MDC(RR)$ indicates that RR dominates other strategies in the range of 31-35% with 33.5% in average.

10.Conclusion

In this paper, we propose a configurable cloud-based VoIP model as a special case of dynamic multi-objective bin-packing. We introduce three configurable parameters for model bases on bin packing: utilization threshold, rental threshold, and prediction interval (PI) to cope with different objective preferences, workloads, and cloud properties. They can be dynamically adapted to environmental changes.

We analyze a multi-objective problem considering billing hours for VM provisioning, number of calls placed to hold, and voice quality criteria. We propose thirty-two on-line non-clairvoyant scheduling strategies with static policy of VM provisioning and ninety-six allocation strategies with dynamic criteria for VM provisioning based on load prediction.

We classify call allocation strategies based on types and amounts of information used for VM provisioning as knowledge-free, utilization-aware, rental-aware, and load-aware.

We conduct comprehensive simulation analysis on the real workload of the MIXvoip company and show that our strategies improve the performance of known strategies and those that are currently in use. They increase voice quality, reduce provider cost and amount of calls placed on hold.

Our experimental analysis results in several contributions:

1) To provide effective guidance in choosing a good strategy, we perform a joint analysis of conflicting goals based on the degradation in performance of each strategy under each metric and consider coverage rank and coverage count to provide guidance to select an adequate strategy.

2) We show that our strategies have a high quality of service. Only 0.073% of calls are placed on hold during one month, with 0.92% of quality reduction, in the worst case, comparing with the deterministic solutions.

3) We demonstrate that proposed strategies with RT and RoC dynamic prediction outperform known ones in provider cost, slightly decreasing quality of service. They can reduce the billing hour up to 30%, increasing calls placed on hold from about 2 to 9 per day.

4) We evaluate their robustness as the ability to withstand adverse variations of VM startup time delays. We demonstrate that the strategies have a low variation of three criteria even with high dispersion of VM startup time delays.

However, further study is required to assess their actual performance and effectiveness in a real domain. This will be the subject of future work.

Acknowledgements

This work is partially supported by Russian Foundation for Basic Research (RFBR) 18-07-01224 and Act 211, Government of the Russian Federation [contract number 02.A03.21.0011]; and CONACYT, México [grant number 178415].

References

[1] Amazon Web Services (AWS), https://aws.amazon.com/es/solutions/case-studies/?nc2=h_ql_ny_livestream_blu, accessed August 20, 2017.

[2] Google Cloud Platform (GCP), https://cloud.google.com/customers, accessed August 20, 2017.

[3] MIXvoip, https://www.mixvoip.com/, accessed August 20, 2017.

[4] L. Madsen, J. V. Meggelen, and R. Bryant. Asterisk: The definitive guide. O'Reilly Media, Inc., 2011.

[5] H. P. Singh, S. Singh, J. Singh, and S. A. Khan. VoIP: State of art for global connectivity—A critical review. Journal of Network and Computer Applications, 37, 365-379, 2014.

[6] J. M. Cortés-Mendoza, A. Tchernykh, A. M. Simionovici, P. Bouvry, S. Nesmachnow, B. Dorronsoro, and L. Didelot. VoIP service model for multi-objective scheduling in cloud infrastructure. International Journal of Metaheuristics, 4(2), 185-203, 2015.

[7] J. M. Cortés-Mendoza, A. Tchernykh, F. A. Armenta-Cano, P. Bouvry, A. Yu. Drozdov, and L. Didelot. Biobjective VoIP Service Management in Cloud Infrastructure. Scientific Programming, vol. 2016, Article ID 5706790, 2016.

[8] J. M. Cortés-Mendoza, A. Tchernykh, A. Yu. Drozdov, and L. Didelot. Robust cloud VoIP scheduling under VMs startup time delay uncertainty. 9th International Conference on Utility and Cloud Computing, 234-239, 2016.

[9] J. M. Cortés-Mendoza, A. Tchernykh, A. Feoktistov, I. Bychkov, P. Bouvry, and Didelot. Load-Aware Strategies for Cloud-based VoIP Optimization with VM Startup Prediction. 7th IEEE Workshop Parallel and Distributed Computing and Optimization (PDCO), 2017.

[10] C. Eleftheriou. 3CX Phone System and ATOM N270 Processor Benchmarking. http://www.3cx.com/blog/voip-howto/atom-processor-n270-benchmarking, accessed August 20, 2017.

[11] A. Montazerolghaem, S. Shekofteh, M. Yaghmaee, and M. Naghibzadeh. A load scheduler for SIP proxy servers: design, implementation and evaluation of a history weighted window approach. Int. J. Commun. Syst, 2015.

[12] P. Montoro, and E. Casilari. A Comparative Study of VoIP Standards with Asterisk. In Fourth International Conference on Digital Telecommunications, 1–6, 2009.

[13] A. Montazerolghaem, M. Hossein, A. Leon-Garcia, M. Naghibzadeh, and F. Tashtarian. A Load-Balanced Call Admission Controller for IMS Cloud Computing. IEEE Transactions on Network and Service Management, 2016.

[14] Mobicents, http://www.mobicents.org , accessed August 20, 2017.

[15] Y. Li, X. Tang, and W. Cai. Dynamic bin packing for on-demand cloud resource allocation. Parallel and Distributed Systems, IEEE Transactions on. 27(1):157-170, 2016.

[16] W. Song, Z. Xiao, Q. Chen, and H. Luo. Adaptive resource provisioning for the cloud using online bin packing. Computers, IEEE Transactions on. 63(11): 2647-2660, 2014.

[17] A. Wolke, B. Tsend-Ayush, C. Pfeiffer, and M. Bichler. More than bin packing: Dynamic resource allocation strategies in cloud data centers. Information Systems, 52, 83-95, 2015.

[18] A. M. Simionovici, A. A. Tantar, P. Bouvry, and L. Didelot. Predictive modeling in a voip system. Journal of Telecommunications and Information Technology, 2013.

[19] A. M. Simionovici, A. A. Tantar, P. Bouvry, A. Tchernykh, J. M. Cortés-Mendoza, L. Didelot. VoIP traffic modelling using Gaussian mixture models, Gaussian processes and interactive particle algorithms. In 2015 IEEE Globecom Workshops, 1-6, 2015.

[20] Y. Lu, J. Panneerselvam, L. Liu, and Y. Wu. RVLBPNN: A Workload Forecasting Model for Smart Cloud Computing, Scientific Programming, vol. 2016, Article ID 5635673, 2016.

[21] L. M. Campos, I.D. Scherson. Rate of change load balancing in distributed and parallel systems. Parallel Computing, 26(9), 1213-1230, 2000.

[22] D. Tsafrir, Y. Etsion, and D. Feitelson. "Backfilling using system-generated predictions rather than user runtime estimates". IEEE Transactions on Parallel and Distributed Systems 18(6): 789-803, 2007.

[23] A. Tchernykh, L. Lozano, U. Schwiegelshohn, P. Bouvry, J. E. Pecero, S. Nesmachnow, A. Yu. Drozdov. Online Bi-Objective Scheduling for IaaS Clouds with Ensuring Quality of Service. Journal of Grid Computing. Springer-Verlag, vol. 14, Issue 1, 5–22, 2016.

[24] A. Tchernykh, U. Schwiegelsohn, E.-g. Talbi, M. Babenko. Towards Understanding Uncertainty in Cloud Computing with risks of Confidentiality, Integrity, and Availability. Journal of Computational Science. Elsevier, 2016.

[25] M. Mao and M. Humphrey. A performance study on the vm startup time in the cloud. In Cloud Computing (CLOUD), IEEE 5th International Conference on, 423-430, 2012.

[26] K. Hoffman, http://blog.cloud66.com/ready-steady-go-the-speed-of-vm-creation-and-ssh-key-access-on-aws-digitalocean-linode-vexxhost-google-cloud-rackspace-and-microsoft-azure/, accessed August 20, 2017.

[27] S. K. Garg and R. Buyya. NetworkCloudSim: Modelling Parallel Applications in Cloud Simulations, Proceedings of the 4th IEEE/ACM International Conference on Utility and Cloud Computing (UCC 2011, IEEE CS Press, USA), December, 2011.

[28] Zitzler, E., Laumanns, M., and Bleuler, S. A tutorial on evolutionary multiobjective optimization. In Metaheuristics for multiobjective optimization (pp. 3-37). Springer, Berlin, Heidelberg, 2004.

[29] Zhang, G. P. Neural networks for classification: a survey. IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews), 30(4), 451-462, 2000.

Appendix A

Table 19. Mean degradations and ranking for strategies with and without *Rental-Threshold*.

Strategy	$\bar{b}$	$\bar{c}$	Mean	R $\bar{b}$	R $\bar{c}$	R mean	Rank
BFit	0.0578	0.2316	0.1447	27	30	7	7
BFit_05	0.1081	0.2342	0.1712	43	31	16	11
BFit_10	0.1238	0.3026	0.2132	50	36	23	18
BFit_15	0.1091	0.5263	0.3177	44	52	59	27
FFit	0.0493	0.2632	0.1562	16	34	13	4
FFit_05	0.0901	0.2763	0.1832	40	35	18	12
FFit_10	0.1195	0.4105	0.2650	49	43	38	23
FFit_15	0.1134	0.5500	0.3317	45	55	64	31
MaxFTFit	0.0643	1.0184	0.5414	30	78	99	38
MaxFTFit_05	0.0702	1.4816	0.7759	31	87	113	48
MaxFTFit_10	0.0562	1.5553	0.8057	26	88	114	44
MaxFTFit_15	0.0530	1.7184	0.8857	21	90	116	41
MidFTFit	0.1440	0.6158	0.3799	59	59	78	48
MidFTFit_05	0.1500	0.4763	0.3132	60	45	57	35
MidFTFit_10	0.1310	0.6000	0.3655	53	57	75	40
MidFTFit_15	0.1161	0.8658	0.4910	46	68	91	44
MinFTFit	0.5346	0.0132	0.2739	122	2	42	54
MinFTFit_05	0.3771	0.0711	0.2241	80	14	31	25
MinFTFit_10	0.2720	0.1763	0.2242	70	26	32	27
MinFTFit_15	0.1567	0.5395	0.3481	61	54	69	45
Rand	0.4029	0.0368	0.2199	88	8	28	27
Rand_05	0.2795	0.0605	0.1700	71	12	15	16
Rand_10	0.2054	0.0579	0.1317	68	11	5	14
Rand_15	0.1731	0.0579	0.1155	64	11	1	12
RR	**0.4222**	**0.0184**	**0.2203**	**97**	**4**	**29**	**32**
RR_05	0.2812	0.0579	0.1695	72	11	14	16
RR_10	0.2075	0.0868	0.1472	69	15	10	17
RR_15	0.1747	0.0868	0.1308	65	15	4	15
WFit	0.4611	0.0211	0.2411	108	5	35	43
WFit_05	0.2951	0.0711	0.1831	73	14	17	19
WFit_10	0.1894	0.1842	0.1868	67	27	19	25
WFit_15	0.1390	0.5053	0.3221	57	48	60	35

Table 20. Mean degradations and ranking for *Rate of Change* strategies.

Strategy	$\bar{b}$	$\bar{c}$	Mean	R $\bar{b}$	R $\bar{c}$	R mean	Rank
BFit_s10	0.0550	0.2211	0.1380	24	29	6	6
BFit_s20	0.0513	0.2474	0.1493	19	32	11	5
BFit_s30	0.0517	0.2026	0.1272	20	28	3	3
BFit_StUp	0.0411	0.6447	0.3429	8	60	67	9
FFit_s10	0.0425	0.2500	0.1462	9	33	9	1
FFit_s20	0.0427	0.2474	0.1451	10	32	8	1
FFit_s30	0.0464	0.2026	0.1245	15	28	2	2
FFit_StUp	0.0322	0.6158	0.3240	4	59	62	8
MaxFTFit_s10	0.0578	0.8789	0.4684	27	70	90	28
MaxFTFit_s20	0.0538	0.9868	0.5203	23	75	94	29
MaxFTFit_s30	0.0620	0.9789	0.5205	29	74	95	33
MaxFTFit_StUp	0.0443	1.3763	0.7103	13	85	111	29
MidFTFit_s10	0.1373	0.2500	0.1936	56	33	20	21
MidFTFit_s20	0.1362	0.5132	0.3247	55	51	63	36
MidFTFit_s30	0.1341	0.1737	0.1539	54	25	12	14
MidFTFit_StUp	0.1174	0.6632	0.3903	47	61	80	38
MinFTFit_s10	0.5253	0.0132	0.2692	121	2	41	53
MinFTFit_s20	0.5032	0.0289	0.2661	117	7	40	54
MinFTFit_s30	0.5118	0.0000	0.2559	118	1	36	49
MinFTFit_StUp	0.4655	0.3553	0.4104	111	41	82	67
Rand_s10	0.4053	0.0237	0.2145	89	6	25	26
Rand_s20	0.3898	0.0368	0.2133	84	8	24	23
Rand_s30	0.3918	0.0000	0.1959	85	1	21	18
Rand_StUp	0.3598	0.3763	0.3681	77	42	76	49
RR_s10	0.4192	0.0158	0.2175	95	3	27	29
RR_s20	0.4057	0.0289	0.2173	92	7	26	30
RR_s30	0.4056	0.0000	0.2028	91	1	22	23
RR_StUp	0.3727	0.3763	0.3745	78	42	77	50
WFit_s10	0.4543	0.0132	0.2338	106	2	33	38
WFit_s20	0.4438	0.0289	0.2364	102	7	34	39
WFit_s30	0.4455	0.0000	0.2227	105	1	30	36
WFit_StUp	0.4128	0.3763	0.3946	94	42	81	62

Table 21. Mean degradations and ranking for *Neural Network* strategies.

Strategy	$\bar{b}$	$\bar{c}$	Mean	R $\bar{b}$	R $\bar{c}$	R mean	Rank
BFit_s10	0.0534	1.0079	0.5307	22	77	97	30
BFit_s20	0.0455	1.0868	0.5662	14	79	101	24
BFit_s30	0.0407	1.1579	0.5993	7	81	103	20
BFit_StUp	0.0084	3.9974	2.0029	2	99	126	32
FFit_s10	0.0434	0.5553	0.2993	12	56	52	9
FFit_s20	0.0377	0.6842	0.3609	6	62	73	9
FFit_s30	0.0322	0.8737	0.4530	4	69	87	10
FFit_StUp	0.0000	4.1342	2.0671	1	100	127	32
MaxFTFit_s10	0.0611	1.5579	0.8095	28	89	115	47
MaxFTFit_s20	0.0559	1.7474	0.9016	25	91	117	46
MaxFTFit_s30	0.0494	1.9237	0.9866	17	93	119	40
MaxFTFit_StUp	0.0157	4.4158	2.2158	3	101	128	34
MidFTFit_s10	0.1401	0.7421	0.4411	58	63	85	51
MidFTFit_s20	0.1292	0.9211	0.5251	51	73	96	54
MidFTFit_s30	0.1180	0.9921	0.5551	48	76	100	54
MidFTFit_StUp	0.0842	3.3842	1.7342	38	98	125	62
MinFTFit_s10	0.5361	0.1368	0.3365	123	20	65	64
MinFTFit_s20	0.5149	0.3184	0.4167	119	38	83	68
MinFTFit_s30	0.4825	0.5053	0.4939	113	48	93	69
MinFTFit_StUp	0.4027	2.7211	1.5619	87	96	124	73
Rand_s10	0.4066	0.1605	0.2836	93	24	45	47
Rand_s20	0.3888	0.3421	0.3655	83	40	74	53
Rand_s30	0.3740	0.5395	0.4567	79	54	88	61
Rand_StUp	0.3057	2.7684	1.5371	74	97	123	71
RR_s10	0.4247	0.1342	0.2794	98	19	44	47
RR_s20	0.4054	0.3158	0.3606	90	37	72	57
RR_s30	0.3866	0.4974	0.4420	82	47	86	58
RR_StUp	0.3183	2.7211	1.5197	75	96	121	71
WFit_s10	0.4643	0.1395	0.3019	110	21	53	60

Table 22. Mean degradations and ranking for *History based* strategies.

Strategy	$\bar{b}$	$\bar{c}$	Mean	R $\bar{b}$	R $\bar{c}$	R mean	Rank
BFit_s10	0.0905	0.6132	0.3519	41	58	70	30
BFit_s20	0.0800	0.9026	0.4913	35	72	92	37
BFit_s30	0.0737	0.8079	0.4408	33	64	84	28
BFit_StUp	0.0429	1.8079	0.9254	11	92	118	33
FFit_s10	0.0826	0.5105	0.2966	37	50	51	19
FFit_s20	0.0746	0.5395	0.3070	34	54	54	20
FFit_s30	0.0712	0.4842	0.2777	32	46	43	13
FFit_StUp	0.0354	1.4711	0.7532	5	86	112	22
MaxFTFit_s10	0.0965	1.2447	0.6706	42	83	108	55
MaxFTFit_s20	0.0888	1.1947	0.6418	39	82	107	51
MaxFTFit_s30	0.0804	1.1079	0.5942	36	80	102	46
MaxFTFit_StUp	0.0507	1.9342	0.9924	18	94	120	42
MidFTFit_s10	0.1840	0.4632	0.3236	66	44	61	40
MidFTFit_s20	0.1713	0.5368	0.3541	63	53	71	46
MidFTFit_s30	0.1622	0.9000	0.5311	62	71	98	61
MidFTFit_StUp	0.1293	1.2474	0.6883	52	84	110	62
MinFTFit_s10	0.5787	0.0526	0.3156	126	10	58	62
MinFTFit_s20	0.5654	0.1289	0.3472	125	18	68	64
MinFTFit_s30	0.5455	0.1342	0.3398	124	19	66	64
MinFTFit_StUp	0.4914	0.8553	0.6733	115	66	109	72
Rand_s10	0.4618	0.0684	0.2651	109	13	39	52
Rand_s20	0.4454	0.1447	0.2951	104	22	50	56
Rand_s30	0.4299	0.1500	0.2900	99	23	49	52
Rand_StUp	0.3826	0.8605	0.6216	81	67	104	66
RR_s10	0.4753	0.0421	0.2587	112	9	37	51
RR_s20	0.4608	0.1132	0.2870	107	16	47	53
RR_s30	0.4452	0.1289	0.2871	103	18	48	51
RR_StUp	0.3924	0.8553	0.6238	86	66	105	67
WFit_s10	0.5156	0.0526	0.2841	120	10	46	59

WFit_s20	0.4429	0.3237	0.3833	101	39	79	63	WFit_s20	0.4934	0.1263	0.3098	116	17	56	61
WFit_s30	0.4205	0.5079	0.4642	96	49	89	65	WFit_s30	0.4848	0.1342	0.3095	114	19	55	61
WFit_StUp	0.3480	2.7184	1.5332	76	95	122	71	WFit_StUp	0.4301	0.8500	0.6401	100	65	106	70

Andrei Tchernykh received his Ph.D. degree from Institute of Precise Mechanics and Computer Technology of the Russian Academy of Sciences, Russia in 1986. He gained industrial experience as supercomputer design team leader in Advance Technical Products Corp, and Supercomputer Design Department of Electro-Mechanical Enterprise, Russian leaders in HPC design and development. He is holding a full professor position in Computer Science Department at CICESE Research Center, Ensenada, Baja California, Mexico, and a head of Parallel Computing Laboratory. He is a member of the National System of Researchers of Mexico (SNI), Level II, and a founding member of the Mexican Supercomputer Society. He has published more than 200 papers in refereed journals and conferences, and served as a TPC member and general co-chair of more than 240 professional peer reviewed conferences. He was invited as a visiting researcher at prestigious universities and research centers. He leads a number of research projects and grants in different countries. He has served as a member of the editorial boards and guest editor of several scientific journals. His main interests include resource optimization technique, adaptive resource provisioning, multi-objective optimization, computational intelligence, incomplete information processing, cloud computing and security.

Jorge M. Cortés-Mendoza received his Bachelor's degree in Computer Sciences from the Autonomous University of Puebla (Benemérita Universidad Autónoma de Puebla, México) in July 2008, and his Master's degree in Computer Science from CICESE Research Center in March 2011. Since 2013, he is working on distributed computing, cloud computing, load balancing and scheduling.

Igor Bychkov received the Dr. degree from Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences (ISDCT SB RAS) in 2003. He is Academician of RAS, professor, Ph.D., director of the Matrosov Institute for System Dynamics and Control Theory of SB RAS, scientific leader of the Irkutsk Scientific Center of SB RAS. He is a member of a number scientific and expert councils, editorial boards of scientific journals. He is an expert for the Russian Foundation for Basic Research, Russian Scientific Foundation, Russian Academy of Sciences. He leads a number of national and international research projects. His main interests include artificial intelligence, geoinformation systems, WEB-technologies, systems of intelligent data analysis, mathematical modelling, cloud computing.

Alexander Feoktistov received the Ph.D. degree from Matrosov Institute for System Dynamics and Control Theory of Siberian Branch of the Russian Academy of Sciences (ISDCT SB RAS) in 2000. He is currently a senior research officer in Laboratory of Parallel and Distributed Computing Systems of ISDCT SB RAS, and an associate professor in Graduate school in ISDCT SB RAS. He leads a number of national research projects. His main interests include computational models, distributed computing, multi-agent technologies and simulation model

Loïc Didelot has been committed in several startups in Luxembourg as Project Manager and Team Manager. He invented new solutions to the changing demand for telephony, and founded MIXvoip in 2008. Today, Loïc Didelot continues to develop new solutions to increase the flexibility of VoIP, and audacious convergent mobile applications.

Pascal Bouvry obtained his Ph.D. degree in Computer Science with great distinction at the University of Grenoble (INPG), France in 1994. He performed post-doctoral researches at CWI in Amsterdam. Dr Bouvry gained industrial experience as manager of the technology consultant team for FICS (belonging to S1 corp.) a world leader in electronic financial services. Dr. Bouvry is currently professor at the University of Luxembourg, special advisor to the University President in charge regarding High Performance Computing, heading the PCOG (Parallel Computing and Optimization Group), directing the doctoral program DP-CSCE, and certificate SmartICT for Business innovation. Pascal Bouvry is also Faculty of the Interdisciplinary Center of Security, Reliability, and active in various scientific committees and technical workgroups (IEEE CIS Cloud Computing vice-chair, IEEE TCSC GreenIT steering committee, ERCIM WG, ANR, COST TIST, etc.). Pascal Bouvry is a member of the editorial boards of IEEE Transactions on Sustainable Computing, IEEE Cloud Computing Magazine, Springer journal on Communications and Sustainable Computing, and Elsevier journal in Swarm and Evolutionary Computation.

Gleb Radchenko graduated from South Ural State University (SUSU, Chelyabinsk, Russia) in applied mathematics and computer science in 2006. He obtained his PhD from Moscow State University, Russia in 2010. Since 2010, he worked in SUSU in the department of system programming. He currently holds a position of a director of School of Electrical Engineering and Computer Science of SUSU. He's sphere of interest is focused on such topics as high-performance systems, cloud computing, distributed computing and scientific workflows.

Kirill Borodulin graduated from South Ural State University (SUSU, Chelyabinsk, Russia) in 2012, where he obtained his Master Degree in applied mathematics and computer science. Now, he is a 3-year PhD student in computer science at the SUSU. Since 2008 he worked in SUSU in the Supercomputer Simulation Laboratory. He currently holds a position of a director of the supercomputer center. He's sphere of interest is focused on the following topics: high-performance systems administration, parallel programming, and cloud computing.