2017 IEEE International Parallel and Distributed Processing Symposium Workshops

Load-Aware Strategies for Cloud-based VoIP Optimization with VM Startup Prediction

Jorge M. Cortés-Mendoza[1],
Andrei Tchernykh[2]
Computer Science Department
CICESE Research Center Ensenada
Baja California, México.
[1] *jcortes@cicese.edu.mx*
[2] *chernykh@cicese.mx*

Igor Bychkov[3],
Alexander Feoktistov[4]
Matrosov Institute for System
Dynamics and Control Theory of
SB RAS, Irkutsk, Russia.
[3] *bychkov@icc.ru,*
[4] *agf65@yandex.ru,*

Pascal Bouvry[5]
Computer Science and
Communications Research Unit,
University of Luxembourg.
Luxembourg
[5] *Pascal.Bouvry@uni.lu*

Loic Didelot[6]
MIXvoip S.A.
Sandweiler, Luxembourg.
[6]*ldidelot@mixvoip.com*

***Abstract*— In this paper, we address cloud VoIP scheduling strategies to provide appropriate levels of quality of service to users, and cost to VoIP service providers. This bi-objective focus is reasonable and representative for real installations and applications. We conduct comprehensive simulation on real data of twenty three on-line non-clairvoyant scheduling strategies with fixed threshold of utilization to request VMs, and twenty strategies with dynamic prediction of the load. We show that our load-aware with predictions strategies outperform the known ones providing suitable quality of service and lower cost. The robustness of these strategies is also analyzed varying VM startup time delays to deal with realistic VoIP cloud environments.**

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

I. Introduction

Cloud computing is growing as feasible business model. Many companies have been adopted it as an innovative and profitable solution [23, 24]. Cost savings and scalability are the most important reasons due to business are moving to the cloud, both factors allow them to increase their earnings and provide an adequate quality of services to the users. Some examples of business migration are: TV [23], radio [24], telephone [17], etc.

Voice over IP (VoIP) is a fast growing technology in cloud computing. It is considered as a long-term service roadmap, offering higher flexibility and more features than traditional telephony (PSTN) infrastructure.

The main benefits of this technology, over traditional telephony model and VoIP, are cost effectiveness, flexibility, scalability, extended service variety, etc. It can cope with different workloads, and dynamically adapt resources in response to demand.

Asterisk [1] is the backbone in 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 a complete 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. [2].

Our previous works [3, 18] focused on formulation of the scheduling problem of VoIP services in cloud environments, and proposed a new model for bi-objective optimization. Our most recent study [21] analyzed scheduling strategies focusing on VM time provisioning, an important factor that affects time sensitive and auto-scaling mechanisms crucial in CVoIP.

To this end, we considered the special case of the on-line non-clairvoyant dynamic bin packing problem, and discussed solutions for provider cost minimization, and quality of service optimization. We proposed call allocation strategies that use information about VMs utilization, VMs rental time, and VM start-up time delay (StUp).

In this paper, we extend the previous study and consider dynamic prediction of the amount of VMs needed to provide the VoIP service and reduce the number of active VMs.

We propose and evaluate two mechanisms to request new VMs: when the current and predicted VMs utilization is over the fixed threshold. Our fast dynamic load prediction is calculated based on the speed of utilization increment.

We analyze twenty three on-line non-clairvoyant scheduling strategies, in the first group, and twenty strategies

978-0-7695-6149-3/17 $31.00 © 2017 IEEE
DOI 10.1109/IPDPSW.2017.73

with dynamic load prediction. These bi-objective scheduling strategies consider billing hours and voice quality optimization criteria.

We conduct comprehensive simulation analysis on real workload of the MIXvoip company [17], and show that our strategies improve drawbacks of known strategies. They increase voice quality and reduce amount of calls in waiting queue.

The paper is structured as follows. The next section presents VoIP service considering underlined infrastructure and software. Section III briefly reviews related works on bin packing, call load balancing, and load estimation. Section IV provides the problem definition and proposed model. Section V describes VoIP call allocation strategies. Section VI discusses our experimental setup, workload and studied scenario. Section VII presents experimental analysis of the provider cost and quality of service. Section VIII highlights the final conclusions of the paper and future work.

II. 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 problem of traditional VoIP systems. For instance, availability of the service for any number of user 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 systems to two major problems: overprovisioning and overrunning 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 overprovisioning. The deployment of the virtual infrastructure is easy to implement, faster to provisioning, and integrate services that are dynamically scalable. Further, it adds new features and capabilities for users (data transfer availability, integrity, and security).

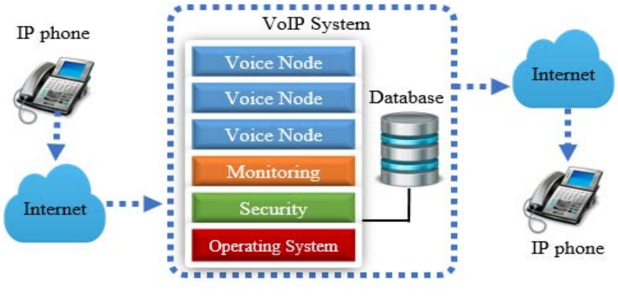

Figure 1. VoIP architecture [18].

Voice Nodes (VNs) are the core part of the CVoIP telephony system (Fig. 1).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 the other end-device. After the call is finished, call details are stored in the Call-Detail-Record (e.g. client id, destination, prefix, duration call).

In CVoIP model proposed in [18], the VNs are operated by VMs. To optimize the overall system performance and reduce provider cost, the VM utilization has to be high. However, it reduces quality of the calls (Section II.B). Hence, load of the VoIP servers should be reduced to guarantee the QoS. On the other hand, the idle time increases the useless expenses of the VoIP provider.

A. *Infrastructure*

Mixvoip [17] 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 the SN 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 [1]. It is a framework, under 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 voice nodes that run and handle calls. Each node has Asterisk running process with unique IP address that is used by end users to connect inside and outside the network.

B. *Quality of service*

The VoIP services have stricter constraints and sensitive factors. Call processing and call delivery are two main issues which determine the QoS (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 in call processing.

The quality of voice is subjectively perceived by the listener. A common 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 provides a certain quality of speech only if processor utilization is low enough. Theoretically, processor utilization of 100% provides the best expected performance. However, Eleftheriou [6] 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 increase the used bandwidth but it is less significant than the same codec adds to CPU utilization.

Montazerolghaem et al. [7] reported that the consumed bandwidth of 6,500 calls per second not exceed 100 Mbps, and 10,000 calls not reach 400 Mbps.

Table I 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 in 100 Mbps connection is between 6,000 and 25,000 depending on codec of the calls.

TABLE I. 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 MBs connection. Hence, the call processing is the key feature to guarantee the QoS.

In [3], authors proposed to limit processor utilization in order to ensure QoS.

C. CPU utilization

Calls have different impact on the processor utilization [3] 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 II shows processor utilization for call without transcoding presented by Montoro et al. 2009 [8].

TABLE II. 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 are analyzed for VoIP [6] considering calls amount, utilization, power consumption, database messaging, registration and call performance. The author concludes that CPU can process from 70 to 500 calls with 100% of utilization.

D. 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 rented (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, two criteria are considered: the billing hours for VMs to provide a service, and voice quality reduction.

III. RELATED WORK

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

A. Call load balancing

The main objective of call load balancing is to reduce the infrastructure cost and guarantee that service will be delivery 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) [9] 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 [22] is a SIP-based proxy for VoIP infrastructure with multiple ingress proxy servers. It allows to avoid 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.

B. Bin packing

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

Variable Item Size Bin Packing (VIS-BP) [9] is a technique to assign data center resources using live VM migration considering a relaxed on-line bin packing model. The main goal is to minimize the amount of wasted space by restricting the combinations of tasks in a bin. The authors evaluate the effectiveness of the algorithm, and provide a theoretical proof for the number of used servers and VM migrations. Its multi-dimensional version considers a mix of CPU and network.

Song et al. [10] 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 in this stage. The reallocation controller triggers VM migration, when under loaded 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.

Wolke et al. [11] analyze the competitive ratio for several versions of Best Fit and First Fit algorithms. They study the request dispatching in cloud gaming modelled as a variant of Dynamic Bin Packing problem. Gaming requests must be dispatched with enough resources of CPU and GPU in order to provide a good user experience. 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.

C. Load estimation

Prediction techniques to anticipate the incoming traffic (calls for VoIP) are applied for an efficient distribution of the load in the system. The goals of traffic prediction on cloud computing is to minimize the infrastructure costs and improve the QoS to the end user.

Simionovici et al. [16], study and compare different prediction models for real VoIP environment. Interactive Particle Systems (IPS), Gaussian Mixture Model (GMM), and Gaussian Process (GP) are used to predict the incoming voice traffic during time frames. 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.

Rate of change (RoC) [20] is a strategy for load balancing tasks in 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 own load; migration does not depend on the load of other *Bal*s. The migration decision depends on current load, load changes in the time interval (rate of load change), and current load balancing parameters. Difference in Load (DL) is used as an estimation of load, its prediction for the next time slot, and detection of the need for load balancing process.

IV. MODEL

The model follows the our previous works [3, 18, 21], 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 request time $st_j \geq 0$, 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 SNC s is provided through public Internet.

We consider n independent calls $J_1, J_2, \dots, J_n$ that must be scheduled on 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 rented VMs and their rental time. We denote the number of Billing Hours in SNC_i by:

$$\bar{b}_i = \int_{t=0}^{C_{max}} k_i(t) \cdot m_i \, dt. \tag{1}$$

and run in all SNC 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:

$$\overline{q_\imath} = \sum_{j=1}^{\bar{b}_i} \sum_{t=ini_j}^{end_j} \gamma(vmu_j(t)) \tag{3}$$

where ini_j and end_j define the rental time of the VM_j and the penalization function $\gamma(\alpha) = ((\alpha - 0.7) * \overline{3.33})^2$ when α>0.7 and 0 otherwise. Total Quality Reduction is defined by:

$$\bar{q} = \sum_{i=1}^{m} \overline{q_\imath} \tag{4}$$

Moreover, we analyze the number of calls waiting on the queue, it occurs when the VMs cannot process the arriving calls, so the calls wait on the queue for a VM. The amount of Calls to Queue 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 $\delta(\alpha)$ is 1 if $\alpha > 0$ and 0 otherwise.

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

This approach allows to adapt 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 [25].

To compare strategies, we perform an analysis based on the degradation methodology proposed in [13], and applied for scheduling in [3, 18, 19]. It shows how the metric generated by our algorithms gets closer to the best found solution as:

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

V. 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. The classic bin packing concerns placing items of arbitrary height into a minimum number of bins with fixed capacity (of one-dimensional space) efficiently. 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 must 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 rate, etc.

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

We consider a scenario where the bin size is equals to 0.7 that corresponds to 70% of VM utilization. The scheduler has no information of the calls arrival rate, and it takes decisions depending on the current system state.

A. Call allocation

In this paper, we consider call allocation taking into account VM startup time delay. A new VM is requested when the utilization is over or predicted over the threshold. During VM StUp, old VM continues call processing with utilization more than threshold reducing QoS. The worst case appears when the current VM does not have enough resources to process arriving calls. In this case, the system puts the calls into a queue, waiting for available resources.

Call allocation strategies are grouped by the type and amount of information used for allocation: (1) knowledge-free (KF), with no information about applications and resources; (2) utilization-aware (UA) with CPU utilization information, (3) time-aware (TA) with VM rental time information (beginning and completion), and (4) load-aware (LA) with VM load information in two time frames.

Table III summarizes the call allocation strategies that request new VM when arriving call exceeds the threshold of utilization in all VMs.

Table IV shows the call allocation strategies with load estimation. Each time interval, the strategy makes an estimation of utilization for next time interval and if it overpasses the threshold current VMs, it requests new VM.

Algorithm 1 describes the BFit_*xx* strategy, where Voice Nodes (VNs) are separated in two lists: Admissible Voice Node List (AVNL) and no AVNL list (nAVNL). AVNL list contains the VNs that their finish time is not less than in *xx* minutes (time-aware). 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.

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

TABLE III. CALL ALLOCATION STRATEGIES

		Description
KF	Rand	Allocates job j to VM randomly using a uniform distribution.
	RR	Allocates job j to VM using a Round Robin algorithm.
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.
TA	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.
	Rand_05	Allocates job j to VM that finishes not less than in 5, 10, 15 minutes using the Rand, and RR strategies.
	Rand_10	
	Rand_15	
	RR_05	
	RR_10	
	RR_15	
UA + TA	BFit_05	Allocates job j to VM that finishes not less than in 5, 10, and 15 minutes using the Bfit, FFit, and WFit strategies.
	BFit_10	
	BFit_15	
	FFit_05	
	FFit_10	
	FFit_15	
	WFit_05	
	WFit_10	
	WFit_15	

TABLE IV. CALL ALLOCATION STRATEGIES WITH PREDICTION

		Description
LA	Rand_stUp	Allocates job j to VM using the Rand, and RR strategies. They use intervals of 10, 20, 30 and stUp seconds to estimate future load
	Rand_s10	
	Rand_s20	
	Rand_s30	
	RR_stUp	
	RR_s10	
	RR_s20	
	RR_s30	
UA +LA	BFit_stUp	Allocates job j to VM using BFit, FFit, and WFit strategies. They use intervals of 10, 20, 30 and stUp seconds to estimate future load
	BFit_s10	
	BFit_s20	
	BFit_s30	
	FFit_stUp	
	FFit_s10	
	FFit_s20	
	FFit_s30	
	WFit_stUp	
	WFit_s10	
	WFit_s20	
	WFit_s30	

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 into the call queue waiting for a new 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 startup time reduces the QoS, so algorithm looks first on nAVNL list. In the worst case, algorithms start a new VN.

Algorithm 1. Best Fit TA (BFit_xx)

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

vnIndex ← -1
Create AVNL and nAVNL lists with TA time.
Sort AVNL by utilization on decreasing order.
Sort nAVNL by utilization on decreasing order.
vnIndex ← Best_Fit(AVNL, 0.7, call)
   if vnIndex < 0 then
      vnIndex ←Best_Fit(nAVNL, 0.7, 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
   Send call to call_queue
   Start a new node voice
else
     Insert call into VN with index vnIndex
 Endif
```

B. *Load-aware strategies*

Rate of Change (RoC-LB) [20] is a dynamic distributed load balancing algorithm, it achieves the goal of minimizing processor idling times without incurring into unacceptably high load overheads. Resources calculate the change in their load between two sample intervals. The Sampling Interval (Si) is an adaptive parameter, and its length may vary. Each resource asynchronous calculates the Difference in Load (Dl), and use it as estimation on load for the next Si, this estimation allows to allocate and reallocate jobs more efficiently. A finer sampling allows to improve the balance the system, but it increases the overhead.

We use the concept of Dl as a mechanism to predict requests for new VMs, which can be provided after StUp time. Dl (utilization amount of arriving calls) are used to estimate the number of VMs after Si. 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 sample interval Si=$[t - Si, t]$ is defined by:

$$\Delta_i(t) = (u_i(t) - u_i(t - Si))/k_i(t) \qquad (7)$$

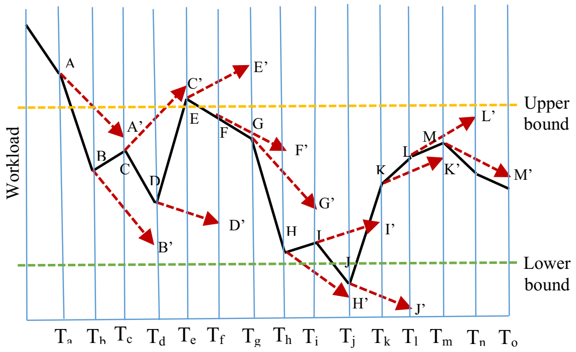

Figure 2. Calls prediction mechanism.

Fig. 2 shows Dl changes scenario. Solid line represents real workload and dashed lines are estimated workloads. If the load is larger than Upper bound (Ub) then the broker request for a new VM. If the predicted load is less than Lower bound (Lb) the broker can reduce the amount of running VMs. Ub and Lb are adjustable parameters than depend on the number of running VMs and the utilization threshold.

At time T_c, the broker immediately initiates a new VM based on predicted future load. Estimation J' on T_j is under Lb but broker cannot initiates a reduction process because the request at T_h is not finished yet. New request can be generated only after T_h.

VI. Experimental Setup

We perform experiments using standard trace based simulator CloudSim [15] extended by our algorithms supporting dynamic calls arrival, VM startup delays, and statistical analysis.

A. *Workload*

We use traces of real VoIP service [16] that 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.

Number of calls per day and call duration are presented in Table V and Table VI. 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.

TABLE V. 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 VI. 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

VII. 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 time-aware versions, when strategies take into account exact time of VM provisioning. Then, we study the advantages of load-aware versions when strategies consider loads of VMs and their variations. Finally, we perform a comparative study of the best found strategies. In order to evaluate their robustness, we incorporate to all strategies eight StUp delays: 0, 45, 90, 135, 180, 225, 270, and 315 seconds as test cases [4, 5].

In general, we do not analyze quality reduction ($\bar{q}$), see (4), due to the difference between the no QoS degradation and the worst strategy is about 2.813×10^{-3}. In this case, calls to queue ($\bar{c}$) is a better representation of QoS reduction.

A. *Time-aware strategies*

We evaluate twenty three strategies: BFit, FFit, MaxFTFit, MidFTFit, MinFTFit, Rand, RR and WFit, and three time-aware version of BFit, FFit, Rand, RR and WFit.

Fig. 3 shows the Billing Hours ($\bar{b}$), see (2), degradation versus StUps. We observe that strategies with better performance are BFit and FFit, and the worst strategies are MinFTFit and WFit.

We see that our scheduling strategies tend to be robust with respect to $\bar{b}$, the StUp does not affect considerably $\bar{b}$ for all strategies.

The average Calls to Queue ($\bar{c}$), see (5), is about 4 for MaxFTFit in a day during 30 days (the worst strategy) with StUp equals to 315 sec (Fig. 4).

Table VII shows the average reduction on $\bar{b}$, time-aware technique can benefit strategies with high fluctuation of $\bar{b}$, like Rand, RR and WFit, and affects other with low fluctuation of $\bar{b}$ (BFit and FFit).

Table VIII presents the average $\bar{c}$ of the strategies. We see that BFit_xx and FFit_xx increase $\bar{b}$ and average $\bar{c}$, while time-aware strategies reduce $\bar{b}$ of both strategies.

TABLE VII. AVERAGE BILLING HOURS REDUCTION (%)

xx Strategy	05 min	10 min	15 min.
BFit	- 4.69	- 6.27	- 4.8
FFit	- 4.79	- 6.4	- 5.18
Rand	8.88	14.21	16.61
RR	9.97	15.17	17.67
WFit	11.3	18.56	21.98

TABLE VIII. AVERAGE CALLS TO QUEUE PER DAY

xx Strategy	No TA	05 min	10 min	15 min.
BFit	1.95	1.95	2.06	2.42
FFit	2	2.02	2.23	2.45
Rand	1.64	1.68	1.68	1.72
RR	1.61	1.68	1.72	1.72
WFit	1.62	1.7	1.88	2.38

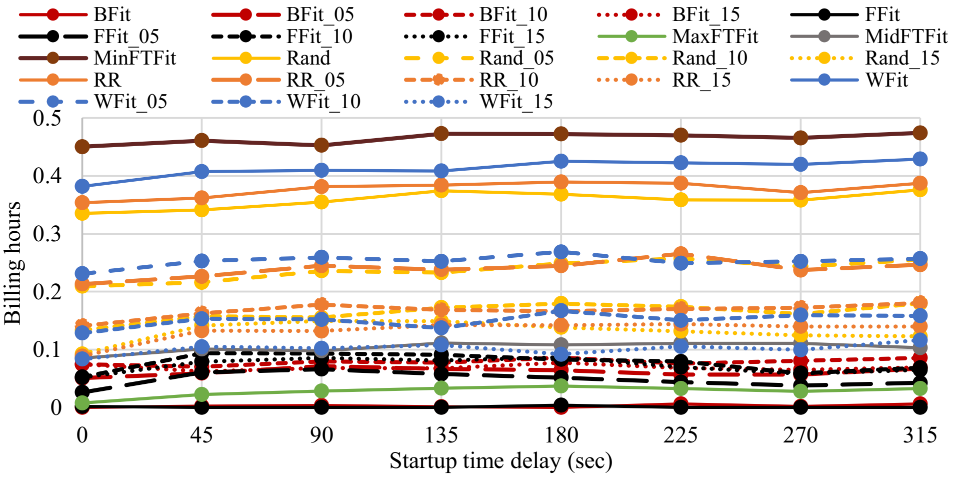

Figure 3. Billing hour degradation.

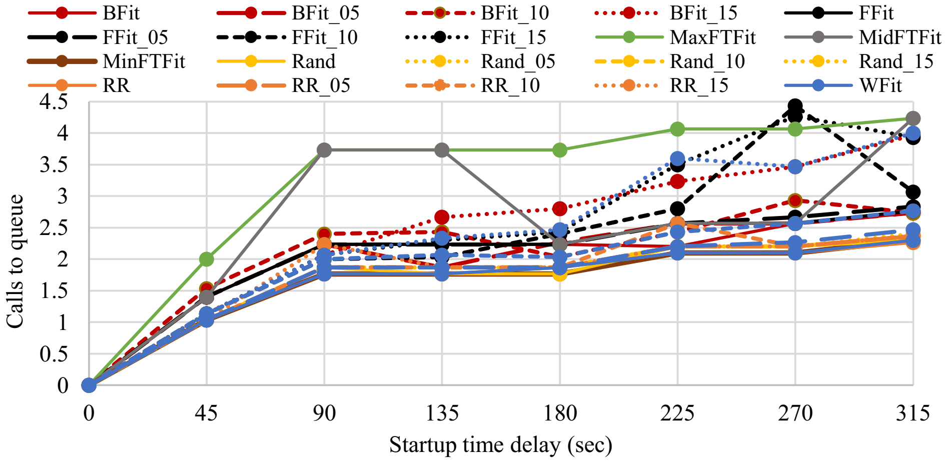

Figure 4. Average calls to queue.

B. *Load-aware strategies*

We evaluate five strategies: BFit, FFit, Rand, RR and WFit, with four prediction, see (7), time intervals each 10, 20, 30, and StUp (time interval is equal to startup time delay), in total twenty strategies.

Fig. 5 shows the Billing Hours ($\bar{b}$) degradation versus StUps. We observe that strategies with better performance are BFit_stUp and FFit_stUp, and the worst strategies are WFit_s10, WFit_s20 and WFit_s30.

Similar to time-aware strategies, load-aware strategies tend to be robust with respect to $\bar{b}$. The StUp does not affect considerably $\bar{b}$ (for all strategies). The average Calls to Queue ($\bar{c}$) is about 4 per day for the worst strategy BFit_stUp with StUp equals to 315 sec. (Fig. 6). The best strategy WFit_s30, $\bar{c}$ varies between 0.9 and 2.26 calls per day.

C. *Bi-objective Analysis*

In multi-objective analysis, the problem can be simplified to a single objective problem 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 (6).

First, we present the analysis of rented VMs ($\bar{b}$) and the number of call to queue ($\bar{c}$) separately. Then, we find the strategy that generates the best compromise between them.

In Table IX, we present the average degradation of $\bar{b}$, $\bar{c}$ and their means. The last three columns of the table contain the ranking of each strategy regarding to the provider cost, quality, and their means. Rank $\bar{b}$ is based on the billing hours degradation. Rank $\bar{c}$ refers to the position in relation to the calls to queue. Rank is the position based on the averaging two ranking.

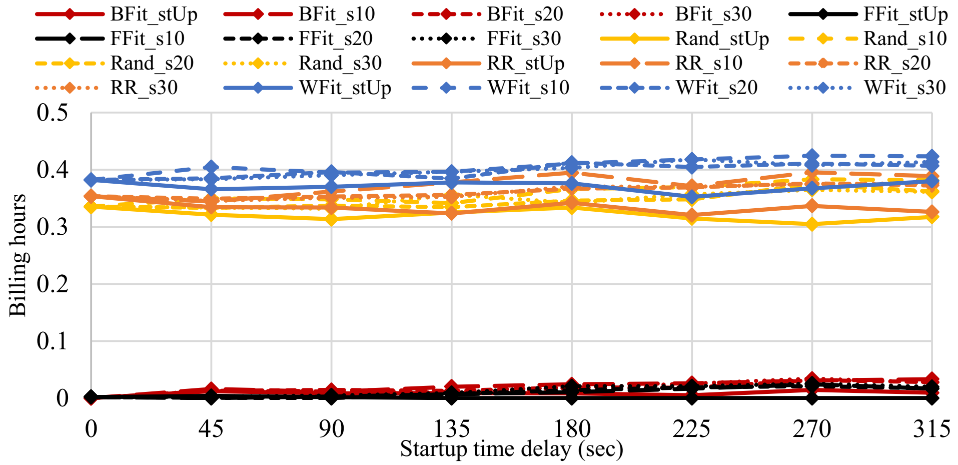

Figure 5. Billing hours degradation.

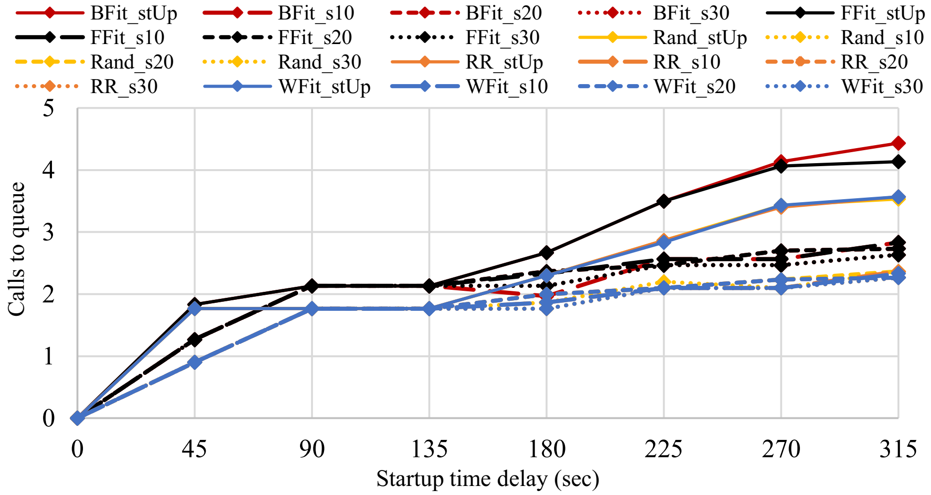

Figure 6. Average calls to queue.

We see that the best strategy for $\bar{b}$ is FFit_stUp (Fig. 7) that allocates calls based on best fit strategy with perdition time similar to VM startup time, where we put the call into the first VM. However, it is the second worst strategy for $\bar{c}$. It tends to increase utilization, and reduce quality.

The best strategy for $\bar{c}$ is Rand_15 (Fig. 8), where we put the call randomly into the VM. It tends to underutilize VMs reducing $\bar{c}$, but increases $\bar{b}$.

A good compromise are FFit_s30 and BFit_s30 strategies that allocate the call to VM using FFit and BFit respectably, and prediction of the load.

VIII. CONCLUSION

In this paper, we formulate and study scheduling problems addressing cloud-based VoIP load-aware scheduling strategies with VM startup prediction. We use bi-objective model with provider cost, contributed by billing hours for used VMs, and quality of service, affected by call processing, optimization criteria.

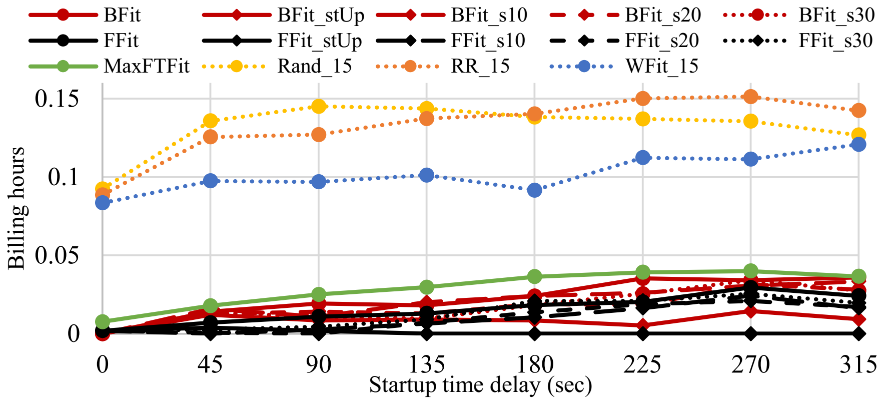

Figure 7. Billing hours degradation.

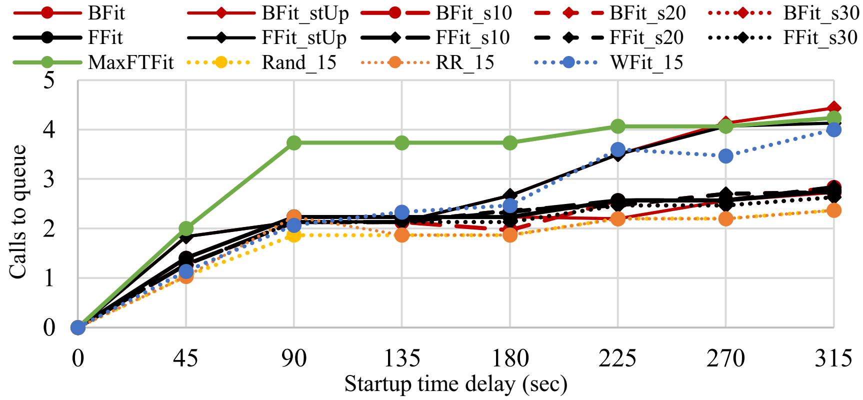

Figure 8. Average calls to queue.

TABLE IX. MEAN DEGRADATIONS AND RANKING

Strategy	Deg $\bar{b}$	Deg $\bar{c}$	Mean	Rank $\bar{b}$	Rank $\bar{c}$	Rank
BFit	0.0214	0.1642	0.0928	10	6	7
BFit_stUp	0.0073	0.5547	0.2810	2	13	6
BFit_s10	0.0190	0.1542	0.0866	9	5	5
BFit_s20	0.0160	0.1791	0.0975	7	7	5
BFit_s30	0.0163	0.1368	0.0766	8	3	**2**
FFit	0.0145	0.1940	0.1043	6	10	7
FFit_stUp	0.0000	0.5274	0.2637	1	12	4
FFit_s10	0.0087	0.1816	0.0952	3	9	3
FFit_s20	0.0090	0.1791	0.0940	4	8	3
FFit_s30	0.0120	0.1368	0.0744	5	4	**1**
MaxFTFit	0.0279	0.9080	0.4679	11	14	9
Rand_15	0.1306	0.0000	0.0653	13	1	5
RR_15	0.1315	0.0274	0.0794	14	2	7
WFit_15	0.1007	0.4229	0.2618	12	11	8

The analysis is based on real data collected during one month of the MIXvoip company service to deal with realistic VoIP cloud environments. We show that our strategies have a high quality of service putting 0.1% of calls to the waiting queue total during one month with 0.28% of quality reduction in the worst case. We show that the proposed strategies with dynamic prediction outperform known ones in terms of quality of service and provider cost including those currently in use.

We also evaluate their robustness in face of uncertainty of VM startup time delays variations, and show that they have the low variation in billing hours even with high dispersion of VM startup time delays. The evaluations demonstrate their potential benefits and stability with respect to handling call arrival rates and startup time delays variation. However, further study is required to assess their actual performance and effectiveness in a real domain. This will be the subject of future work.

Acknowledgment

The work is partially supported by RFBR, project no. 16-07-00931, Council for Grants of the President of Russian Federation, project NSh-8081.2016.9, and CONACYT, México, grant no. 178415.

References

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

[2] 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.

[3] 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.

[4] 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.

[5] K. Razavi, L. Razorea, and T. Kielmann. Reducing vm startup time and storage costs by vm image content consolidation. In Euro-Par 2013: Parallel Processing Workshops, 75-84, Springer Berlin Heidelberg, 2013.

[6] 3CX Phone System and ATOM N270 Processor Benchmarking. http://www.3cx.com/blog/voip-howto/atom-processor-n270-benchmarking, accessed September 20, 2016.

[7] 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.

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

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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 September 20, 2016.

[15] CloudSim: A framework for modeling and simulation of Cloud Computing infrastructures and services. http://www.cloudbus.org/cloudsim/, accessed Sept. 20, 2016.

[16] 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.

[17] https://www.mixvoip.com/, accessed September 20, 2016.

[18] 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.

[19] Andrei Tchernykh, Luz Lozano, Uwe Schwiegelshohn, Pascal Bouvry, Johnatan E. Pecero, Sergio Nesmachnow, Alexander 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.

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

[21] 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.

[22] http://www.mobicents.org , accessed November 10, 2016.

[23] https://aws.amazon.com/es/solutions/case-studies/?nc2= h_ql_ny_livestream_blu, accessed November 10, 2016.

[24] https://cloud.google.com/customers , accessed November 10, 2016.

[25] 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.