# Adaptive Energy Efficient Distributed VoIP Load Balancing in Federated Cloud Infrastructure

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

Johnatan E. Pecero, Pascal Bouvry, Dzmitry Kliazovich
Computer Science and Communications Research Unit
University of Luxembourg
Luxembourg
{Johnatan.Pecero, Pascal.Bouvry, Dzmitry.Kliazovich}@uni.lu

***Abstract*— Cloud computing is widely being adopted by many companies because it allows to maximize the utilization of resources. However, the complexity of cloud computing systems with the existence of many cloud providers makes infeasible for the end user the optimal or near-optimal resource provisioning and utilization, especially in presence of uncertainty of very dynamic and unpredictable environment. Hence, adaptive load balancing algorithms are a fundamental part of the research in cloud computing. We formulate the problem and propose an adaptive load balancing algorithm for distributed computer environments. We also discuss the energy efficiency of our solution for the domain of VoIP computations on federated clouds.**

***Keywords*— *Cloud computing, load balancing, Voice over IP (VoIP), power consumption.***

## I. INTRODUCTION

Cloud computing is a pay-per-use model for enabling on-demand computing resources. It is defined as a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction.

Load balancing is a job distribution decision-making process used in many production systems and computing. It is widely known as a technique for the efficient utilization of resources, and it can be implemented with hardware and software support. Jobs arrival rate, communication delay, the variability of the job parameters, and other factors affect the performance of the systems, to deal with such complex factors it is essential to design efficient and scalable load balancing algorithms. Load balancing of services, computational jobs, virtual machines (VM), virtual storages, and database requests are identified as a major concern for the efficient use of cloud computing. This is especially relevant for end users when considering that the cloud offer is numerous and we address federated clouds.

A federated cloud is a next frontier of Cloud interoperability between private clouds, company clouds, partner clouds, and public clouds, wherein some applications are hosted internally while others are delivered via multiple cloud providers. It provides: Scalability to address peak demands; Collaboration to share infrastructure between partners; Multi-site deployments by aggregation of infrastructure across distributed data centers; Reliability by fault tolerance across sites; Performance by deployment of services; Low cost by dynamic cost aware resource allocations to reduce the overall infrastructure and operational costs; Low energy consumption by power aware service provisioning, etc. Federated clouds bring together three key stakeholders for a successful use of virtualized infrastructures: users, resource providers, and technology/software providers.

Virtualized, dynamically scalable computing resources, storages, software, and services add a new dimension to the load balancing problem. The manner in which the job allocation and re-allocation can be done depends not only on the job property and resources, but also users that share resources at the same time, in contrast to dedicated resources governed by a queuing system.

The QoS guarantee that has to be delivered to the end users is one of the major challenges for cloud computing. For VoIP, it comprises requirements on all the aspects of a connection such as service response time, throughput, loss, interrupts, jitter, latency, resource utilization, and so on. Several ways exist to provide QoS: scheduling, admission control, traffic control, dynamic resource provisioning, etc. Additional parameters need to be considered when classical load balancing techniques are used in cloud computing environment: resources are provided on demand in dynamic and scalable manner, and services are flexible. Cloud computing inherited many features from predecessors, clusters, and grid systems, but incorporates its own characteristics: scalable performance and storage capacity; elasticity; and extended functionality. The development of an effective dynamic load balancing algorithm involves many important issues: load estimation, load levels comparison, performance indices, system stability, amount of information exchanged among nodes, job resource requirements estimation, job selection for transfer, remote nodes selection, etc. [1]. Important aspects of the problem are: distribution of the nodes and virtual machine migrations. Scalability of the load balancer is also an important aspect.

Some algorithms are efficient only if the nodes are closely located and the communication delays are negligible. However, it is necessary to consider communication delay in the cloud infrastructure. A heavily loaded node can migrate its VMs to reduce the overload. However, determining which VMs need to be moved to which destination nodes, and what is the benefit of such a migration are questions difficult to answer.

Energy consumption is determined by hardware efficiency, resource management system deployed on the infrastructure, and the efficiency of applications running on the system. The efficiency is very important due to its impact on users in terms of resource usage costs, which are typically determined by the total cost of ownership incurred by a resource provider [2]. The

978-1-4799-2730-2/14/$31.00 ©2014 IEEE

goal is to avoid provisioning of more resources than it is required by applications. One solution to this problem is to migrate VMs from one node to another to consolidate resources and shut down idle nodes. Such a Dynamic Component Deactivation (DCD) policy switches off parts of the computer system that are not utilized.

In this paper, we focus on both important aspects of the load balancing: QoS and energy efficiency. The main beneficiary of their optimization is technology/software providers running its software on the cloud (e.g. the VoIP provider) that interacts with other stakeholders: the owner of the federated cloud and its broker, and cloud providers.

Proposed solutions include adaptive thresholds to determine when and where load balancing is initiated, from and to where migrations are considered beneficial, and how power consumption optimization could be reach.

## II. INTERNET TELEPHONY

The Internet telephony (VoIP–voice over internet protocol) refers to the provisioning of voice communication services over the Internet, rather than via the traditional telephone network. VoIP services significantly reduce calling rates. While the selection of a cloud based VoIP can further reduce costs, add new features and capabilities, provide easier implementations, uniform deployments, and integrates services that are dynamically scalable.

Other benefits include data availability, integrity, and security. VoIP requires that the information regarding clients is available in real time. Traditionally, the approach to deal with this issue is to invest in a large number of infrastructures to avoid loss of call and provide a correct functionality of the VoIP service. However, that infrastructure is underutilized. Most of servers are not fully used and will be replaced in time due to resource degradation. Cloud-based solutions can solve these aspects, such that VoIP service providers can avoid large investments by using smaller number of resources for one solution.

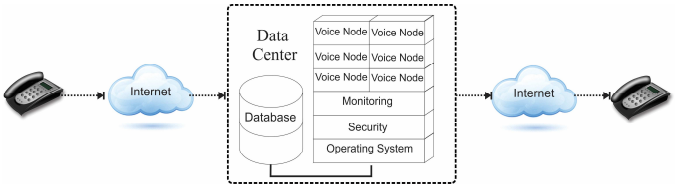


Fig. 1. Cloud VoIP architecture

To deploy and manage effective telephony tools via clouds a variety of factors need to be improved. The most important is the utilization of the infrastructure. General VoIP architecture includes elements of communication infrastructure that connect phones remotely through the Internet. The servers provide gateway, interconnection switch, session controller, firewall etc. It uses software to emulate a telephone exchange. A drawback of this architecture arises when the hardware reaches its maximum amount of connections. Traditional VoIP solutions are not very scalable. It is necessary to duplicate infrastructure or replace existing physical hardware. Fig. 1 shows a cloud based VoIP solution. The voice nodes are operated as VMs that provide variety of services (call transfer, voice mail, music on hold, etc.). The advantage of this architecture consists in increased scalability.

Next step in distributed cloud based VoIP architecture is when voice nodes are grouped on data centers geographically. However, it has several unsolved problems. To optimize the overall system performance, the processor load of the voice signal processing over IP (jobs) should be balanced.

The overload of a processor reduces quality of the call. A similar problem could happen with excess of network capacity. Furthermore, the processor idle time increases the useless expenses of the cloud provider. Load-balancing maximizes VoIP system performance by minimizing the number of processing units and the inter-processor communications. It is necessary to design a multi-level distributed VoIP load balancer to improve the local load imbalance in data centers, and new techniques to scale on federation of data centers. The most important cause of load imbalance in VoIP is the dynamic nature of the problem over time (in both computational and communication costs). Other causes may include the interference from other users that are allowed to use the same resources in time-sharing mode, the migration process, the time arrival, variability on the utilization process, etc.

Most load balancing algorithms focus on deterministic environments assuming having precise knowledge of the user jobs and system parameters. In general, it is impossible to get exact knowledge about the system. Parameters like processor speed, number of available processors, and actual bandwidth are changing constantly over the time. However, load balancing algorithms should search how to improve resources and ensure Quality of Service in a dynamic context.

## III. FORMAL DEFINITION

We address load balancing problem in the hierarchical federated cloud environment, where clouds of different providers collaborate to be able to fulfill requests during peak demands. We assume heterogeneous clouds and data centers with different number of servers, execution speed, energy efficiency, amount of memory, bandwidth, etc.

### *A. Infrastructure model*

Let us consider cloud $C$ that consists of $m$ heterogeneous nodes (data centers, sites) $D_1, D_2, \ldots, D_m$ with relative speeds $s_1, s_2, \ldots, s_m$. Each $D_i$, for all $i = 1..m$, consists of $b_i$ servers (blades, boards) and $p_i$ processors per board. We assume that processors inside one data center are identical and have the same number of cores $m_i$.

We denote the total number of cores belonging to the data center $D_i$ by $\bar{m}_i = b_i * p_i * m_i$, and belonging to all data centers of the cloud $C$ by $\bar{m} = \sum_{i=1}^{m} \bar{m}_i$. The processor of data center $D_i$ is described by a tuple $\{m_i, s_i, mem_i, band_i, eff_i\}$, where $s_i$ is a relative measure of instruction execution speed, $mem_i$ is the amount of memory (MB), $band_i$ is the available bandwidth (Mbps), and $eff_i$ is energy efficiency (MIPS per watt). A data center contains a set of routers and switches that transport traffic between the servers and to the outside world. They are characterized by the amount of traffic flowing through it (Mbps). A switch connects a redistribution point or computational nodes. The connections of the processors are static but their utilization is changed. The interconnection network architecture is three-tier architectures that include: access, aggregation, and core

layers. The interconnection between clouds will be done through public Internet.

In addition, to satisfy requests during peak demands that exceed the capacity of the cloud C, it collaborates with k external independent clouds (sites) $C_1, C_2, \dots, C_k$. Each cloud $C_i$ is characterized by the given price per time unit of the allocated instances on a pay-as-you-go basis, its energy efficiency $eff_1, eff_2, \dots, eff_k$, and relative speed $s_1, s_2, \dots, s_k$.

### B. Job model

We consider n independent jobs $J_1, J_2, \dots, J_n$ that must be scheduled on federation of clouds. The job $J_j$ is described by a tuple $\{r_j, w_j\}$ that consist of: its release date $r_j \geq 0$, and amount of processing work (instructions) $w_j$. The release time of a job is not available before the job is submitted, and its processing work (time) is unknown until the job has completed its execution at time $c_j$. A job can be allocated to one cloud only, no replication is allowed. Jobs submitted to one cloud can later be migrated to another one. We denote the start time of the job as $st_j$.

### C. Power consumption

We define the energy model by considering two components: power consumption of the cloud C and external resources: $E = E^{cloud} + E^{fed}$, where $E^{cloud} = \sum_{i=1}^{m} E_i$ is the power consumption of m data centers of C, and $E^{fed} = \sum_{i=1}^{k} E_i^{fed}$ is the power consumption of jobs migrated to the external clouds of the federation.

In this model, we consider three levels of power consumptions of resources: turned off (*off,* standby); turned on but not used (*idle*), and in use (*used*). We assume that power consumption of all system components has a constant part regardless of the machine activity. For instance, the processor in the off state includes the power consumption related with cooling. It can be considered as a stepwise function.

The power consumption of a core at time t consists of a constant part $e\_off_i^{core}$ and two variable parts $e\_idle_i^{core}$, and $e\_used_i^{core}$:

$e_i^{core}(t) = e_{off_i}^{core} + o_i(t) * \left(e_{idle_i}^{core} + w_i(t) * e\_used_i^{core}\right)$,

where $o_i(t) = 1$, if the core is on at time t, otherwise, $o_i(t) = 0$, and if the core is in operational state at time t, $w_i(t) = 1$, otherwise $w_i(t) = 0$.

The power consumption of the processor cores is $e_{proc}^{cores}(t) = \sum_{i=1}^{m_i} e_i^{core}(t)$. The power consumption $e_i^{proc}(t)$ of processor at time t consists of a constant part $e\_off_i^{proc}$ and one variable parts $e\_idle_i^{proc}$:

$$e_i^{proc}(t) = e_{off_i}^{proc} + o_i(t) * \left(e\_idle_i^{proc} + e_{proc}^{cores}(t)\right)$$

where $o_i(t) = 1$, if the processor is on at time t, otherwise, $o_i(t) = 0$.

The power consumption of processors of a server is $e_{server}^{proc}(t) = \sum_{i=1}^{p_i} e_i^{proc}(t)$.

The power consumption $e_i^{server}(t)$ of a server at time t consists of a constant part $e\_off_i^{server}$ (in the off state) and one variable parts $e\_idle_i^{server}$:

$$e_i^{server}(t) = e_{off_i}^{server} + o_i(t) * \left(e\_idle_i^{server} + e_{server}^{proc}(t)\right)$$

where $o_i(t) = 1$, if the server is on at time t, otherwise, $o_i(t) = 0$.

The power consumption of the servers in the site is $e_{site}^{server}(t) = \sum_{i=1}^{b_i} e_i^{server}(t)$.

The power consumption of the site is

$e_i^{site}(t) = e_i^{site} + o_i(t) * \left(e\_idle_i^{site} + e_{site}^{server}(t)\right)$.

Total power consumption of the site $D_i$ is $E_i = \sum_{t=1}^{C_{max}} e_i^{site}(t)$.

Finally, $E_i^{fed} = \sum_{i=1}^{k} eff_i \cdot w_j / s_j$ is the total power consumption to process jobs on external resources, where $eff_i$ is the energy efficiency of cloud $C_i$ and $w_j$ is the total work of all jobs allocated on $C_i$.

### D. Optimization criteria

In this paper, we use two criteria: $L$ and $E^{cloud}$. Mean latency (service response time) $L = 1/n(st_j - r_j)$ is the average time a user spends waiting for a connection. It reflects load balancing latency, network delays, and job waiting time. It measures the user satisfaction for the VoIP service. Energy consumption $E^{cloud}$ allows VoIP providers to measure the performance in terms of cost parameters that helps him to establish utility margins.

## IV. Related Work

In this section, we give a brief overview of the load balancing algorithms in distributed computer environments. Table I presents their main characteristics, and metrics used to study quality of the algorithms. Brief description of the most relevant algorithms is presented below. Details of other algorithms can be found in the referred papers.

Job-Idle-Queue (JIQ) [6] is a large-scale load balancing algorithm dynamically scalable with distributed dispatchers for cloud data centers. The central idea is to decouple the discovery of the lightly loaded servers based on the job assignment. The algorithm consists of two load balancing systems to assign the jobs on idle servers. I-queue structures maintain the information of the idle processor, when a job arrival the dispatcher assigns the job to the first element of its I-queue. If the I-queue is empty, the dispatcher directs the job to a randomly chosen processor.

Task Scheduling based on LB (TSLB) [11] takes into account the requirements of users and the load balancing in cloud environment. The first level scheduling allocates the user applications in to the VMs and, the second level assigns the VM to host resources. The demands of resources of the jobs are known a priory but can be changed during the execution. In this case, VM could be moved to another node with enough free resources, or VMs residing on the same node be moved away in order to free up resources.

Honeybee Foraging (HF) [12] dynamically allocates web services on servers to regulate the system against demand. This algorithm is used for self-organization and global load balancing via local server actions. Each server group a virtual servers anyone with his own queue of jobs. The notion of "advert board" is used to communicate the global colony profit. The idle servers read the advert board, chosen an advert and serve the request or randomly serve a servers queue request.

Biased Random Sampling (BRS) [12] is the self-organization algorithm based on random sampling of the system domain, so the node's load is maintained near to a global mean

measure. The load on a server is represented by edges and network that provides a measure of initial availability status. The nodes accept jobs depending on a value, if the value is equal or greater than a threshold the node executes the job, else the job is sent to a random neighbor.

Active Clustering (AC) [12] is self-aggregation algorithm for large scale Cloud systems. It connects similar services, intended to group instances together by local rewiring the network. Algorithms work well when they can delegate workload to other nodes that are aware of similar nodes.

Two-phase scheduling (OLB+LBMM) [14] combines Opportunistic Load Balancing (OLB) and Load Balance Min-Min (LBMM) scheduling. In the first phase, the OLB scheduling algorithm assigns tasks to the service manager by the manager. In the second phase, LBMM scheduling algorithm chooses the suitable node to execute subtasks.

Compare and Balance (CB) [17] reduces the migration time of VMs. Based on sampling to reach the equilibrium, it calculates the cost of the nodes in the system. Then, it compares its current cost node with other randomly chosen. If the cost of current node exceeds the cost of chosen node, then VM is migrating with certain probability.

VM to physical Machine Mapping (VMMP) [21]. First, the utilization of resources in the nodes are summed and normalized to get weight of each node, the higher weighted of a node represents more available resources. Then a probability approach is used to select a node, this probability is proportional to the weight of a node.

Power Aware Load Balancing (PALB) [23] is a power aware algorithm. It maintains the state of all nodes and decides the number of compute nodes that should be operating based on utilization percentage. It has three sections: first section assigns the VM in a node, second section is used to power on additional compute nodes, and last section is responsible for powering down idle nodes.

Self-Organized Agent System for dynamical load-balancing (SOAS) [25] is a decentralized algorithm for distributed systems. It is based on the sand pile model, where avalanches can reconfigure the state of the system. This behavior fits well with the idea of non-clairvoyant scheduling with Bags-of-Tasks. Avalanches are the load-balance process that results in a new allocation of tasks in computing resources.

## V. Adaptive Load Balancing Algorithm

Campos and Scherson proposed a dynamic distributed load balancing algorithm named Rate of Change (RoC-LB) [27]. The balancer (*Bal*) makes job distribution decisions at run-

TABLE I. Load Balancing Algorithms.

| | Characteristics | | | | | | | | | | | | | | Metrics | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Centralized | Decentralized | Hierarchical | On line | Off line | Non-clairvoyant | QoS | Migration cost | Network delay | Limited historical | Static | Dynamic | Replication | Migration | Utilization | Performance | Overhead | Response time | Scalability | Fault tolerant | Throughput | Energy | |
| VD | | | • | | • | • | | | | | | • | | VM | • | | | | | | | | [3] |
| WCAP | | | • | • | | • | | | | | | • | Server | | • | • | | | | • | | | [4] |
| VWR | • | | | • | | • | | | • | | | • | Server | Request | • | • | | | | | | | [5] |
| JIQ | | • | | • | | • | | | | | | • | | | | • | • | • | | | | | [6] |
| LFM | • | | | • | | • | | | | | | • | | | | • | | | | | • | | [7] |
| SVB | • | | | • | | • | | • | | • | | • | | VM | • | | • | | | | | | [8] |
| CLBVM | • | | | • | | • | | | | | | • | | VM | • | • | | • | | | • | | [9] |
| LBVS | • | | | | | | | | | • | | • | Files | | | • | | • | • | • | | | [10] |
| TSLB | • | | | • | | • | | | | • | | • | | VM | • | • | | • | | | | | [11] |
| HF | | • | | • | | • | | | | | | • | | Job | | • | | | • | | • | | [12] |
| BRS | | • | | • | | • | | | | | | • | | Job | | • | | | • | | • | | [12] |
| AC | | • | | • | | • | | | | | | • | | Job | | • | | | • | | • | | [12] |
| ACCLB | | • | | • | | | | | | | | • | | Job | • | • | | | • | • | | | [13] |
| OLB+LBMM | • | | | | | • | | | | • | • | | | | • | • | | | | | | | [14] |
| ED | • | | | • | | • | • | | | | | • | Server | Session | • | | | | • | | | | [15] |
| Carton | | • | | | • | • | | | | | | • | | | • | • | • | | | | | | [16] |
| CB | | • | | | • | • | | • | | | | • | | VM | • | | • | | | | | | [17] |
| TSPBRR | • | | | | • | | | | | | • | | | | • | | | • | | | | | [18] |
| ESCE | • | | | • | | • | | | | • | | • | | | • | | | • | | | | | [19] |
| CLBDM | • | | | • | | • | | | | | | • | | Session, VM | • | | | • | | | | | [20] |
| MR | | • | | | | | | | | | | | | | | • | | | • | | | | [26] |
| VMMP | • | | | • | | • | | | | | | • | | | • | | | | | | | | [21] |
| FBRR | • | | | • | | • | | | | | | • | | | • | | | • | | | | | [22] |
| PALB | • | | | • | | • | | | | | | • | | | • | | | | | | | • | [23] |
| LBMM | • | | | | • | • | | | | • | | • | | Task | • | | • | • | | | • | | [24] |
| Min-Min | • | | | | • | • | | | | • | • | | | | • | | • | • | | | • | | [24] |
| Max-Min | • | | | | • | • | | | | • | • | | | | • | | • | • | | | • | | [24] |
| SOAS | | • | | • | | • | | | | | | • | | Task | • | | | | | | • | | [25] |
| RoC-LB | | • | | • | | • | | • | • | | | • | | Task | • | • | • | • | • | | | | [27] |

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 Change), and current load balancing parameters.

To define WHEN load balancing should be started the algorithm considers three thresholds: $Ub$ upper bound, $Lb$ lower bound, and $Cb$ critical bound. If the load is larger than $Ub$ then the *Bal* is considered as a source of the load and can satisfy job requests. If the predicted load is less than $Lb$ the *Bal* is considered as a sink. When the current load is between these two bounds, the *Bal* is in the neutral state. However, if the load or the predicted load fall below $Cb$, the *Bal* immediately initiates a request for a load.

We extend this algorithm and present a new Power Aware Adaptive Rate of Change (PA-AdRoC) algorithm that is based on an adaptive decision policy and energy optimization.

Adaptability is essential for the efficient use of cloud infrastructure. Clouds differ from previous computing environments in the way that they introduce a continuous uncertainty into the computational process. The uncertainty becomes the main feature of the cloud computing and the principal difficulty of the efficient resource management. There are several major sources of uncertainty: dynamic elasticity, dynamic performance changing, virtualization, loosely coupling application to the infrastructure, among many others. A workload in such an environment is not predictable and can be changed dramatically. It is impossible to get exact knowledge about the system. Parameters such as an effective processor speed, number of available processors, and actual bandwidth are changing over the time.

PA-AdRoC takes into account these uncertainties. The accuracy of each balancing decision depends on the actual cloud characteristics at the moment of balancing.

Cloud parameters are changing over time and balancing parameters should be adapted to these changes. This adaptive approach can cope with different workloads, and cloud properties. To adjust $Ub$, $Lb$, and $Cb$, the past information within a given time interval can be analyzed to determine an appropriate parameters. This interval should be set according to the dynamics of the system.

Let $n_i(t)$ be the load of *i*th *Bal* at time $t$. Let $\Delta_i(t) = (n_i(t) - n_i(t - si))/si$ be the load change during sample interval $si$=$[t - si, t]$. We named it load change speed or load consumption speed. The sampling time interval $si$ is an adaptive parameter; finer sampling allows detecting the need to balance the system faster, but it generates a larger communication overhead.

*Bal* uses $\Delta_i(t)$ as a predictor of the future load. It can be also used to estimate the number of sampling intervals to reach an idle state. Let $rd_i(t)$ be the response delay of at time $t$. It is an adaptive parameter, and it is defined as the time it takes between the initiation of a load request and the reception of load. If the time to reach idle state is less than $rd_i(t)$, then *Bal* must initiate a migration request. Let us note that $si$ and $nd_i(t)$ are independent from others *Bal*s.

Fig. 2 shows possible load balancing scenarios. Solid line shows real workload, dashed lines are predicted workloads.

Let us assume that at time $T_\alpha$ the workload is $A$. Based on the $\Delta_i(t)$ calculated on the previous $si$ interval, predicted workload is $A' = A - \Delta_i(t) \cdot si$. However, after $si$ real workload becomes $B$. Based on this information, new prediction $B'$ is calculated. It depends on the real workload consumption speed $\Delta_i(t + si)$ . For this example, $\Delta_i(t)$ is constant and equals to two times of $si$. Since $A'$ is between $Ub$ and $Lb$ the *Bal* does not issue a load request.

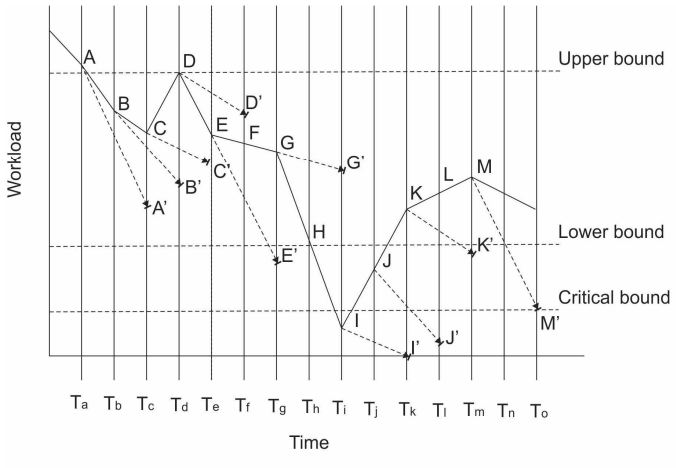


Fig. 2. Load balancing scenarios.

At time $T_i$ the *Bal* immediately initiates a request for load regardless of the predicted future load based upon the estimation $\Delta_i(t)$ value. Estimation $J'$ on $T_j$ is under $Cb$ but *Bal* cannot initiates a request because the request at $T_i$ is not arrived, a new request only can be generate at $T_k$. In PA-AdRoC, unlike RoC-LB, if the load is larger than $Ub$ then the *Bal* sends jobs to the sinks.

To define WHERE a load is requested from or send to, each *Bal* keeps two lists. The sink list records *Bal*s that previously needed jobs, and source list enrolls *Bal*s that previously offered jobs. *Bal* that initiates a request is considered to be a sink. A sink selects a *Bal* from its source list for a load request, and sends a requesting message. The source can accept the request or broadcast the request to other *Bal*s from its own source list. *Bal* does not send several load requests at the same time. It has to wait an answer for the first request until it sends another message. The result of this message is the load coming from other *Bal* or the request comes back as unfulfilled.

In our algorithm, each element of the source and sink lists includes not only IP address like in RoC-LB, but load $n_i(t)$, and energy efficiency $eff_i$ of the corresponding *Bal.* This information is not accurate and updated dynamically. In our case, choosing the sink/source node is a two-parameter problem. In the future work, we consider also requested load $rn_i(t)$, $\Delta_i(t)$, $rd_i(t)$, answer time of the request $rt_i(t)$, admissibility of *Bal*s, total power consumption $E_i(t)$, etc. The goal is to choose the most adequate compromise solution.

## VI. Conclusions

In this paper, we formulate and discuss load balancing problem addressing VoIP in cloud computing federation. We overview the last advances of load balancing in distributed computer environments to understand the main characteristics of

load balancing algorithms. We show that none of these works directly addresses the problem space of the considered problem; however, they form a valuable basis for our work.

We define models of cloud federation and energy consumption, and propose new distributed adaptive power aware load balancing algorithm PA-AdRoC. Adaptability is essential characteristic of this algorithm and makes it suitable for environment with presence of uncertainty. It makes job distribution decisions on-line at run-time, locally and asynchronously, taking into account QoS and energy consumption. It does not take into account job execution time, execution time estimation, topology and communication bandwidth. It takes load balancing decisions depending on the actual cloud characteristics at the moment of balancing such as number of available machines, actual communication delay, workload consumption speed, etc. Due to these parameters are changing over time, load balancer adapts to these changes. This adaptive approach can cope with different workloads, cloud properties, and cloud uncertainties such as elasticity, performance changing, virtualization, loosely coupling application to the infrastructure, parameters such as an effective processor speed, number of available processors, and actual bandwidth, among many others. To this end, the past information within a given time interval can be analyzed to determine an appropriate parameters independently by each balancer. The time interval for this adaptation should be set according to the dynamics of the system.

The proposed algorithm can be used for a wide range of applications. However, further study is required to assess its actual efficiency and effectiveness in each domain. This will be the subject of future work. Moreover, load balancing in cloud environment, where only an admissible subset of resources is available to reduce the amount of the network traffic is another important issue to be addressed.

## ACKNOWLEDGMENT

This work is partially supported by CONACYT (Consejo Nacional de Ciencia y Tecnología, México), grant 178415. The work of P. Bouvry, J. Pecero and D. Kliazovich is partly funded by Green@Cloud (INTER/CNRS/11/03) and ECO-CLOUD (C12/IS/3977641).

## REFERENCES

[1] A. M. Alakeel. A guide to dynamic load balancing in distributed computer systems. International Journal of Computer Science and Network Security (IJCSNS), 10(6), pp. 153-160, 2012

[2] A. Beloglazov, J. Abawajy, and R. Buyya. Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. FGCS, 28(5), pp.755-768, 2012.

[3] A. Singh, M. Korupolu, and C. Mohapatra. Server-storage virtualization: integration and load balancing in data centers. In Proceedings of the 2008 ACM/IEEE conference on Supercomputing, pp. 53, IEEE, 2008.

[4] H. Mehta, P. Kanungo, and M. Chandwani. Decentralized content aware load balancing algorithm for distributed computing environments. In Proceedings of the International Conference & Workshop on Emerging Trends in Technology, pp. 370-375, ACM, February 2011.

[5] A. M. Nakai, E. Madeira, and L. E. Buzato. Load balancing for internet distributed services using limited redirection rates. In Dependable Computing (LADC), 2011 5th Latin-American Symposium on, pp. 156-165, IEEE, April 2011.

[6] Y. Lu, Q. Xie, G. Kliot, A. Geller, J. R. Larus, and A. Greenberg. Join-Idle-Queue: A novel load balancing algorithm for dynamically scalable web services. Performance Evaluation, 68(11), 1056-1071, 2011.

[7] X. Liu, L. Pan, C. J. Wang, and J.Y. Xie. A Lock-Free Solution for Load Balancing in Multi-Core Environment. 3rd Inter. Workshop on Intelligent Systems and Applications (ISA), 2011, pp. 1-4, IEEE, 2011.

[8] J. Hu, J. Gu, G. Sun, and T. Zhao. A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. In Parallel Architectures, Algorithms and Programming (PAAP), 2010 Third International Symposium on, pp. 89-96, IEEE, December 2010.

[9] A. Bhadani, and S. Chaudhary. Performance evaluation of web servers using central load balancing policy over virtual machines on cloud. In Proceedings of the Third Annual ACM Bangalore Conference, p. 16, ACM, January 2010.

[10] H. Liu, S. Liu, X. Meng, C. Yang, and Y. Zhang. LBVS: A load balancing strategy for virtual storage. In Service Sciences (ICSS), 2010 International Conference on, pp. 257-262, IEEE, May 2010.

[11] Y. Fang, F. Wang, and J. Ge. A task scheduling algorithm based on load balancing in cloud computing. In Web Information Systems and Mining, pp. 271-277, Springer Berlin Heidelberg, 2010.

[12] M. Randles, D. Lamb, and A. Taleb-Bendiab. A comparative study into distributed load balancing algorithms for cloud computing. In Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on, pp. 551-556, IEEE, April 2010.

[13] N. J. Kansal, and I. Chana. Cloud Load Balancing Techniques: A Step Towards Green Computing. International Journal of Computer Science Issues (IJCSI), 9(1),2012.

[14] S. C. Wang, K. Q. Yan, W. P. Liao, and S. S. Wang. Towards a load balancing in a three-level cloud computing network. In Computer Science and Information Technology (ICCSIT), 2010 3rd IEEE International Conference on, vol. 1, pp. 108-113, IEEE, July 2010.

[14] V. Nae, R. Prodan, and T. Fahringer. Cost-efficient hosting and load balancing of massively multiplayer online games. In Grid Computing (GRID), 2010 11th IEEE/ACM Int. Conf. on, pp. 9-16, IEEE, 2010.

[16] R. Stanojevic, and R. Shorten. Load balancing vs. distributed rate limiting: an unifying framework for cloud control. In Communications, 2009. ICC'09. IEEE International Conference on, pp. 1-6, IEEE, 2009.

[17] Y. Zhao, and W. Huang. Adaptive distributed load balancing algorithm based on live migration of virtual machines in cloud. In INC, IMS and IDC, 2009. NCM'09. Fifth Int. Joint Conference on, pp. 170-175, IEEE.

[18] P. Samal, and P. Mishra. Analysis of Variants in Round Robin Algorithms for Load Balancing in Cloud Computing, Int. Journal of Computer Science and Inf. Technologies, 4(3), pp. 416–419, May 2013.

[19] J. Kaur, J. Comparison of load balancing algorithms in a cloud. International Journal of Engineering Research and Applications, 2(3), pp. 1169-173, 2012.

[20] B. Radojevic, and M. Zagar. Analysis of issues with load balancing algorithms in hosted (cloud) environments. In MIPRO, 2011 Proceedings of the 34th Inter. Convention, pp. 416-420, IEEE, 2011.

[21] J. Ni, Y. Huang, Z. Luan, J. Zhang, and D. Qian. Virtual machine mapping policy based on load balancing in private cloud environment. In Cloud and Service Computing (CSC), 2011 International Conference on, pp. 292-295, IEEE, December 2011.

[22] S. Sethi, A. Sahu, and S. K. Jena. Efficient load balancing in cloud computing using fuzzy logic. IOSR Journal of Engineering, 2(7), pp. 65-71, 2012.

[23] J. M. Galloway, K. L. Smith, and S. S. Vrbsky. Power aware load balancing for cloud computing. In Proceedings of the World Congress on Engineering and Computer Science, vol. 1, pp. 19-21, October 2011.

[24] T. Kokilavani, and D. I. Amalarethinam. Load Balanced Min-Min Algorithm for Static Meta-Task Scheduling in Grid Computing. International Journal of Computer Applications, 20,2011.

[25] J. J. Laredo, B. Dorronsoro, J. Pecero, P. Bouvry, J.J. Durillo, and C. Fernandes. Designing a self-organized approach for scheduling bag-of-tasks. In P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC), 2012 7th Int. Conference on, pp. 315-320, IEEE, 2012.

[26] T. Gunarathne, T. L. Wu, J. Qiu, and G. Fox. MapReduce in the Clouds for Science. In Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second Int. Conference on, pp. 565-572, IEEE, 2010.

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