Taxonomy of resource allocation algorithms for inner IaaS cloud data centres

Dang Minh Quan
Institute of Information Technology for Economics, National Economics University
207 Dai Co Viet, Hanoi, Vietnam. Email: {quandm}@neu.edu.vn

***Abstract* - Cloud computing has become more and more popular with the widely deployment of several cloud infrastructures. Infrastructure-as-a-service (IaaS) Cloud computing replaces bare computer hardware. The cloud user will use the virtual machines (VMs) to fullfil their computing requirements. Among the components of IaaS cloud software stack, the resource allocation module is very important as it selects suitable VMs and the place to execute VMs. This paper focuses on studying and classifying algorithms used in the resource allocation module. The issues of how to apply those algorithms are also discussed.**

Keywords – Cloud computing, IaaS data centre, resource allocation algorithm.

I. INTRODUCTION

Cloud computing has become more and more popular with the widely deployment of several cloud infrastructures [1]. The core principle of cloud computing is delivering services from shared hardware. The goal of this computing model is to make a better use of distributed resources, put them together to make higher throughput and be able to handle large-scale computation problem. People often categorize Cloud computing into three levels of use model or cloud computing services as presented in Fig. 1.

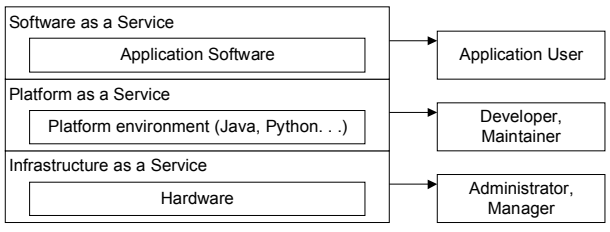

Figure 1. General Cloud categorization

Infrastructure-as-a-service (IaaS): Cloud computing replaces bare computer hardware. Users of IaaS have the ability to support operating systems and applications, but don't wish to buy server, storage and networking hardware and a data centre to house the hardware. Examples of those providers are companies such as Amazon[2], ENKI[3], GoGrid[4].

Platform-as-a-service (PaaS): Cloud computing replaces an execution environment for a computer language by providing a system ready to execute the user's software. The user of PaaS is the programmer. Examples of those providers are companies such as Engine Yard[5] or Google[6].

Software-as-a-Service (SaaS): The user interacts directly with the Cloud-hosted software, and often pays for "seats" or "users" instead of computer time. Examples of those providers are NetSuite[7], SalesForce.com[8], Google Apps[9].

Within the scope of this paper, we focus on IaaS Cloud. Figure 2 presents the typical architecture of an IaaS cloud. An IaaS cloud has many computing nodes grouped together to form clusters. For each node, there is virtualization component. It is a special purpose operating system that creates and maintains the VMs as well as serves their requests for accessing to hardware resources.

An NC (Node Controller) executes on every node that host VM instances. An NC makes queries to discover the node's physical resources – the number of cores, the size of memory, the available disk space - as well as to learn about the state of VM instances on the node. The information collected is propagated up to the Cluster Controller.

The Cluster Controller (CC) generally executes on a cluster front-end machine. CC has three primary functions: issue running instances to specific NCs, control the instance virtual network overlay, and gather/report information about a set of NCs.

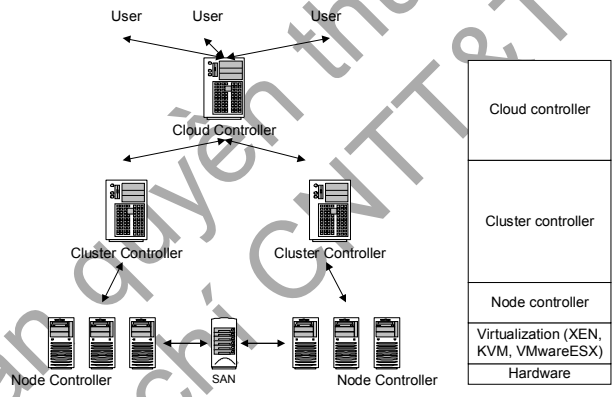

Figure 2. Typical IaaS cloud architecture

Cloud Controller is the entry-point into the cloud for users and administrators. It queries node managers for information about resources, makes resource allocation decisions, and implements them by making requests to cluster controllers.

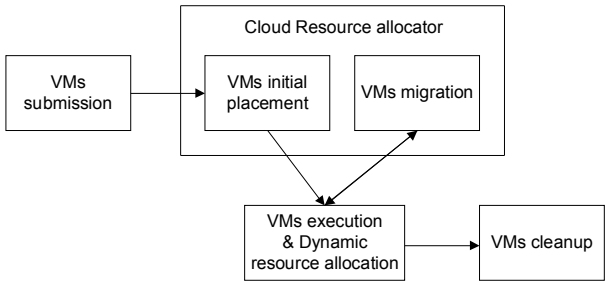

Figure 3. Role of resource allocation in VM life cycle

Among the components of IaaS cloud software stack, the resource allocation module is very important as it assigns resources to VMs. Figure 3 presents the role of resource allocation in VMs' life cycle. When a user submits VMs to the IaaS cloud system, the cloud resource allocation module will find the acceptable VMs and find the initial places to run those VMs. During the VMs execution process, the cloud system may migrate VMs out of the initial place to other computing nodes. The cloud resource allocation module decides which nodes to migrate VMs to. While the node is executing VMs, the OS of the node may perform coarse-grained dynamic resource allocation to VMs [62].

In this paper, we want to concentrate on the natural distributed character of the IaaS cloud, but not on the OS within a physical node. Thus, this paper focuses on studying and classifying algorithms used in the cloud resource allocation module to decide which VMs to run and define the computing nodes to run those VMs. The rest of the paper is organized as follows. Section 2 summarizes related survey works. Section 3 presents taxonomies for resource allocation algorithms. Section 4 surveys various algorithms that have used for cloud resource allocation. Finally, we end this paper with discussion on open issues, lesson learned in section 5 and conclusion in Section 6.

II. RELATED WORKS

According to our knowledge, we have not noticed any comprehensive taxonomy journal article on IaaS cloud resource allocation approaches. However, a number of related surveys and review book chapters that referred to IaaS cloud resource allocation have been published. In this section, we describe only those surveys and reviews. The detail description of referred algorithms and their original reference are in section IV.

In [10], the authors studied open source cloud platforms. This work compared solutions and their business model (hardware, middleware and user level) according to configuration flexibility. It also compared the service, infrastructure and users of those systems. The important of cloud resource allocation was stated, but none of detail issues were discussed.

The work in [1] extended a taxonomy and survey of cloud computing system to both open source and commercial cloud platforms. The cloud systems are mainly characterized with architecture, virtualization management, service, fault tolerance and security. Related to resource allocation, the authors referred only the load balance feature. In which, most studied systems use simple algorithms such as Round Robin, Greedy or server load equalization at IaaS level.

In [11], the author presented the taxonomy and survey of energy-efficient data centres and cloud computing systems. This work discussed many energy-saving techniques ranging from hardware level, OS level, Virtualization level to data centre

level. At data centre level, the authors described several research works about saving energy techniques. Those techniques mostly are based on DVFS (Dynamic Voltage and Frequency Scaling), VM consolidation and power switching.

In [12], the authors discussed various scheduling techniques for traditional distributed systems, grid computing systems and cloud computing systems. However, the author only touched the surface of real works for cloud computing. The IaaS cloud employs the VM concept. Each VM could have multiple CPUs and must be allocated completely within a physical machine. From the point of resource allocation, this is the distinguished character of Cloud IaaS compared with traditional distributed system and Grid computing. In other types of distributed system, one job including many processes can be spread out to multiple physical machines.

The work in [63] took some scheduling algorithms for cloud computing and performed experiment to do comparative analysis. The main comparative criteria include execution time, resource use rate and cost of algorithm. Also following this way, the work in [64] focused on scheduling schemes for on-demand IaaS requests. However, the authors of [64] used the analytical model and studied the ability of reducing energy consumption.

III. RESOURCE ALLOCATION ALGORITHMS TAXONOMY

A. *Overview of the cloud resource allocation problem*

The resource allocation algorithm responds for finding the resource allocation solution that satisfies a specific goal of the cloud provider. This goal could be optimizing power consumption, optimizing cost, ensuring SLA, etc. The typical resource allocation architecture for IaaS cloud is presented in Figure 4.

In general, the input for resource allocation includes resource information and workload information. Based on this input information, the resource allocation algorithm finds out resource allocation solution.

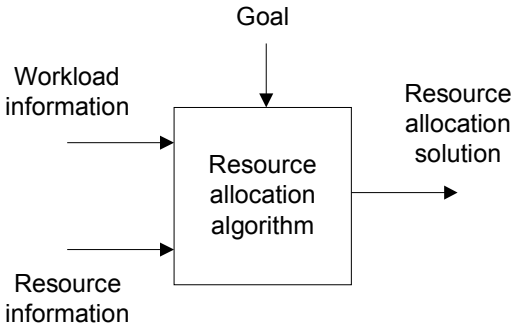

Figure 4. Typical resource allocation architecture

B. *Taxonomy of resource allocation algorithms*

From our point of view, we classify IaaS's resource allocation algorithms according to three main criteria: execution phase of VMs, business model of providers and goal of the algorithm. We build the hierarchy of those algorithms as presented in Fig. 5. In this section, we discuss in detail each criterion.

1) *Execution phase of VMs*

This classification is based on activities of the cloud system during VM's lifecycle. At the beginning, when a VM comes to the system, the system will do the initial placement. During the VM execution, the system will perform VM migration if necessary. The differences of task and workload parameters make the working mechanism of resource allocation algorithms different from each other at each execution phase.

Initial placement - When users submit VMs to cloud, initial placement algorithms are executed. Their task is determining if the VM can be admitted and where to execute it. The algorithms need the resource information including static and dynamic information. The static information includes the computing node information such as number of computing nodes, number of CPUs, number of cores, memory capacity, storage capacity, etc. The dynamic information include the current usage of resource such as CPU load, memory load, network load, etc. At this phase, the user's preference is strictly respected. The workload in this case is a group of VMs with their required resource information. It usually includes the number of vCPUs, amount of memory, amount of

storage and bandwidth. In some cases, this information also includes the bid price of the user.

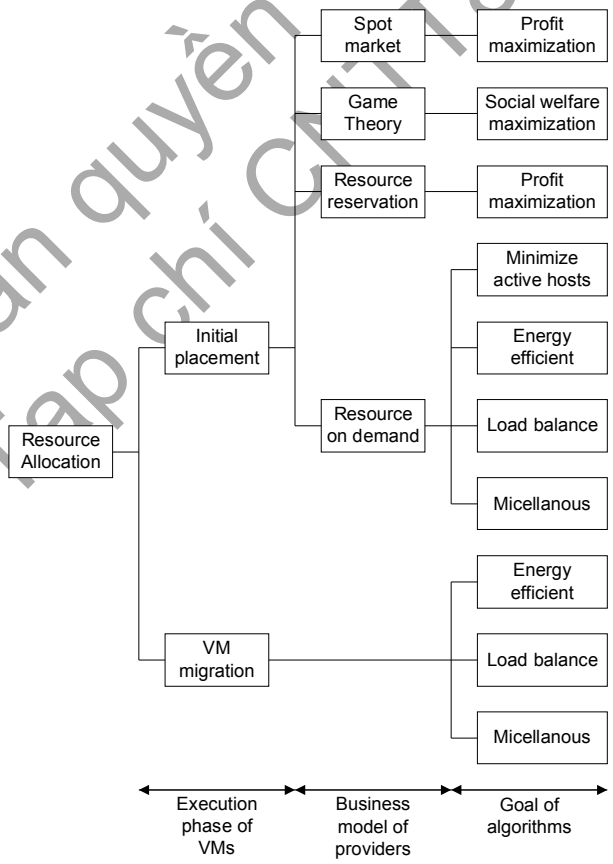

Figure 5. Taxonomy of resource allocation algorithms

VM migration - VMs migration algorithms are triggered by internal data centre policy during the operation process of data centre. Unlike the initial placement algorithms, migration algorithms do not care about discarding some VMs. They try to find new place for running VMs to satisfy data centre's goal. To do this task, the resource information is the same as in the previous case. But, the workload in this case is all VMs running in the data centre with their current resource information usage. It usually includes the CPU usage rate, amount of memory usage, amount of storage usage and bandwidth usage.

2) Business model of providers

This classification is based on the way IaaS cloud providers distribute cloud resource to users. The IaaS cloud providers can distribute resource under spot market model, resource reservation model, game theory model or resource on demand model. The difference in business model makes the working mechanism of the underlying resource allocation algorithms different from each other.

Spot market - In the spot market model, user has to bid the maximum price he/she is willing to pay per resource hour. If the user's maximum price bid exceeds the current Spot Price, the user can run VMs on the gained resource within that period. After that, user has to bid again to gain the right of using resource for the next period. It is noted that the resource allocation mechanism for spot market includes two separate steps. The first step is determining which VM with its correlated bid price could be admitted to the system. The second step is determining which admitted VM running on which physical machine (PM).

Resource reservation - In the resource reservation model, user choose the VMs instance types and the period to use those VMs. Reserved Instances offer a capacity reservation, so that user can launch several instances he/she has reserved when he/she needs them. For long-term resource reservation, such as in Amazone EC2, Reserved Instances give user the option to make a low, one-time payment for each instance he/she wants to reserve. In turn, he/she receives a significant discount on the hourly charge for that instance. In the elastic cloud environment, the provider must make sure they have enough resources to meet demand. Otherwise, the provider will need to pay compensation to those customers whose reserved VM instances have not been executed when they wanted.

Game theory - In the game theory model, the complex interactions between multiple clients using the cloud simultaneously are considered. At each scheduling round, the system considers all the existing workloads and new workloads. The existing workloads are prioritized to be scheduled first. After that, users having new requests interactively select the right resources from the available pool. This selection process takes place through several rounds until it reaches equilibrium. This equilibrium ensures that clients are charged (near) optimal *prices* for their

resource usage and resources of the cloud are used near their optimal capacity.

Resource on demand – This is the current most popular resource distribution model for IaaS cloud. In the resource on demand model, when users need resource to run their VMs, they can get them from the cloud. They can use them as long as they wish. The price of using resource for each VM instance is usually fixed.

3) Goal of the algorithm

This classification is based on the expectancy of the cloud providers with their resource usage. Some main focuses are energy-efficient, profit maximization, load balance or number of active PMs minimization. Besides that, some other goals include fault tolerance, ensuring SLA, etc. It is clear that different goals lead to different resource allocation algorithms.

Profit maximization - Profit maximization is an important goal for commercial cloud providers. There are two ways to gain profit maximization. If the cloud providers have many prices for a single resource, in the spot market for example, selling the same amount of resource with highest price will reach profit maximization. If the cloud providers have fixed price for a single resource, maximizing the workload running by the same amount of resource also ensures profit maximization.

Social welfare maximization – Social welfare maximization is a goal of economic that tried to apply to cloud environment. The mechanisms manage to distribute resources in a way that reaches equilibrium. This equilibrium ensures that clients are charged (near) optimal *prices* for their resource usage and resources of the cloud are used near their optimal capacity.

Energy efficient - Saving money in the energy budget of a data centre, without sacrificing SLAs is an excellent incentive for data centre owners. It would at the same time be a great success for environmental sustainability. Usually the power consumption of a cloud data centre is calculated through the power consumption of the IT equipment and the PUE (Power usage effectiveness) value of the data centre.

$$Pcenter = Pit\_equipments * PUEcenter \quad (1)$$

With a data centre, its PUE is relatively stable. Thus, efforts to cut energy consumption of a data centre focuses on reducing the energy consumption of servers. The power consumption model of a server is further divided to power consumption of CPU, disk, memory, etc. Detail server power consumption model can be seen in [13]. Allocating the same amount of workload using minimal amount of energy will assure energy-efficient.

Number of active PMs minimization – This goal derived from the goal of the online bin-packing problem. Objects of different volumes must be packed into a finite number of bins of capacity V in a way that minimizes the number of bins used. In the IaaS cloud environment, the resource allocation module has to allocate a set of VMs into a finite number of PMs in a way that minimizes the number of PMs used.

Load balance - Load balance is one of the main challenges in cloud computing. It requires distributing the dynamic workload across multiple nodes to make sure that no single node is overwhelmed. It helps in optimal use of resources and hence in enhancing the performance of the system. For the IaaS cloud, the load of each host is calculated with:

$$Load = \frac{\sum VM\_entitlement}{Host\_capacity} \quad (2)$$

Usually, the VM entitlement is the occupied CPU resource or memory resource of the VM running on the host. The host capacity is the available resource of the host, which can be provided to VMs.

Miscellaneous – Besides main goals as stated above, the literature also recorded some research works about IaaS cloud resource allocation with other goals such as ensuring SLA, fault tolerance, multi objectives, etc.

IV. SURVEY OF RESOURCE ALLOCATION ALGORITHMS

A. VMs initial placement algorithms

1) Profit optimization algorithm for clouds support resource reservation

The work in [14] proposed a mechanism to allocate computing resource to workloads with various mixes of best-effort and advance reservation requests. Incoming advance reservation leases are always scheduled right away, where best-effort leases are put on a queue. The scheduling function periodically evaluates the queue. It uses an aggressive backfilling algorithm [21, 23] to decide whether any best-effort leases can be scheduled. When an advance reservation lease comes, the scheduler will try to choose nodes that will not need preempting another lease.

In [17], the authors presented a scenario of cloud platforms that supports resident applications and resource reservation. If some resource reservation requests come, VMs running resident applications can easily be shifted somewhere [18] to squeeze out enough resources for the reservation. However, doing this might lead to negative impact to the applications originally resident due to limited total resource amount. The authors defined a revenue function and the adaptive QoS-aware resource reservation management algorithm. With each physical node, the algorithm checks the capacity, forms a new configuration according to the reservation and forecasts the revenue. It then select the best node giving the largest revenue.

From the description above, we can see that both algorithms in [14] and [17] support reservation as well as non-reservation workloads. They both give higher priority to reservation request. When a new reservation request comes, the algorithm in [14] and [17] may suspend or migrate non-reservation VMs to have enough cloud resources for this request. The difference between two algorithms is the behavior with the affected VMs. While the algorithm in [14] does not care much about the affected non-reservation workloads, the algorithm in [17] has to consider the fine before deciding to migrate the non-reservation VMs.

2) Profit optimization algorithms for clouds support spot market

Motivated by low resource use, Amazon EC2 introduced the spot instance mechanism to allow customers to bid for unused Amazon EC2 capacity [2]. Amazon EC2 at each availability zone has fixed number of virtual machine types and fixed number of instances of each VM type. Amazon EC2 runs one spot market for each VM type in each availability zone. A customer submits a request that specifies the type, the number of instances, the region desired and the biding price per instance-hour. The provider assigns resources to bidders in decreasing order of their bids until all available resources have been allocated or all resource requests have been satisfied. The selling price (i.e. the spot price) is equal to the lowest winning bid. The actual VM – physical host assignment is done with Round Robin algorithm [1], which is described in detail in Section 4.1.4.

Unlike Amazone EC2, in [19], the authors developed the model to allow users to bid for bundles of items. This model is called combinatorial auction. Determine the set of winning users with the goal of maximizing the income in this model is a NP-hard problem [1]. To solve this problem, the authors proposed the CA-GREEDY algorithm. It collects users bundle of VMs and bid price, does weighted sum of VM of each bid, calculates the bid density, sorts bids by density and then allocates highest to lowest, until resources exhaust.

The above works all assume fixed number of virtual machine types and fixed number of instances of each VM type. In [20], the authors proposed a model called Dynamic VM Provisioning and Allocation (DVMPA). It allows users to bid for a bundle of VMs of different types. It can configure the set of available computing resource into different numbers and types of VM instances. The author proposed the CA-PROVISION algorithm to solve the problem. It first collects users' bundle of VMs of different types and bid price, does weighted sum of VM of each bid, calculate the bid density, sort bids by density and then allocates highest to lowest, until resources exhaust. Once the winners are determined, the mechanism determines the VM configuration to fulfill the winning users' requests.

From the description above, we can see the increasing complexity of the workload and resource assumption along the line of description. The algorithm in [2] and [19] assume single VM and bundle VMs bid with fixed amount of available

resources. The algorithm in [20] assumes bundle VMs bid and a pool of resources. With the increasing complexity in input description, the algorithm is more complex and more efficient.

3) Social welfare maximization algorithms for clouds support game theory

In the work of [21], the authors defined a new class of games called Cloud Resource Allocation Games (CRAGs). CRAGs solve the resource allocation problem in clouds using game-theoretic mechanism. At the start of each round, all the clients submit the jobs for that round to the cloud. The provider first calculates his resource allocation vector for jobs that are running on the cloud. The provider then advertises the amount of resources left at each machine to the clients. Each client chooses the machines that would satisfy its resource need and minimize its total cost. Next, the clients will actively change their resource allocation as long as they can decrease their costs. They can follow any of the standard update mechanisms in the literature [26] to determine how they change their strategy. All the clients must follow the same update mechanism. When no client can decrease its cost by changing its strategy alone, the system has achieved a stable equilibrium.

Also applying game theory to resource allocation, however, instead of letting user take part in many allocation rounds, the work in [22] proposed a practical approximated solution with the following two steps. First, each participant solves its optimal problem independently. A Binary Integer Programming method is proposed to solve the independent optimization. Second, an evolutionary mechanism is designed to change multiplexed strategies of all initial optimal solutions with the goal of minimizing their efficiency losses. The algorithms in the evolutionary mechanism take both optimization and fairness into account.

From the description above, we can see that the algorithm in [21] is not as convenient as the algorithm in [22]. The requirement of complex users' attendance in the scheduling process makes the algorithm in [21] impractical. It should be used only as the source for further refinement. The algorithm in [22] fixed the drawback of the algorithm in [21] but its execution time could be long with the evolution mechanism.

4) Load balance algorithms for clouds support resource on demand

Several commercial and open source cloud systems use simple load balance algorithms like Round robin [2,4,23,24], random [24], least connect [4], weighted selection [23,46] as described in [1].

Round robin is among the most widely used allocation algorithms for VM initial placement. The servers are in a circular list. There is a pointer pointing to the last server that received the previous request. The system sends a new resource allocation to the next node with enough physical resources to handle the request.

In random load balance, requests are routed to servers at random. Random load balance is recommended only for homogeneous cluster deployments, where each server instance runs on a similarly configured machine. Random load balance distributes requests evenly across server instances in the cluster when the cumulative number of requests increases. Over a small number of requests, the load may not be balanced exactly evenly.

The Least Connections methods are relatively simple. The system passes a new connection to the node that has the least number of current connections. Least Connections methods work best in environments where the servers or other equipment have similar capabilities. These are dynamic load balance methods. They distribute connections based on various aspects of server performance, such as the current number of connections per node or the fastest node response time.

In many enterprise environments, server nodes of unequal processing power & performance characteristics are used to host services. It is often necessary to distribute the load based on individual server capabilities so that some servers are not unfairly burdened with requests.

Weighted Round Robin is an advanced version of the round robin that eliminates the deficiencies of the plain round robin algorithm. In the weighted round robin algorithm, one can assign a weight to each server in the group. If one server is capable of

handling twice as much load as the other, the powerful server gets a weight of 2. In such cases, the scheduler will assign two requests to the powerful server for each request assigned to the weaker one.

Similar to Weighted Round Robin, the weighted random load balance policy allows system to specify a processing load distribution ratio for each server with respect to others. In addition to the weight, endpoint selection is then further refined using random distribution based on weight. The server that has higher weight will have higher probability of receiving the request.

The work in [25] presented a scheduling strategy on load balance of VM resources based on genetic algorithm. According to historical data and current state of the system and through genetic algorithm, this strategy computes ahead the influence on the system after the deployment of the needed VM resources. It then chooses the least-affective solution, through which it achieves the best load balance and reduces or avoids dynamic migration. This strategy solves the problem of load imbalance and high migration cost by traditional algorithms after scheduling.

The work in [27] discussed resource allocation methods for large-scale Cloud Systems. The first method is Biased Random Sampling. In the large-scale Cloud system, each computing node in the cloud has a set of neighbor nodes. When a request comes, the node will choose randomly a neighbor node according to the light loaded level. The walk like that will continue k steps and the lightest load nodes will be selected for allocation.

The other method discussed in [27] is Active Clustering. Active Clustering works on the principle of grouping similar nodes together and working on these groups. Each computing node has a set of neighbor nodes. When a request comes, the initial node selects another node called the matchmaker node from its neighbors. This selection satisfies the criteria that it should be of a different type than the former one. The so-called matchmaker node then forms a connection between neighbors of it that is of the same type as the initial node. The matchmaker node then detaches the connection between itself and the initial node. The walk like that will continue k steps and the lightest load nodes will be selected to allocation.

In [28], the authors presented a load balance method for a three-level cloud-computing network: the service node, the service manager and the request manager. The proposed two-phase scheduling algorithm integrates OLB (Opportunistic Load Balancing) and LBMM (Load Balance Min-Min) to assist in the selection for effective service nodes. First, it finds all nodes having available resource > threshold. It then randomly distributes sub-tasks to those nodes (OLB algorithm). With set of sub-tasks and set of available nodes belong to a service manager, it applies LBMM. It sorts available nodes according min execution time, sorts sub-tasks according to min completion time, assigns min completion time sub-task to min execution time nodes, moves the assigned sub-task out of list, moves the assigned node to the end of the node list and repeats assign process until all sub-tasks assigned.

The work in [29] proposed a power aware load balance algorithm. The PALB algorithm first gathers the utilization percentage of each active compute node. If all compute nodes n are above 75% utilization, PALB instantiates a new virtual machine on the compute node with the lowest utilization. Otherwise, the new virtual machine (VM) is booted on the compute node with the highest utilization (if it can accommodate the size of the VM). If all currently active compute nodes have utilization over 75%, PALB sends turning on command to power on additional compute nodes (as long as there are more available compute nodes). If the compute node is using less than 25% of its resources, PALB sends a shutdown command to that node.

From the description above, pure load balance algorithms such as round robin, random, least connect are only suitable for homogeneous cloud system as they treat each computing node equally. With heterogeneous system, using those algorithms may create unbalanced state and weighted round robin or weighted random algorithms are more suitable. Algorithms in [27,28] mainly applied to large-scale cloud system. Unlike other algorithms, the algorithm

in [29] considers not only load balance, but also energy saving by turning off servers.

5) Energy efficient algorithms for clouds support resource on demand

In [30], a simple energy-aware policy incorporating allocation scheme of virtual servers is proposed to achieve the aim of green computing. The allocation schemes are popular strategy such as round robin, first fit, etc. The policy automatically governs physical hosts to a low-energy consuming state when no virtual servers are allocated in a specific physical host. It automatically manages a physical host into the operating state of full functionality when virtual servers are assigned.

The open source IaaS Cloud package Eucaplytus has integrated Power save policy as a scheduling option [31,32]. The core of the Power save policy is the First - Fit heuristic. The First- Fit method attempts to deploy a virtual machine to the first machine in a physical machine list that can accommodate this virtual machine. If no physical machine is found, then a new physical machine will be booted to host this virtual machine.

In the work of [33], the authors tried to save energy by optimizing the computing resource usage. The main idea is turning on a number of sufficient servers to meet the allocation demand. Other servers are turned off. To realize this idea, the common techniques are determining the demand and determining suitable set of available servers to meet the demand for the next period. This strategy is very similar to the idea of [34,35] applying to the generic data centre. The works in [64,65,66,67] have the same idea but does not predict the demand. Instead, the authors build cloud-managing models to determine the optimized number of server to be turned on. This optimized value must balance the energy consumption and other criteria such as high performance [65], customer impatience [66], blocking probability [64,67].

Both works in [36,37] used the same energy aware Best Fit strategy to allocate resource for the new coming VM. At first step, a power consumption model for data centre is defined. After that, the algorithm walks through all servers to determine if the server has enough resources to host the VM. With the feasible one, it estimates the future power if the VM is deployed on that machine. The feasible server with the smallest future power will be selected.

In [38], the authors proposed an algorithm for Energy aware VM consolidation. With the experiment, the authors found that there exists an optimal combination of CPU and disk utilization. At this optimal point, the energy per transaction is minimum. Next, as each request arrives, it is allocated to a server, resulting in the desired workload distribution across servers. The used heuristic maximizes the sum of the Euclidean distances of the current allocations to the optimal point at each server. If the request cannot be allocated, a new server is turned on. Then all requests are re-allocated using the same heuristic, in an arbitrary order.

From the description above, we can see the difference in saving energy mechanism of described algorithms. The work in [30] saves energy by setting servers to the lower power consumption state when there is no VM on the machine. The works in [31,32,33,65,66,67,68] try to use sufficient number of servers to host load and other free machines are turned off. The works in [36,37] go further by allocating load to the least power consumption servers. Unlike other algorithms that deal with VM, the work in [38] goes deeper with transactions distribution among VMs.

6) Number of active hosts minimization algorithms for clouds support resource on demand

The placement algorithm for VMs in a data centre allocates various resources such as memory, bandwidth, processing power, etc. from a physical machine (PM) to VMs with the goal of minimizing the number of PMs used. This problem can be viewed as a multi-dimensional packing problem. In [39] the authors discussed several heuristics to solve this problem. Each host is presented by the host's vector of capacities $H = (h_1, h_2, \ldots, h_d)$. Each VM is represented by its vector of demands $V = (v_1, v_2, \ldots, v_d)$.

The popular heuristic for bin packing problem is the First Fit Decreasing (FFD). This heuristic orders the bins and the objects in size decreasing order. Starting with the first bin, it iterates over the object. It

places objects into the first bin till no more objects can be placed into it. It then considers the first bin to be filled and proceeds to the second bin with the same procedure.

The important task for applying the FFD heuristic is determining the size of the bin and the size of the object. For cloud computing, the authors presented several ways to handle this task and thus, created several variations of the FFD heuristic.

FFDProd heuristic sorts the servers and VMs according to

$$\text{Volume(V)} = \prod_i v_i \quad (3)$$

$$\text{Volume(H)} = \prod_i h_i \quad (4)$$

FFDSum sorts the servers according to

$$\text{Volume(V)} = \sum_i w_i * v_i \quad (5)$$

$$\text{Volume(H)} = \sum_i w_i * h_i \quad (6)$$

$$w_i = \sum_{VM} \frac{v_i}{h_i} \quad (7)$$

Dot-Product heuristic - At time t let H(t) denote the vector of remaining or residual capacities of the current open host, i.e. subtract from the host's capacity the total demand of all VMs currently assigned to it. It places the VM that maximizes the dot product with the vector of remaining capacities $\sum_i w_i * v_i * h(t)_i$ without violating the capacity constraint.

Norm-based Greedy heuristic - for the l_2 norm distance metric, from all unassigned VMs, it places the VM v that minimizes the quantity $\sum w_i * (v_i - h(t)_i)^2$ and the assignment does not violate the capacity constraints.

Do not use heuristic, the work in [40] modeled the problem as the quadratic programming problem to be solved with [41] and the integer linear programming problem to be solved with a standard LP solver [42].

In [43], the author described the Global Decision Module (GDM) which is responsible for two main tasks: determining the VM allocation vectors *Ni* for each application *ai* (*VM Provisioning*), and placing these VMs on PMs in order to minimize the number of active PMs (*VM Packing*). These two phases are expressed as two *Constraint Satisfaction Problems* (CSP) which are handled by a *Constraint Solver*.

From the above description, we can see that there are two classes of algorithms. Algorithms described in [39] use heuristics and they are quite fast. Algorithms in [40,43] use global optimization techniques such as quadratic programming or linear programming. Thus, they may have long execution time.

7) Miscellaneous objectives algorithms for clouds support resource on demand

The work in [45] proposed a mechanism called CLUSTER-AND-CUT to improve the Scalability of Data Centre Networks. It first partitions VMs into VM-clusters and partitions slots into slot-clusters. VM-clusters are obtained via classical min-cut graph algorithm [2]. The data centre operators can obtain slot-clusters manually. The algorithm then maps each VM-cluster to a slot cluster. For each VM-cluster and its associated slot-cluster, it calls cluster-and-cut for a smaller problem size.

In [47], the authors recognized the serious risks of host server failures that induce unexpected downs of all hosted virtual machines and applications. To protect required high-availability applications from unpredictable host server failures, redundant configuration using virtual machines can be an effective countermeasure. The proposed method estimates the requisite minimum number of VMs according to the performance requirements of application services. It then decides an optimum VM placement so that minimum configurations survive at any k host server failures.

In [48], the authors presented a custom optimization mechanism. The mechanism allows operator to select one among following goals: Reserve a single PM for a specific VM; Minimize traffic; Spread VMs across separate PMs; Reduce the number of PMs used; Offload a specific PM. The mechanism is two-phase optimization process. During the first phase, it selects a subset of PMs called cohort with properties that best serve the VM placement. Resource is divided into level according to migration ability. The first phase algorithm gradually explores all cohort levels in search of a promising "neighborhood". In the second phase, the mechanism solves a constraint

satisfaction problem that yields a near optimal VM-to-PM mapping.

From the description above, we can see the difference in goals of those algorithms. While the work in [45] focuses on the scalability of the cloud system, algorithm in [47] tries to deal with failure by allocating redundant resources to load. The work in [48] allows custom optimization.

B. VMs migration algorithms

1) Load balance

To perform load balance, there are two main tasks: detect if the system is imbalance and perform migration.

The well-known VMware system [49] has handled the first task by comparing the "current hosts load standard deviation" (CHLSD) metric to the "target host load standard deviation" (THLSD). CHLSD is calculated as the standard deviation of the hosts' normalized entitlement value. This value of each host is calculated by dividing the sum of the entire virtual machine load on the same host by the capacity of the host as described in Formula.

$$Load = \frac{\sum VM\_entitlement}{Host\_capacity} \quad (8)$$

The operator predefines THLSD value. If the CHLSD exceeds the THLSD, the cluster is considered imbalanced. For the second task, VMware uses the following algorithm. It checks if the cluster is imbalanced (CHLSD > THLSD), simulates moving each VM from the highly load host to the lower load host to calculate CHLSD, adds migration information giving best improving CHLSD to a list and then repeats the process until CHLSD < THLSD.

In [50,51], the authors proposed the Reactive Load Balancing mechanism using local state of each PM. The mechanism detects the imbalance of each physical machine. If a PM has the resource usage > threshold for any type of resource, it is considered imbalance. VMs from that PM must be migrated to the under loaded PM. To select VM to be migrated, the work in [50] defined the parameter

$$VSR = \frac{\frac{1}{1-cpu} * \frac{1}{1-net} * \frac{1}{1-mem}}{VM_{memorySize}} \quad (9)$$

The VM having minimum VSR will be selected.

Unlike the work in [50], the work in [51] defined the parameter L for VM selection.

$$L = migration\_\cos t_{vector} * utilization_{vector} \quad (10)$$

The mechanism chooses the VM having minimum L and migrate to the PM that has least enough residual capacity.

In [52, 53], the authors proposed the Proactive Load Balancing mechanism considering the global state of the PM. For a cluster, a PM is imbalance if the coefficient of variance of PM's load > threshold. In [52], the overloaded VM from overloaded PM will be migrated to the under loaded PM. The work in [53] move under loaded VM from over loaded PM to the under loaded PM.

In [54], the authors presented the Compare and Balance algorithm. Algorithms are executed concurrently on independent physical hosts, and having limited information or no information about what the other parts of the algorithm are doing. It picks a VM_i running in the current host, picks randomly another host in the set of active hosts, calculates usage level of the current host c and the selected host c', if c>c' migrates VM_i to the selected host with the probability of c – c', and then repeats the process with other VMs.

From the description above, we can see that the load balance algorithms depend on how they define the balanced concept. The work in [49] considered the imbalance as CHLSD > THLSD. The works in [50,51,52,53] tried to detect the imbalance using a threshold value. Using the probability, the work in [54] considered that the probability of having imbalance increases along with the greater difference of resource usage of a PM compared with another PM.

2) Energy efficient

In [55], the authors proposed the *min Power Placement algorithm with History, mPPH*. *mPPH* algorithm tries to minimize migrations by migrating

as few VMs as possible with two phases. In the first phase, it determines a target utilization for each server based on the power model for the server. In the second phase, the mechanism calls the bin-packing algorithm.

The work in [56] proposed a rank-based VM consolidation method for power saving in data centres. Every PM has a server rank that is a unique value representing selection priority of the PM. Usually, the rank is determined by the data centre operator. The mechanism sets 2 values R_{high} and R_{low}. The algorithm only considers VMs in PMs having load > R_{high} and load < R_{low} as candidate for migration. The remapping process is then done with the First Fit Decreasing heuristic. VMware also has the same idea as this one for its DPM (Dynamic Power Management) module [49].

In [58], the author proposed a mechanism that uses migration in three basis activities of the system: Workload Arrival Event, Workload Departure Event and Workload Resizing Event. For insert, it uses Best-Fit algorithm. For departure, when a workload finishes its work and departs from node *x*, the algorithm reinserts the other workloads on *x*. For Workload Resizing, it can be transformed to a *Pop workload size x* and an *Insert* Procedure workload size y.

The work in [57] proposed the dynamic Round-Robin algorithm for energy efficient VM migration including two rules. In the first rule, if a VM has finished and there are still other VMs hosted on the same physical machine, this physical machine will accept no more new VM. When the rest of the virtual machines finish their execution, this physical machine can be shutdown. In the second rule, if a physical machine is in the "retiring" state for a sufficiently long period of time, it will not wait for the residing virtual machines to finish. The physical machine will be forced to migrate the rest of the virtual machines to other physical machines, and shutdown after the migration finishes.

The work in [37] proposed a heuristic for energy efficient VM migration. The optimization of the current VM allocation is carried out in two steps. At the first step, the algorithm selects VMs need to be migrated. At the second step, the chosen VMs are placed on the hosts using the modified best fit decreasing algorithm.

In [36], the authors proposed an algorithm called F4G-CG to optimize the energy consumption of data centres using VM migration. The F4G-CG algorithm has two main phases. In the first phase, the algorithm moves the VMs from low load servers to higher load servers if possible in order to free the low load server. The free low load server can be turned off. In the second phase, the algorithm moves the VMs from the old servers to the modern servers. The free old servers can be turned off.

From the description above, we can see the difference in input parameters for migrations algorithm. While the work in [55] considers all VMs of the cloud system for migration, the work in [37, 56] focus on the VMs on very high or very low load servers. The work in [58] deals with even the individual work size change. All those works do not support server turning on/off policy. The work in [57] moves VMs from low load servers to high load servers in order to turn off free servers. Also having the same idea, the work in [36] extends it by moving load from old servers to more modern servers to turn off old servers.

3) VM migration with Miscellanous objectives

The work in [60] proposed a method for Multi-objective Virtual Machine Placement in Virtualized Data Centre Environments. The objectives include minimizing total resource wastage, minimizing power consumption and minimizing thermal dissipation costs. An improved genetic algorithm with fuzzy multi-objective evaluation is proposed for efficiently searching the large solution space and conveniently combining possibly conflicting objectives.

The work in [61] proposed a method for Dynamic Placement of Virtual Machines to manage SLA Violations. The overall approach of the paper is to predict future resource requirements based on recent time series of resource demand. Based on the future resource demand, the algorithm remaps the VM to PM using best-fit algorithm.

From the description above, we can see the difference in goal as well as the mechanism of the described algorithms. The work in [60] supports multi objectives and thus, it uses time-consuming genetic algorithm. The work in [61] tries to mange SLA violation and it uses best fit heuristic with fast run-time.

V. DISCUSSION

We analyzed the difference among algorithms in each subsection of section 4. In this section, we focus on the applicability of the above studied solutions to the real environment. In the real environment, there are two main types of IaaS cloud data centres: the public and private clouds.

The public clouds or the commercial clouds provide resources to everyone having ability to afford the cost of resource usage. The trend of IaaS cloud data centers is providing wide range of products over one resource infrastructure. A represent example is Amazone EC2 [2] with spot market VM instances, reserved VM instances and resource on demand VM instances. This policy provides users flexible options of using cloud resources. Thus, it motivates users to transfer from traditional computing to cloud computing. The main goal of public cloud providers is maximizing profit. As discussed in Section 3.2.3, this goal can be gained by selling the same amount of resource with highest price or by maximizing the workload on a fixed amount of resources. For the first approach, algorithms for spot market such as [20] can be used. The second approach can be realized with energy-efficient algorithms such as [36,37]. It is because energy efficient mechanisms try to use smallest number of PMs to host workload. The literature also showed that the initial placement mechanism makes significant progress to reach the goal of the Cloud data centre compared with the VM migration mechanism. With the energy efficient initial placement algorithm, the VM migration algorithm increases the efficiency only few percent.

There should be no problem with the wide range of products policy if the total demand is smaller than the available capacity. The situation becomes more complicated if the total demand is larger than the available capacity. In this situation, how to allocate resources among many products with profit optimization is still an open issue. Another situation is that there is a peak of demand in a short period. Reserving enough resources to deal with this situation may lead to inefficient resource usage. An initial possible solution to this issue is proposed in [14] with the addition of bets-effort VMs. However, a detail study for a robust scenario like Amazone [2] is still necessary.

The private clouds provide resources to users inside the border of an organization. Depending on the policy of the organization, the goal of the private clouds may be different. With each goal, the system can apply different solutions. In general, the set of resource allocation algorithms for both initial VMs placement and VM migration can be divided into two classes: simple heuristics and application of complex algorithms. For example, to make initial VMs placement with the goal of load balance, the system can use simple heuristic such as Round robin [2,4,23,24], random [24], least connect [4], weighted selection [23,46] or apply genetic algorithm [25]. The simple heuristics have the advantage of fast execution and easy to implement. The application of complex algorithms usually has better performance. However, they are slower and more complicated to implement. It seems that simple heuristics are preferred in real systems [1].

VI. CONCLUSION

Cloud computing is the promising model for delivering IT services as computing utilities. The resource allocation module is an important part of each IaaS Cloud system. In this paper, we have studied and classified different algorithm to map virtual machines to physical machines inside the cloud computing systems. Recent research developments have been discussed and categorized over the execution phases, business models and goals of resource allocation.

Efficient resource allocation in IaaS Cloud computing systems is a well known and extensively studied in the past problem. The allocation decision is made for both homogeneous as well as heterogeneous

IaaS cloud infrastructure under different business model such as spot market, game theory, resource reservation and resource on demand. The proposed allocation algorithms range from simple heuristics to applications of well-known methods such as genetic algorithm, Linear Programming, Constrain Satisfaction Programming, etc. We also discussed the applicability of studied solutions to two main Cloud data centre types: public Cloud (or commercial Cloud) and private Cloud. From the analysis, open issues and future direction are stated.

REFERENCES

[1] Rimal, B.P., Choi, E., Lumb, I., 2009, A Taxonomy and Survey of Cloud Computing Systems, Proceeding of the Fifth International Joint Conference on INC, IMS and IDC, pp. 44 – 51.

[2] http://aws.amazon.com/ec2/

[3] http://www.enki.co/

[4] http://www.gogrid.com/

[5] http://www.engineyard.com/products/cloud

[6] http://www.google.com/apps/intl/en/business/cloud.html

[7] http://www.netsuite.com/portal/home.shtml

[8] http://www.salesforce.com/ap/?ir=1

[9] https://developers.google.com/appengine/

[10] Endo, P. T., Gonçalves, G. E., Kelner, J., Sadok, D., 2010, A Survey on Open-source Cloud Computing Solutions, Proceedings of the 28th edition of the Brazilian Symposium on Computer Networks and Distributed Systems (SBRC 2010), pp. 3-16.

[11] Beloglazov, A., Buyya, R., Lee, Y. C., Zomaya, A. Y., 2011, A Taxonomy and Survey of Energy-Efficient Data Centers and Cloud Computing Systems. Advances in Computers 82, pp. 47-111.

[12] Teng, F., 2012, MANAGEMENT DES DONN´EES ET ORDONNANCEMENT DES T`ACHES SUR ARCHITECTURES DISTRIBU´EES, PhD thesis of E´COLE CENTRALE PARIS ET MANUFACTURES.

[13] Basmadjian, R., Ali, N., Niedermeier, F., Meer H. d., and Giuliani, G., 2011, A Methodology to Predict the Power Consumption for Data Centres, Proceedings of e-Energy 2011, pp. 1-10.

[14] Sotomayor, B., Keahey, K., Foster, I. T., 2008, Combining batch execution and leasing using virtual machines, Proceedings of HPDC 2008, pp. 87-96.

[15] Lifka, D. A., 1995, The ANL/IBM SP scheduling system, Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing, IPPS '95, pp. 295–303.

[16] Mu'alem, A. W., and Feitelson, D. G., 2001, Utilization, predictability, workloads, and user runtime estimates in scheduling the IBM SP2 with backfilling. IEEE Trans. Parallel Distrib. Syst., 12(6), pp. 529–543.

[17] Wang, X., 2011, Research on Adaptive QoS-Aware Resource Reservation Management in Cloud Service Environments, Proceedings of 2011 IEEE Asia-Pacific Services Computing Conference (APSCC 2011), pp. 147 – 152.

[18] Zhao, M., and Figueiredo, R. J., 2007, Experimental study of virtual machine migration in support of reservation of cluster resources, Proceedings of the 2nd international workshop on Virtualization technology in distributed computing, pp. 1-8.

[19] Zaman, S., and Grosu, D., 2010, Combinatorial auction-based allocation of virtual machine instances in clouds, Proceedings of the 2nd IEEE Intl. Conf. On Cloud Computing Technology and Science, pp. 127–134.

[20] Zaman, S., and Grosu, D., 2011, Combinatorial Auction-Based Dynamic VM Provisioning and Allocation in Clouds, Proceedings of CloudCom 2011, pp. 107-114.

[21] Jalaparti, V., Nguyen, G. D., Gupta, I., Caesar, M., 2010, Cloud Resource Allocation Games, Illinois Technical Report, pp. 124-133.

[22] Wei, G., Vasilakos, A. V., Zheng, Y., Xiong, N., 2010, A game-theoretic method of fair resource allocation for cloud computing services, J Supercomputer v. 54, pp. 252–269.

[23] http://opennebula.org/

[24] http://www.enomaly.com/

[25] Hu, J., Gu, J., Sun, G., and Zhao, T., 2010, A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment, Third International Symposium on Parallel Architectures, Algorithms and Programming (PAAP), pp. 89-96.

[26] Nisan, N., Roughgarden, T., Tardos, E., and Vazirani, V. V., 2007, Algorithmic Game Theory. Cambridge University Press.

[27] Randles, M., Lamb, D., and Taleb-Bendiab, A., 2010, A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing, Proceedings of 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, pp. 551-556.

[28] Wang, S., Yan, K., Liao, W., and Wang, S., 2010, Towards a Load Balancing in a Three-level Cloud Computing Network, Proceedings of the 3rd IEEE International Conference on Computer Science and Information Technology (ICCSIT), pp. 108-113.

[29] Galloway, J. M., Smith, K. L., Vibsky, S. S., 2011, Power Aware Load Balancing for Cloud Computing, Proceedings of WCECS2011, pp.127-132.

[30] Do, T. V., 2011, Comparison of Allocation Schemes for Virtual Machines in Energy-Aware Server Farms, The Computer Journal 54(11), pp. 1790-1797.

[31] http://open.eucalyptus.com/.

[32] Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D., 2009, The eucalyptus open-source cloud-computing system, Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid, CCGRID '09, pp. 124–131.

[33] Mazzucco, M., Dyachuk, D., and Deters, R., 2010, Maximizing cloud providers' revenues via energy aware allocation policies, Proceedings of the IEEE International Conference on Cloud Computing, pp.131–138.

[34] Lubin, B., Kephart, J. O., Das, R., Parkes, D. C., 2009, Expressive Power-Based Resource Allocation for Data Centers, Proceedings of the 21st international joint conference on Artifical intelligence, pp.1451-1456.

[35] Chase, J. S., Anderson, D. C., Thakar, P. N., Vahdat, A. M., Doyle, R. P., 2001, Managing energy and server resources in hosting centers, ACM SIGOPS Operating Systems Review, vol. 35, nr. 5, pp. 103-116.

[36] Quan, D. M., Basmadjian, R., Meer, H. d., Lent, R., Mahmoodi, T., Sannelli, D., Mezza, F., Telesca, L., Dupont, C., 2011, Energy Efficient Resource Allocation Strategy for Cloud Data Centres, Proceedings of ISCIS 2011, pp. 133-141.

[37] Beloglazov, A., Abawajy, J. H., Buyya, R., 2012, Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing, Future Generation Comp. Syst. vol.28. nr. 5, pp. 755-768.

[38] Srikantaiah, S., Kansal, A., Zhao, F., 2008, Energy aware consolidation for cloud computing, Proceedings of the 2008 conference on Power aware computing and systems, pp. 1-10.

[39] Lee, S., Panigrahy, R., Prabhakaran, V., Ramasubrahmanian, V. , Talwar, K., Uyeda, L. and Wieder, U., 2011, Validating Heuristics for Virtual Machine Consolidation, Microsoft Research, MSR-TR-2011-9, pp. 1-14.

[40] Bellur, U., Rao C., and Kumar, M., 2010, Optimal Placement Algorithms for Virtual Machines, Proceedings of CoRR, pp.103-110.

[41] Kozlov, M. K., Tarasov, S. P., and Khachiyan, L. G., 1980, The polynomial solvability of convex quadratic programming, USSR Computational Mathematics and Mathematical Physics, vol. 20. nr.5, pp. 223–228.

[42] lp-solve. http://lpsolve.sourceforge.net/5.5/

[43] Van, H., and Tran, F., 2009, Autonomic resource management for service host platforms, Proceedings of Workshop on Software Engineering Challenges in Cloud Computing, pp. 1-8.

[44] Meng, X., Pappas, V., and Zhang, L., 2010, Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement, Proceedings of IEEE 2010 INFOCOM, pp. 1-9.

[45] http://en.wikipedia.org/wiki/Gomory%E2%80%93Hu_tree

[46] Chandrasekaran, B., Purush, R., Douglas, B., and Schmidt, D., 2007, Virtualization Management Using Microsoft System Center and Dell OpenManage, Dell Power Solutions, pp. 40-44.

[47] Machida, F., Kawato, M., and Maeno, Y., 2010, Redundant Virtual Machine Placement for Fault-tolerant Consolidated Server Clusters, Proceedings of the 12th IEEE/IFIP Network Operations and Management Symposium, pp. 32-39.

[48] Tsakalozos, K., Roussopoulos, M., and Delis, A., 2011, VM Placement in non-Homogeneous IaaS-Clouds, Proceedings of 9th International Conference on Service Oriented Computing (ICSOC 2011), pp. 172-187.

[49] Epping, D., Denneman, F., 2010, VMware vSphere 4.1 HA and DRS Technical Deepdive, CreateSpace, ISBN-10: 1456301446.

[50] Wood, T., Shenoy, P., and Arun, 2007, Black-box and gray-box strategies for virtual machine migration, NSDI 2007, pp. 229–242.

[51] Khanna, G., Beaty, K., Kar, G., and Kochut, A., 2006, Application performance management in virtualized server environments, Proceedings of 10th IEEE/IFIP Network Operations and Management Symposium NOMS 2006, pp. 373 –381.

[52] Arzuaga, E., and Kaeli, D. R., 2010, Quantifying load imbalance on virtualized enterprise servers, Proceedings of the first joint WOSP/SIPEW international conference on Performance engineering, pp. 235–242.

[53] Singh, A., Korupolu, M., and Mohapatra, D., 2008, Server-storage virtualization: Integration and load balancing in data centers, Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis, pp. 1 –12.

[54] Zhao, Y., and Huang, W., 2009, Adaptive Distributed Load Balancing Algorithm based on Live Migration of Virtual Machines in Cloud, Proceedings of 5th IEEE International Joint Conference on INC, IMS and IDC, pp. 170-175.

[55] Verma, A., Ahuja, P., and Neogi, A., 2008, pMapper: Power and Migration Cost Aware Application Placement in Virtualized Systems, Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware, pp. 243-264.

[56] Takeda S., and Takemura, T., 2010, A rank-based vm consolidation method for power saving in datacenters. Information and Media Technologies, vol. 5, nr. 3, pp. 994-1002.

[57] Lin, C. C., Liu, P., Wu, J. J., 2011, Energy-efficient Virtual Machine Provision Algorithms for Cloud Systems, 2011 Fourth IEEE International Conference on Utility and Cloud Computing, pp.81-88.

[58] Li, B., Li, J., Huai, J., Wo, T., Li, Q., Zhong, L., 2009, EnaCloud: An Energy-saving Application Live Placement Approach on Cloud Computing

Environments, IEEE International Conference on Cloud Computing, 2009. CLOUD '09, pp. 17 - 24.
[59] Lee, C. C., and Lee, D. T., 1985, A simple on-line bin-packing algorithm. Journal of the ACM, 32(3), pp. 562-572.
[60] Xu, J., and Fortes, J., 2010, Multi-objective Virtual Machine Placement in Virtualized Data Center Environments, Proceedings of the 2010 IEEE/ACM Conference on Green Computing and Communications, pp. 179-188.
[61] Bobroff, N., Kochut, A., and Beaty, K., 2007, Dynamic Placement of Virtual Machines for Managing SLA Violations, Proceedings of the 10th IFIP/IEEE Symposium on Integrated Network Management, pp. 119-128.
[62] Waldspurger, C. A., 2002, Memory Resource Management in VMware ESX Server, ACM SIGOPS Operating Systems Review - OSDI '02: Proceedings of the 5th symposium on Operating systems design and implementation, pp. 181-194.
[63] Silpa, CS., Basha, S. S. M., 2013, A Comparative Analysis of Scheduling Policies in Cloud Computing Environment. International Journal of Computer Applications 67(20), pp. 16-24.
[64] Do, T. V., Rotter, C., 2012, Comparison of scheduling schemes for on-demand IaaS requests. Journal of Systems and Software 85(6), pp. 1400-1408.
[65] Mitrani, I., 2013, Managing performance and power consumption in a server farm. Annals OR 202(1), pp. 121-134.
[66] Mitrani, I., 2011, Service center trade-offs between customer impatience and power consumption. Perform. Eval. 68(11), pp. 1222-1231.
[67] Do, T. V., Krieger, U. R., 2009, A Performance Model for Maintenance Tasks in an Environment of Virtualized Servers. In: IFIP/TC6 NETWORKING 2009, pp. 931-942.

AUTHOR' BIOGRAPHY

Dang Minh Quan is a lecturer at the Institute of Information Technology for Economic, National Economics University, VietNam. He received his Ph.D. (2006) from the University of Paderborn, Germany. His current research centers on energy saving for data centers. In particular, he puts special focus on designing energy efficient algorithms for traditional data centers, cloud data centers and HPC data centers.