# A Novel Multi-Objective Optimization Scheme for Rebalancing Virtual Machine Placement

Rui Li*†, Qinghua Zheng *†, Xiuqi Li‡, Jie Wu‡

*MOE Key Lab for Intelligent Networks and Network Security, Xi'an Jiaotong University, Xi'an, China
†Department of Computer Science and Technology, Xi'an Jiaotong University, Xi'an, China
‡Department of Computer and Information Sciences,Temple University,Philadelphia, PA, USA

***Abstract*—In cloud computing, load balancing is a very important performance factor. Frequent addition and removal of Virtual Machines (VMs) can cause load imbalance across Host Machines (HMs). Therefore, redistribution of VMs to different HMs needs to be performed periodically. This is called VM load rebalancing (VMrB). Existing VMrB solutions consider either balancing across HMs (inter-HM) or balancing within each individual HM (intra-HM), but not both. In this paper, we give a systematic review of existing VMrB literature, and present a VMrB scheme named MOVMrB that optimizes load balancing of multiple resources across HMs and in each individual HM. Our solution is a true multiple-objective optimization approach that does not scalarize multiple resources into one measurement. Instead, we treat each resource as a separate dimension and convert the load balancing of multiple resources to a complex system optimization problem. In addition, we propose a hybrid VM live migration algorithm that can dramatically speed up the VMrB process. Extensive evaluations using synthetic and real data have been conducted to test the proposed solutions against existing VMrB studies. The results show that our scheme can achieve a more balanced inter-HM and intra-HM load in a more efficient way. To the best of our knowledge, this is the first approach that considers both inter-HM and intra-HM load balancing and does not scalarize multiple resources into one measurement.**

## I. Introduction

Cloud computing is an on-demand computing model for users to share a collection of configurable computing resources and have ubiquitous and on-demand access to these resources. Hardware virtualization creates virtual machines (VM) that are like real computers with operating systems and can run as applications on physical machines (PM), also called host machines (HM). Multiple VMs can run on the same HM. Virtualization is a core technology in cloud computing due to its many advantages, such as reduced power consumption and system cost, higher resource utilization, better application portability, faster deployment, increased system security and reliability, etc. [1]

VMs are created at users' requests. Because of the difficulty in predicting the behaviors of a wide variety of applications running on VMs and the presence of burst in application traffic, users pay each VM instance by size in resource units, not by the actual resource consumption. The paid resource units are usually guaranteed unless there is a very high demand. For example, all pricing options (on-demand, reserved, spot) in Amazon EC2 bill each instance of a particular type hourly. Each instance type is a unique combination of CPU, memory, storage, and network capacity. In this paper, the load of a VM is the amount of resources allocated to it. And the load of a HM is the total load of all VMs running on it.

The load balancing technology in cloud computing attempts to distribute VM instances across a set of coordinating HMs such that no HM is overloaded or under-loaded. Load balancing increases the scalability and availability of cloud computing systems and is a very important design consideration [2]. There are two sub-issues in load balancing: incremental VM load balancing (VMiB) and VM load rebalancing (VMrB). VMiB deals with the on-line balancing needs when cloud users come and the VM instances are dynamically requested and deployed. To ensure service quality and user satisfaction, quick response is the key in VMiB and load balance may not be well achieved. VMrB is similar to server consolidation [3]. Due to the on-demand nature of cloud computing, many users come and go and many VMs are frequently added and deleted. This leads to the imbalance of VMs across HMs after certain time period. VMrB re-computes the placements of VMs on HMs off-line for better load balance. VM live migration completes the actual VM redistribution without causing too much service interruption. We study the VMrB problem in this paper.

There are two aspects of VMrB load balancing. From the entire system perspective, the total amount of any resource demanded by VMs should be spread evenly across different cooperating HMs. We term this *inter-HM (horizontal) load balancing*. On the other hand, from the perspective of an individual HM, to satisfy future VM resource requests, we need to balance the amount of residual resources in each resource type. A HM with much CPU but little memory left would be able to satisfy few VM resource requests. We term this *intra-HM (vertical) load balancing*. Most research efforts focus only on inter-HM load balancing. The others study only intra-HM load balancing. No research has combined both inter-HM and intra-HM load balancing.

The challenge in inter-HM load balancing is that given multiple resources (CPU, memory, storage, etc), as is always the case in the real world, how to achieve a good load balance of each resource across different HMs. When there is a single resource, we can always have a total ordering of candidate solutions and choose the best. However, when there are multiple resources, there may not exist an overall-best solution that reaches the optimal balance along each resource dimension. The problem gets even more complicated when a large number of HMs and many resources have to be considered.

Some researchers considered only one resource [4], [5].

Others attempted to meet the challenge by using multiple objective optimization methods to solve the multiple resource inter-HM load balancing problem [1], [6]–[10]. Each objective is the optimization of a resource load. However, they avoided the complexity in the original problem by combining multiple objectives into a single scalarized objective and then optimizing this single objective. This is essentially combining multiple resource loads into one composite load and then balancing this composite load instead of each individual resource load. Scalarization does not always guarantee Pareto optimality [11]. No existing work has directly tackled the problem using a true multiple objective optimization method.

In this paper, we study these two unexplored territories and propose a VMrB solution called MOVMrB that optimizes the load balancing of multi-dimensional resources both across different HMs and within each individual HM. Our solution is a true multiple-objective optimization approach. We treat each resource as a separate dimension and solve the multiple resource load balancing problem using a complex system optimization method. In addition, we design a hybrid live VM migration technique that reduces the migration cost via an interval optimization method and decreases the number of VM migrations by avoiding useless VM migrations.

We also give a systematic review of the current research in load balancing in cloud computing. We categorize existing research methods using multiple classification taxonomies. A significant number of experiments have been conducted to evaluate the proposed work against existing similar ones using both the synthetic data and real-world data sets. The results demonstrate that our VMrB solution can achieve a more balanced inter-HM load and intra-HM load in a more efficient way. And the proposed hybrid VM migration approach dramatically expedites the VM migration process, 9 to 35 times faster, to be exact.

The rest of the paper is organized as follows. In Section II, we classify and review the current and past research in VMrB load balancing. In Section III, we present our MOVMrB framework in great detail. The experimental setup and results are discussed in Section IV. The paper is summarized in Section V.

## II. Related Work

The existing research in VMrB can be classified into two categories: *inter-HM (horizontal) load balancing* and *intra-HM (vertical) load balancing*.

### A. Inter-HM Load Balancing

The researches on inter-HM load balancing can be classified into two categories based on the number of resources considered: single-dimensional and multi-dimensional.

**Single-dimensional load balancing**. Only one resource is considered. It can be CPU, memory, or network bandwidth. Network bandwidth is considered in [4]. [5] minimizes the variance of CPU utilization.

**Multidimensional load balancing**. Multiple resources are considered. The balancing problem has been converted into a single objective problem based on four load scalarization methods.

(1) *Weighted Model*. Weighted sum [6] and weighted product [7] are widely used. In [7], the volume of a HM or VM is defined as the product of its CPU, network and memory loads. [6] takes weighted sum of CPU and memory utilization. However, the weight coefficients in these researches are closely related to the specific applications, and may be easily influenced by subjective factor.

(2) *Minmax Model*. In [8], the load is scalarized by uniform norm of multidimensional resources utilization. As variation ranges of different loads may be different, the imbalance in some resources may be ignored in the minmax model.

(3) *Constraint Model*. [1] and [9] convert the multi-objective optimization problem into a constrained single-objective optimization problem, the memory load is regarded as one of the constraints, and the measure of CPU load is regarded as the objective function. However, as different application types may require different cost model (e.g. CPU-intensive or memory-intensive), there does not exist a unified schema for all application types.

(4) *Ideal Point Model*. [10] introduces skewness to mix loads with different resource requirements together, which is defined as variance around the resource vectors. However, this kind of load balancing policies based on sorting hosts is not heuristic enough and has a relatively complex computation.

Another classification is MinMax strategy [1], [7]–[9] or Minimum Variance strategy [6], [10], based on the method to quantify the evenness over HMs. MinMax strategy is used in VMrB to achieve load balance status by minimizing the maximum of the HMs' loads. However, if some big size VMs exist, the effect of MinMax strategy will be weakened more or less. Minimum Variance is used to minimize the variance of HMs' loads to ensure the loads fluctuated within a narrow range. However, the premise of comparing solutions using variance value is that the mean values of these solutions need to be equal. The mean values may vary with the number of used HMs, which may change for the reason of energy saving, utility improving, etc.

### B. Intra-HM Load Balancing

Due to the multiple dimensionalities of physical resources, there always exists a waste of resources, which results from the imbalanced use of multi-dimensional resources. [12] balance the multidimensional resources of each server by minimizing the sum of differences between the smallest scalarized residual resource and each of the other residual resources. In [13], the equilibrium is represented by the ratio of differences between the residual resource pairs to sum of the used resource pairs. In [14], the equilibrium is represented by RIV which refers to the standard deviation of multi-dimensional resources in each HM.

All VMrB solutions above can be classified into deterministic algorithm (including binary search [8], [9], Best Fit Decrease (BFD) [1], [7], [10], [14]) or non-deterministic algorithm (including genetic algorithm [6], [12] and ant colony algorithm [4], [13]). A deterministic algorithm has a faster

convergence speed but easily falls into a local minimum. A non-deterministic algorithm has the ability of finding the global optimum but with a relatively slow convergence speed, especially when the scale of problem is large.

## III. The MOVMrB Framework

In this section, we first explain our new mathematical model of the VMrB problem, including the major components: inter-HM load unevenness, intra-HM load disequilibrium, hybrid live VM migration cost, and the model itself as a whole. Then we detail the steps in the hybrid live VM migration algorithm and the VMrB load rebalancing algorithm.

The load of a VM or a HM is represented by a multi-dimensional vector. Each dimension refers to the relative utilization of a specific resource requested by a VM or located on a HM, which is named VM load and HM load, respectively. Assuming that a shared storage system [15] is used, only CPU, memory, and bandwidth loads are considered in this study.

### A. Inter-HM Load Unevenness

The load of each resource $r$ on different HMs may be very different. This imbalance of resource $r$ across different HMs is denoted $unevenness(r)$ and computed using the Coefficient of Variance (CV) theory.

$$unevenness(r) = \sqrt{M \cdot \sum_{j=1}^{M} \left(\frac{L_j^r}{\overline{L^r}} - 1\right)^2}, \tag{1}$$

where $M$ is the number of used HMs. Let $l_i^{CPU}$, $l_i^{MEM}$ and $l_i^{NET}$ be the CPU, memory and network load of the VM $i$, respectively. $L_j^r = \sum_{i=1}^{N} l_i^r \cdot x_{ij}$ is the load of resource $r$ in the HM $j$, and $\overline{L^r} = \frac{1}{M}\sum_{j=1}^{M}\sum_{i=1}^{N} l_i^r \cdot x_{ij}$ is the average load of resource $r$ over all HMs. $N$ is the number of VMs and the value of $x_{ij}$ shows whether VM $i$ is placed on $j$ (value 1) or not (value 0). Equation (1) quantifies the relative variation in the amount of resource $r$ used across different HMs.

### B. Intra-HM Load Disequilibrium

The amount of different resources left on each HM may be different. To accommodate future VM instances, we need to strive to balance the amount of residual resources along each resource dimension on any individual HM. We use $disequilibrium(j)$ to quantify the imbalance in the relative utilization of multiple resources on HM $j$. Let $\overline{L_j} = \frac{1}{R}\sum_{r=1}^{R}\sum_{i=1}^{N} l_i^r \cdot x_{ij}$ be the average load of all resources on HM $j$; $R$ refers to the number of resources in consideration.

$$disequilibrium(j) = \sqrt{\frac{1}{R} \cdot \sum_{r=1}^{R} \left(\frac{L_j^r - \overline{L_j}}{\overline{L_j}}\right)^2}, \tag{2}$$

### C. Hybrid Live VM Migration Cost

Live VM migration process must be fast enough to avoid noticeable disruption and performance degradation of running services. The pre-copy approach [16] is a popular live VM migration technique that includes an iterative pre-copy phase followed by a minimal stop-and-copy phase.

The cost of live VM migration depends on the migration time for pre-copy, the down time for stop-and-copy, and the time for non-memory-transfer operations called overheads [17]. Both the migration time and the down time are determined mainly by the amount of memory to transfer and the speed of the network used for migration. The overhead cost is fixed, but significant in particular when the network is fast. The entire migration cost is modeled as follows.

$$MigCost_{ij} = |x_{ij} - x'_{ij}| \left(Overheads + Time_i^{mig} + Time_i^{down}\right), \tag{3}$$

$$\frac{l_i^{MEM}}{LinkSpeed} \leq Time_i^{mig} \leq \frac{(w+2) * l_i^{MEM}}{LinkSpeed}, \tag{4}$$

$$0 \leq Time_i^{down} \leq \frac{l_i^{MEM}}{LinkSpeed}, \tag{5}$$

where binary variables $x'_{ij}$ and $x_{ij}$ indicate whether VM $i$ is placed on HM $j$ before and after rebalancing. $Time_i^{mig}$ is the migration time. The pre-copy terminates after the VM's entire RAM is transferred more than $w$ times. Therefore $Time_i^{mig}$ is bounded by the two values given in Eq. (4). $Time_i^{down}$ is the down time and also has two bounds, as stated in Eq. (5) . *Overheads* is the time for overhead operations. We set $w$ to 3, and *Overheads* to 314ms as suggested in [17].

Our proposed hybrid migration strategy extends the basic pre-copy approach by integrating the information about the network for migration into the design. The above base live VM migration cost model is therefore adjusted accordingly. The new hybrid live VM migration cost is denoted as $HybridMigCost()$. The computation of this new cost will be discussed in detail together with the hybrid live VM migration algorithm in Subsection E.

### D. VMrB Problem Formulation

Given $N$ VMs to be re-distributed on $M$ HMs for load rebalancing, assume that all resources needed by a VM can be satisfied by a single HM. Let $T_j^{CPU}$, $T_j^{MEM}$ and $T_j^{NET}$ denote the maximum amount of used CPU, memory, and network bandwidth of HM $j$ respectively. Let $y_j$ represent whether HM $j$ is in use (value 1) or not (value 0). The VMrB problem is to simultaneously meet all the objectives listed in Eqs. (6)-(10) without violating any of the constraints listed in Eqs. (11)-(15).

$$\text{Minimize }: unevenness(cpu), \tag{6}$$

$$\text{Minimize }: unevenness(memory), \tag{7}$$

$$\text{Minimize }: unevenness(bandwidth), \tag{8}$$

$$\text{Minimize }: \sum_{j=1}^{M} disequilibrium(j), \tag{9}$$

$$\text{Minimize }: HybridMigCost(), \tag{10}$$

Subject to:

$$\sum_{j=1}^{M} x_{ij} = 1, i = [1, ..., N], \tag{11}$$

$$\sum_{i=1}^{N} l_i^{CPU} \cdot x_{ij} \leq T_j^{CPU} \cdot y_j, j = [1, ..., M], \tag{12}$$

$$\sum_{i=1}^{N} l_i^{MEM} \cdot x_{ij} \leq T_j^{MEM} \cdot y_j, j = [1, ..., M], \tag{13}$$

$$\sum_{i=1}^{N} l_i^{NET} \cdot x_{ij} \leq T_j^{NET} \cdot y_j, j = [1, ..., M], \tag{14}$$

$$y_j, x_{ij} \in \{0, 1\}, i = [1, ..., N] \text{ and } j = [1, ..., M], \tag{15}$$

The first three objectives are to minimize inter-HM load imbalance: the unevenness in the use of each single resource (CPU, memory, or bandwidth) across HMs. The fourth

objective is to minimize the intra-HM load imbalance: the disequilibrium in the use of different resources within each single HM.

The first constraint in Eq. (11) ensures that each VM resides in only one HM. The next three constraints make sure that the total usage of each resource by all VMs running on any single HM does not exceed the upper bound (lower than 100%) set for that resource on that HM. The last constraint specifies that $y_j$ and $x_{ij}$ must be binary variables. Therefore any single HM is either used or not and any VM is either running on a single HM or not.

### *E. Hybrid Live VM Migration Algorithm*

Given a set of $N$ VMs to be lively migrated over $M$ HMs, the existing research in live VM migration are either serial migration or parallel migration. In a serial migration, the migration of these VMs are carried out in sequence. The migration of the next VM will not start until the migration of the previous VM is completed. Each data transfer is operated at the full channel bit rate. The total live migration cost for all VMs is $\sum_{i=1}^{N}\sum_{j=1}^{M} MigCost_{ij}$ [18].

In a parallel migration, the migration of multiple VMs are carried out concurrently [18], [19]. The results in [19] show that parallel migration is faster than serial migration, in particular when a large number of VMs are migrated together. The benefit comes as a result of overlapping the suspend and resume phases of multiple VMs. However, this performance statement is based on the assumption that VMs have small memory capacity and slow page dirty rate. The evaluations in [18] demonstrate that sequential migration is better than concurrent migration in terms of the migration time. With the increase in concurrency granularity, the migration time of each node also increases significantly because the network for migration becomes the bottleneck.

**Algorithm 1** HybridMigration

1: Filter invalid migrations and set migration batch $mb$=0;
2: **for** $k = 1$ to $K$ **do**
3: **if** this $VM_k$ has not been arranged **then**
4: Find out the set $VMC_k$ of VMs satisfying the following three conditions:
(1) Each VM has not been arranged;
(2) The transfer route of each VM does not cross with $VM_k$;
(3) The transfer route of each VM does not cross with each other;
5: Set both $VM_k$ and each VM in $VMC_k$ to be arranged;
6: $mb++$;
7: Set the migration sequence number of $VM_k$ and each VM in $VMC_k$ to be $mb$;
8: **end if**
9: **end for**
10: Find the $VM_{mb}$ with the maximum memory $l_{mb}^{MEM}$ in migration sequence $mb$;
11: $HybridMigCost^{lowerBound}$=$\sum(Overheads + \frac{l_{mb}^{MEM}}{LinkSpeed})$
$HybridMigCost^{upperBound}$=$\sum(Overheads + \frac{(w+3)*l_{mb}^{MEM}}{LinkSpeed})$

In this study, we propose a hybrid live VM migration algorithm that utilizes the information about the network for migration. It is based on the observation that full duplex Ethernet is widely deployed in data centers. In our hybrid design, we have each link carry the data transfers of two simultaneous live VM migrations. Each transfer route is dedicated to only one live VM migration process. The live migrations of multiple VMs are executed concurrently on different transfer routes. The details on how to arrange the transfer routes for a set of VMs and HMs are described in Algorithm 1.

The problem of minimizing the hybrid migration cost in Eq. (10) is an interval optimization problem: $Minimize{<}HybridMigCost^{lowerBound}, HybridMigCost^{upperBound}{>}$. As $HybridMigCost^{upperBound} = HybridMigCost^{lowerBound} + \sum\frac{(w+2)*l_{mb}^{MEM}}{LinkSpeed}$, we can simplify the two objectives into one: *Minimize* $HybridMigCost^{lowerBound}$.

Our hybrid design also includes an invalid migration filter. The filter is used to exclude the scenario where the two HMs hosting two VMs with the same resource demands are exchanged after the live migration. After removing invalid VM migrations, the hybrid migration cost is computed using Eqs. (3)-(5).

We assume that a 10-gigabit Ethernet is dedicated for the sole purpose of live VM migration, and is separated from the network for normal business operations. This is consistent with the system configuration in VMware ESXi, which assigns a VMkernel network for VMotion. It is also assumed that the network connecting HMs is a *Tree* architecture consists of access switches and aggregate switches, as shown in [20]. Each aggregate switch is connected to two access switches, each of which further connects to five HMs.

### *F. MOVMrB VM Rebalancing Algorithm*

The VMrB problem is formulated as a multiple objective optimization problem in Subsection D. To find the corresponding Pareto optimal solutions, we do not scalarize multiple objectives into one. Instead we regard the VMrB problem as a true multiple objective complex system optimization issue and solve it using BBO/Complex [21]. BBO/Complex is an extension to the basic BBO, which is different from other population-based optimization methods [22].

MOVMrB converts VMrB to a complex system that has multiple subsystems. Each subsystem $S_k$ performs self-optimization to meet its own objectives subject to its own constraints. Subsystems also help each other to improve by sharing information mutually. Together they make the entire complex system optimized.

We decompose the complex system using the optimization objectives. There are two types of subsystems: *Subs-H* and *Subs-V*. *Subs-H* is used to minimize the horizontal (inter-HM) load imbalance (Eqs. (6)-(8)) and the hybrid migration cost (Eq. (10)). *Subs-V* is intended to minimize the vertical (intra-HM) load disequilibrium (Eq. (9)) and the hybrid migration cost. Any subsystem must be classified as either *Subs-H* or *Subs-V*. All constraints in Eqs. (11)-(15) must be satisfied by each and every subsystem.

MOVMrB optimizes the VMrB complex system using BBO/Complex. A subsystem $S_k$ corresponds to an archipelago $A_k$ of islands. Each island is a candidate VMrB solution, i.e. a rebalanced VM placement matrix. We will use island and matrix interchangeably in the remaining sections of the paper. There are two types of archipelagos, *Arch-H* and *Arch-V*, corresponding to the subsystem types, *Subs-H* and *Subs-V*.

Let $E = \{A_1^H, A_2^H, \ldots, A_h^H, A_1^V, A_2^V, \ldots, A_t^V\}$ denote an ecosystem of $h + t$ archipelagos.

$A_k^H = \{I_{k1}^H, I_{k2}^H, \ldots, I_{kT}^H; O_1, O_2, O_3, O_5; C_1, C_2, C_3, C_4, C_5\}$ represents an arbitrary archipelago of *Arch-H*, which contains $T$ islands, four objective, and five constraints. $A_k^V = \{I_{k1}^V, I_{k2}^V, \ldots, I_{kT}^V; O_4, O_5; C_1, C_2, C_3, C_4, C_5\}$ represents an arbitrary archipelago of *Arch-V*, which contains $T$ islands, two objective, and five constraints. The five objectives $O_1, O_2, O_3, O_4, O_5$ correspond to Eqs.6 to 10. The five constraints $C_1, C_2, C_3, C_4, C_5$ correspond to Eqs.11 to 15. All islands in the same archipelago have the same objectives and same constraints.

Each island is defined by a vector of $N$ SIVs. E.g. island $kt$ in archipelago *Arch-H* $A_k^H$ is denoted by $I_{kt}^H = [SIV_{kt1}^H, SIV_{kt2}^H, \ldots, SIV_{ktN}^H]$, where $k \in \{1, 2, ..., h\}$ , $t = 1, 2, \ldots, T$. Each SIV, $SIV_{ktN}^H$, is an integer that refers to the index of a HM hosting a VM. The HSI (Habitat Suitability Index) of an island $I_{kt}^H$ is denoted by $HSI_{kt}^H$. It represents the goodness of a candidate solution.

The MOVMrB algorithm includes three major operations: initialization, subsystem optimization, and elitists selection. The last two operations are performed in sequence in each iteration. The algorithm stops when the termination criteria is met.

**Initialization**. The initial population (placement matrix sets) is the number of subsystems multiplied by the number of islands (placement matrix) per subsystem. Each subsystem is a matrix set where each matrix is an island and each column is an SIV. Each SIV is randomly generated and meets all five constraints in Eqs. (11)-(15).

**Algorithm 2** Subs-H/V Optimization

1: $P_{m,n}^{'}$ sets =FeasibilityTest($P_{m,n}$ sets ); //verify the feasibility of each $P_{m,n}$(solution matrix)
2: $< P_{m,n}^{'}$, costs$>$ sets = Cost Functions($P_{m,n}^{'}$ sets );//calculate the costs of each $P_{m,n}$
3: sorted $< P_{m,n}^{'}$, rank$>$ set =NonDomSorting($<P_{m,n}^{'}$, costs$>$ sets)
4: Save the sorted $< P_{m,n}^{'}$, rank$>$ set as $ST_k$;
5: Perform within-subsystem imgration;
6: Perform inter-subsystem imgration;
7: Perform probabilistic mutation;
8: Clear the duplicates and get the $P_{m,n}^{''}$ set = RemoveDuplicate($P_{m,n}^{'}$);
9: $P_{m,n}^{'''}$ set = FeasibilityTest($P_{m,n}^{''}$ set);
10: $< PP_{m,n}^{'''}$, costs$>$ set = Cost Functions($P_{m,n}^{'''}$ set);
11: $< P_{m,n}^{'''}$, rank$>$ set = NonDomSorting($< P_{m,n}^{'''}$, costs$>$ set);
12: Replace the worst $N^{elite}$ matrices of current set with the best $N^{elite}$ matrices saved in step four as $< P_{m,n}^{\#}$, costs$>$ set;

**Subsystem Optimization**. It includes within-subsystem migration and cross-subsystem migration. The within-subsystem migration is executed on each subsystem. It is used for each subsystem to perform self-optimization with respect to its own objectives and constraints. During within-subsystem migration, different islands in the same subsystem share information with each other. The cross-subsystem migration is only carried out in selected subsystem pairs. It is designed for the chosen subsystems to exchange information with each other and improve each other's performance, which further optimize the entire complex system.

As shown in Fig. 1, feasibility test and cost functions are carried out on each subsystem, based on the constraints and objectives that we have set. The feasible solutions of each objective have been obtained from different subsystems. The subsystems are loosely coupled and they communicate with each other by cross-subsystem migration. It provides an efficient way to communicate between subsystems and provides a unique migration strategy to share information both within and across subsystems. Cross-subsystem migration can significantly enrich communication among subsystems compared to more traditional methods.

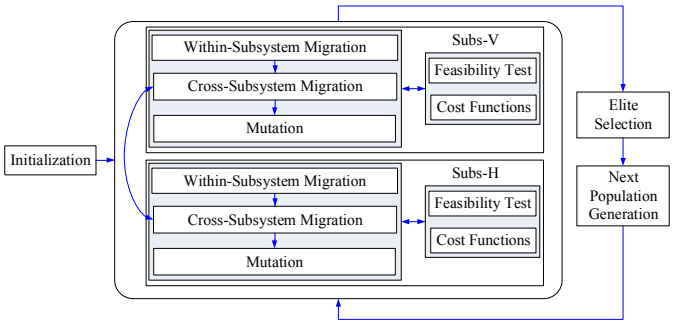



Fig. 1: Subsystems in VMPMBBO

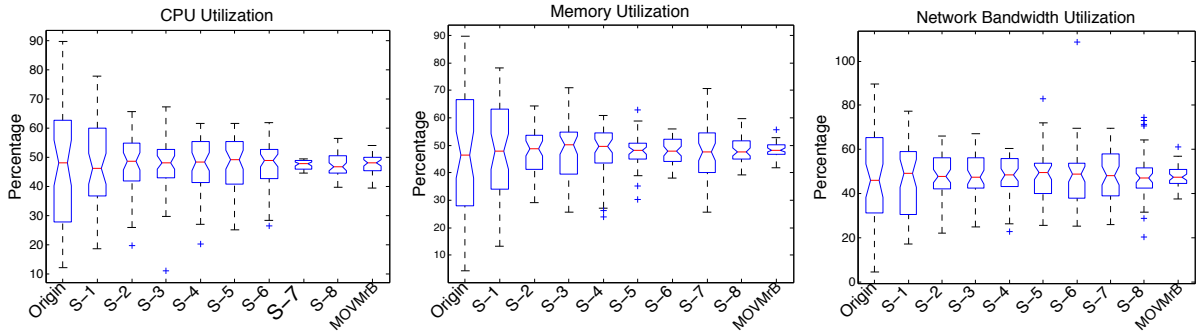
Algorithm 2 describes the optimization of the two types of subsystems *Subs-H* and *Subs-V*. The cost functions for the matrix in *Subs-H* are *unevenness(cpu)*, *unevenness(memory)*, *unevenness(bandwidth)* and $HybridMigCost()$. The cost functions for the matrix in *Subs-V* are $HybridMigCost()$ and $\sum_{j=1}^{M} disequilibrium(j)$. Function FeasibilityTest verifies that each candidate solution (a rebalanced placement matrix) satisfies all constraints in Eqs. (11)-(15). If any constraint is violated, the matrix is replaced with a newly generated matrix based on First Fit Decrease algorithm. If the adjusted matrix still violates some constraint, discard this new matrix and reuse the matrix before the current rebalancing iteration. Function NonDomSorting uses the non-dominated ranking algorithm in [3],though not included, due to space limitations.

During probabilistic mutation, each island in a subsystem is chosen for mutation with probability $P_{mutation}$. The mutation is to replace a randomly selected SIV in the chosen island with a Gaussian mutation operator. Following the mutation, duplicate islands are removed and the feasibility of each island (candidate solution) is checked against the constraints.

**Elitists Selection**. Rank candidate solutions (matrices) using the non-dominated ranking algorithm in [3]. Then choose a set of $N^{elite}$ best non-dominated solutions from the best solutions in the current generation and the elitists from the last generation. Next generate new population for the next generation and replace the worst candidate solutions with the newly selected $N^{elite}$ elitists.

## IV. Evaluation

In this section, we describe the settings of our experiments and present the evaluation results comparing the proposed MOVMrB with eight existing VMrB solutions.




(a) (b) (c)

Fig. 2: Boxplots of inter-HM resource utilization in Scenario-1

TABLE 1: Solutions for comparison

| Solution | Load scalarization Model | Optimization Algorithm |
|---|---|---|
| S-1 | Weighted Model | Best Fit Decrease (BFD) |
| S-2 | Minimax Model | |
| S-3 | Constraint Model | |
| S-4 | Ideal Point Model | |
| S-5 | Weighted Model | Genetic Algorithm (GA) |
| S-6 | Minimax Model | |
| S-7 | Constraint Model | |
| S-8 | Ideal Point Model | |

TABLE 2: VM instances and Server specification in EC2

| Instance Specs | | | Server Specs | |
|---|---|---|---|---|
| Instance Type | vCPU | Memory(GB) | CPU | Memory(GB) |
| T2.micro | 1 | 1 | 40 logic processors | 128 |
| T2.small | 1 | 2 | | |
| T2.medium | 2 | 4 | | |
| M3.medium | 1 | 3.75 | | |
| M3.large | 2 | 7.5 | | |
| M3.xlarge | 4 | 15 | | |
| M3.2xlarge | 8 | 30 | | |

### A. Experimental Setting

*1) Solution Configuration:* The eight competing solutions are combinations of two existing optimization algorithms and four existing multidimensional load balancing models (adopted in [7], [8], [1], [10],respectively), as listed in Table 1. The minimum variance strategy is adopted to quantify inter-HM evenness in the eight solutions. In the first four solutions, the VMrB process is simulated using the existing Best Fit Decrease algorithm (BFD) in [1], [7], [10]. In the other four solutions, the VMrB process is simulated using the existing Genetic Algorithm (GA) [6].

MOVMrB is configured with the parameters: the number of subsystems is 2. The number of islands per subsystem (archipelago), $P_{mutation}$ and $N^{elite}$ are 3 and 0.05 and 1 respectively, which are adopted in our early research [3]. Every test was repeated with 10 runs for each algorithm in each scenario and the average resultsover 10 independent runs are reported. Based on the complexity of the different datasets, the termination criterion is set to 100,000 function evaluations.

MOVMrB and eight competing solutions were implemented and tested using CloudSim 3.03 [23]. The test cases were conducted on a VM with 4 vCPU (2.2GHz each) and 8 GB of memory.

*2) Datasets:* We evaluate the performance of our solution using use both synthetic and real dataset. For the synthetic dataset (called Scenario 1), we generate VM demand sets based on the widely adopted normal distribution $N(0.12, 0.05)$ [24]. The system in the simulation consists of 50 HMs and 200 VMs. A random VM to HM mapping is used in the initial layout. For the real dataset (called Scenario 2), we consider the resource requirements of seven different types of VM instances in Amazon EC2, as shown in Table 2. A total of 3,000 requests from these seven types of VM instances are randomly generated.

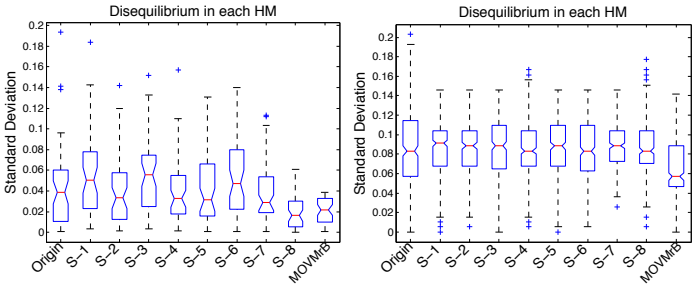
### B. Experimental Results

*1) Solution Quality:* The first scenario is based on Synthetic Dataset. The results are shown in Figs. 2 and 3(a), and Table 3. Fig. 2 depicts the statistical distribution of resource utilization among HMs before and after load rebalancing in 10 simulation runs. origin in the figures refers to the VM to HM placement before the rebalancing is done. The median in boxplot refers to the middle value of resource utilization of each HM. The more concentrated the data, the more balanced the inter-HM resource. It is revealed that the intervals between upper quartile and lower quartile of loads in GA (S5-S8) and MOVMrB are smaller than those in BFD (S1-S4), respectively. In Fig. 2 (a), the CPU utilization across HMs is more balanced in S-7 than in MOVMrB, and MOVMrB is better than the others. However, from Figs. 2 (b) and (c), we can find that both the memory and network loads are most balanced in MOVMrB, and the variation range of inter-HM load is reduced to below 20%.

In Fig. 3(a), we use boxplot to depict the statistical distribution of intra-HM load disequilibrium values through their quartiles in 10 runs. The smaller the value, the more balanced the intra-HM resource. Because the other solutions do not include intra-HM load rebalancing, the multiple loads on each HM become more disequilibrium than the original state except solution S-8. However, the intra-HM load reachs a more balanced state in MOVMrB, where the standard deviation of the intra-HM load is below 0.04.

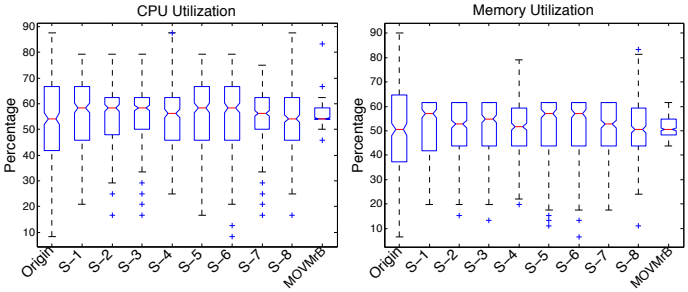
TABLE 3: Live migration cost in Scenario-1

| Solution | S-1 | S-2 | S-3 | S-4 | S-5 | S-6 | S-7 | S-8 | MOVMrB |
|---|---|---|---|---|---|---|---|---|---|
| $HybridMigCost^{lowerBound}$ | 83.6 | 83.6 | 82.4 | 83.4 | 83.7 | 81.8 | 83.4 | 83.1 | 8.4 |
| $HybridMigCost^{upperBound}$ | 496.7 | 496.7 | 487.9 | 495.5 | 496.7 | 485.6 | 495.2 | 493.4 | 51.6 |




(a) Scenario-1 (b) Scenario-2

Fig. 3: Boxplots of intra-HM load disequilibrium




(a) (b)

Fig. 4: Boxplots of inter-HM resource utilization in Scenario-2

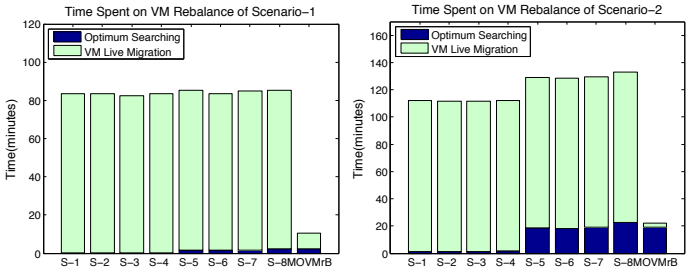
Table 3 lists the average VM live migration costs with nine solutions in 10 runs. This table shows that MOVMrB accelerates the live migration process more than nine times.

Scenario 2 is based on real dataset. The results are shown in Figs. 3(b), 4 and Table 4. Figs 2(b) and 3 plot the average results of 10 runs. The statistical distribution in Fig. 4 shows that the variation of inter-HM resource utilization in MOVMrB is controlled within the range of 15% for CPU utilization and 20% for memory utilization, which is much less than the other eight solutions.The advantage of MOVMrB is not very obvious in intra-HM balance in Scenario 2, where nine solutions can balance intra-HM resources more or less. However, MOVMrB has significantly improved the inter-HM balance when taking intra-HM disequilibrium at a lower level.

From Table 4, it is obvious that MOVMrB accelerates the live migration process about 35 times. And it just takes no more than 15 minutes to migration a total of 2775 VM instances. The speed-up ratio is much more than that in Scenario 1, because there are more access switches and aggregate switches in Scenario 2 than Scenario 1, and there is much more room for migration parallelization.

In summary, MOVMrB can achieve a better inter-HM and intra-HM balance at the smallest migration cost. MOVMrB has an overall superior performance compared to the other eight solutions.

*2) Rebalance Efficiency:* The entire VMrB rebalancing process involves two stages: optimum searching and VM live migration. Fig. 5 shows the average time spent on each stage in nine solutions of 10 runs. The curves illustrate that: (1) The time for optimum searching in BFD solutions is always less than GA solutions and MOVMrB. This is because both GA and MOVMrB are non-deterministic algorithms, which have the ability of finding the global optimum but at a relatively slow convergence speed. (2) The time for VM live migration is much more than the time for optimum searching in all cases except MOVMrB in scenario 2 where there is much more room for migration parallelization. (3) Compared to GA solutions, MOVMrB has about the same computational time, which is about 20 minutes in Scenario 2. (4) The total rebalancing time in MOVMrB is always less than that of the other eight solutions.




(a) (b)

Fig. 5: Total rebalance time of each scenario in nine solutions

## V. Conclusion

In this paper, we proposed a novel multiple objective optimization framework named MOVMrB for rebalancing the placements of VMs in order to achieve load balance of multiple resources in cloud computing. Our approach does not simplify the problem complexity by scalarizing multiple objectives into one. Instead, we treat each resource as a separate dimension and simultaneously maximize the load balance of each resource.

Our proposed VM rebalancing solution considers both the load balancing of each resource across HMs (inter-HM load balancing) and the load balancing of different resources within the same HM (intra-HM load balancing) at the same time. This design makes the entire system reach a good load balance. We also present a hybrid live VM migration strategy that utilizes the information about the migration network architecture and the interval optimization method to speed up the live migration of a set of VMs.

In addition, we systematically reviewed the existing research in load balancing of cloud systems, and classified them using

TABLE 4: Live migration cost in Scenario-2

| Solution | S-1 | S-2 | S-3 | S-4 | S-5 | S-6 | S-7 | S-8 | MOVMrB |
|---|---|---|---|---|---|---|---|---|---|
| $HybridMigCost^{lowerBound}$ | 110.8 | 110.6 | 110.5 | 110.1 | 110.6 | 110.7 | 110.6 | 110.5 | 3.1 |
| $HybridMigCost^{upperBound}$ | 491.4 | 490.4 | 489.9 | 487.9 | 490.4 | 490.9 | 490.4 | 490.1 | 14.3 |

different taxonomies. We conducted an extensive set of experiments to evaluate our approach against existing similar VMrB solutions. The experiments are done using both synthetic data and real world data set. The results demonstrate that our scheme can achieve a more balanced inter-HM and inter-HM load in a more efficient way. To the best of our knowledge, this is the first true multiple objective load rebalancing effort. And this is the first attempt at optimizing the inter-HM load balance and intra-HM load balance simultaneously.

In the future, we will refine our hybrid live VM migration cost model [25] and extend our hybrid live VM migration technique to accommodate other network architectures, such as fat-tree and SDN [26]. The parallelization of BBO/Complex will also be explored and tested on a cluster powered by OpenStack.

## ACKNOWLEDGEMENT

This research was partially supported by the National Natural Science Foundation of China under Grant Nos. 61502379, 61472317, 61428206, 61532015, 91118005 and 91218301, the Ministry of Education Innovation Research Team No. IRT13035, and the National Key Technologies R&D Program of China under Grant No. 2013BAK09B01.

## REFERENCES

[1] C. Tang, M. Steinder, M. Spreitzer, and G. Pacifici, "A scalable application placement controller for enterprise data centers," in *Proceedings of the 16th international conference on World Wide Web*. ACM, 2007, pp. 331–340.

[2]

[3] Q. Zheng, R. Li, X. Li, and J. Wu, "A multi-objective biogeography-based optimization for virtual machine placement," in *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*. IEEE, 2015, pp. 687–696.

[4] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, R. Rastogi *et al.*, "Load balancing of nodes in cloud using ant colony optimization," in *Computer Modelling and Simulation (UKSim), 2012 UKSim 14th International Conference on*. IEEE, 2012, pp. 3–8.

[5] X. Shi, H. Jiang, L. He, H. Jin, C. Wang, B. Yu, and X. Chen, "Developing an optimized application hosting framework in clouds," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1214–1229, 2013.

[6] J. Zhao, Y. Ding, G. Xu, L. Hu, Y. Dong, and X. Fu, "A location selection policy of live virtual machine migration for power saving and load balancing," *The Scientific World Journal*, vol. 2013, 2013.

[7] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and gray-box strategies for virtual machine migration." in *NSDI*, vol. 7, 2007, pp. 17–17.

[8] L. Lu, H. Zhang, E. Smirni, G. Jiang, and K. Yoshihira, "Predictive vm consolidation on multiple resources: Beyond load balancing," in *Quality of Service (IWQoS), 2013 IEEE/ACM 21st International Symposium on*. IEEE, 2013, pp. 1–10.

[9] C. Tian, H. Jiang, A. Iyengar, X. Liu, Z. Wu, J. Chen, W. Liu, and C. Wang, "Improving application placement for cluster-based web applications," *Network and Service Management, IEEE Transactions on*, vol. 8, no. 2, pp. 104–115, 2011.

[10] Z. Xiao, W. Song, and Q. Chen, "Dynamic resource allocation using virtual machines for cloud computing environment," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 24, no. 6, pp. 1107–1117, 2013.

[11] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach," *evolutionary computation, IEEE transactions on*, vol. 3, no. 4, pp. 257–271, 1999.

[12] J. Xu and J. A. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," in *Green Computing and Communications (GreenCom), 2010 IEEE/ACM Int'l Conference on & Int'l Conference on Cyber, Physical and Social Computing (CPSCom)*. IEEE, 2010, pp. 179–188.

[13] Y. Gao, H. Guan, Z. Qi, Y. Hou, and L. Liu, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences*, vol. 79, no. 8, pp. 1230–1242, 2013.

[14] M. Mishra and A. Sahoo, "On theory of vm placement: Anomalies in existing methodologies and their mitigation using a novel vector based approach," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 275–282.

[15] K. Ye, X. Jiang, R. Ma, and F. Yan, "Vc-migration live migration of virtual clusters in the cloud," in *Grid Computing (GRID), 2012 ACM/IEEE 13th International Conference on*. IEEE, 2012, pp. 209–218.

[16] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. USENIX Association, 2005, pp. 273–286.

[17] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the performance of virtual machine migration," in *Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on*. IEEE, 2010, pp. 37–46.

[18] W. Cerroni, "Multiple virtual machine live migration in federated cloud systems," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*. IEEE, 2014, pp. 25–30.

[19] K. Ye, X. Jiang, D. Huang, J. Chen, and B. Wang, "Live migration of multiple virtual machines with resource reservation in cloud computing environments," in *Cloud Computing (CLOUD), 2011 IEEE International Conference on*. IEEE, 2011, pp. 267–274.

[20] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *ACM SIGCOMM Computer Communication Review*, vol. 38, no. 4, pp. 63–74, 2008.

[21] D. Du and D. Simon, "Complex system optimization using biogeography-based optimization," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[22] D. Simon, R. Rarick, M. Ergezer, and D. Du, "Analytical and numerical comparisons of biogeography-based optimization and genetic algorithms," *Information Sciences*, vol. 181, no. 7, pp. 1224–1248, 2011.

[23] R. N. Calheiros, R. Ranjan, A. Beloglazov, C. A. De Rose, and R. Buyya, "Cloudsim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms," *Software: Practice and Experience*, vol. 41, no. 1, pp. 23–50, 2011.

[24] O. Biran, A. Corradi, M. Fanelli, L. Foschini, A. Nus, D. Raz, and E. Silvera, "A stable network-aware vm placement for cloud systems," in *Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE Computer Society, 2012, pp. 498–506.

[25] S. Nathan, U. Bellur, and P. Kulkarni, "Towards a comprehensive performance model of virtual machine live migration," in *Proceedings of the Sixth ACM Symposium on Cloud Computing*. ACM, 2015, pp. 288–301.

[26] H. Wang, Y. Li, Y. Zhang, and D. Jin, "Virtual machine migration planning in software-defined networks," in *Computer Communications (INFOCOM), 2015 IEEE Conference on*. IEEE, 2015, pp. 487–495.