Live Migration-based Resource Managers for Virtualized Environments:

A Survey

Omar Abdul-Rahman
Graduate School of Information Science & Technology
Hokkaido University
Sapporo, Japan
omar@ist.hokudai.ac.jp

Masaharu Munetomo and Kiyoshi Akama
Information Initiative Center
Hokkaido University
Sapporo, Japan
munetomo@iic.hokudai.ac.jp, akama@iic.hokudai.ac.jp

Abstract— **Virtualization is a technology originally developed for mainframe computing. However, recent developments in the virtualization makes it a key technology to address the problems of modern distributed infrastructure like cloud platforms. Perhaps, one of the most important mechanisms provided by virtualization is the ability to migrate running applications without affecting the end user in a seamless manner. So, virtual machine migration is a promising approach to realize the objectives of efficient, adaptive and dynamic resource manager for virtualized environments. In this the paper, we present the state of art migration based resource managers for virtualized environments, compare and discuss different types of the underlying management algorithms from algorithmic issues standpoint.**

Keywords- virtualization; live migration; management algorithms; consolidation; orchestration

I. INTRODUCTION

Virtualization has attracted considerable interest in recent years, particularly from the datacenters, could platforms and cluster computing communities. By defining an intermediate layer that decouples the operating systems (that is in direct control of the hardware) and the applications, virtualization can address the problems of modern distributed infrastructures like load balancing, high availability, rapid infrastructure deployment and application isolation. Perhaps, the biggest advantage of employing virtualization is the ability to flexibly control resource allocations. The dynamic resource allocation requirements of workload can be satisfied by altering the capacity of a virtual machine at runtime. While, the ability to power-on/off, archive, migrate containers and their workloads enhance the capability of the resource manager to work around resource bottlenecks and faults. [1]

Despite virtualization capabilities, a still challenging question is *how an intelligent infrastructure should optimally maps workload and resource requests onto available virtualized resources utilizing theses capabilities?* It is possible to identify four different trends to realize dynamic resource management systems in virtualized environments. A large part of the literature is based on *request distribution policies* [2][3][4][5][6]. In this trend, a controller adopts a policy that dynamically adjusts requests distribution to share resource among the running applications. By using *virtual machine slicing* [7][8], a controller manages resources by dynamically change virtual machines allocations (or fractions of usage).Then, we have resource management using *virtual machine replication/instantiation* technique [9]. Replication/instantiation entails the creation of a local virtual machine's replica (or instantiate a new virtual machine) in the target physical server. The load would be shared between the two instances, diminishing the stress on the local physical server. Finally, we have resource management by *virtual machine migration*.

The successive developments in the field of virtualization technology greatly reduced downtime overhead associated with migration. For example, support for migrating groups of processes across OSs was presented in [10], but applications had to be suspended and it did not address the problem of maintaining open network connections. In [11] Virtualization support for commodity operating systems led towards techniques for virtual machine migration over long time spans, suitable for WAN migration [12]. More recently, the most two popular modern virtualization technologies products from VMware [13] and Xen [14] have realized the notion of *live* or *seamless* migration of VMs that involve extremely short downtimes ranging from tens of milliseconds to a second. Thus, virtual machine migration is emerged as a promising technique to be utilized by resource management algorithms to rapidly resolve resource allocation problems in the virtualized environments. However, up to date this approach has received a little attention. Thus, we limit the scope of this survey to the dynamic resource manager based on the live migration technique. The remaining part of this paper is organized as the follows. A general live migration based resource manager system is presented in Section II. Different types of underlying management algorithms is presented and discussed in Section III. We conclude the paper and highlight possible directions of future research in Section IV.

II. GENERAL LIVE MIGRATION BASED RESOURCE MANAGER

In this section, architecture is presented that highlights the main phases of processing for a general live migration

 ISBN: 978-1-61208-106-9

based resource manager. There are different levels of virtualization; however, we are care here for operating system level virtualization, which supports the scenario shown in Figure 1; on each physical machine (PM) there is a Virtual Machine Monitor (VMM), also called hypervisor that allows for many virtual machines (VMs) to share the physical resources. A dynamic resource manager imposes a fair resource sharing policy on the competing VMs by producing suitable migration orders, which are implemented by VMM. The widely testified assumption is that each application is represented by a single VM running in a shared hosting model. Exceptions are [15], where each application can be represented by one or more VMs, [16][17] that support the notion of virtual clusters of VMs, while, [18] is designed for multi-tier distributed infrastructure. The resource manger is usually processed in an iterative manner. In each iteration, there are three main phases of processing that can be described as follows:

1. Pre-allocation Phase; the duty of the resource manager here is to collect usage data from the running nodes within a specific measurement interval employing a specific monitoring engine. The details of this engine depend largely on the employed virtualization technology and the required data to be collected. Through these data, the manger can keep a general view about the performance level in the running nodes. It triggers re-allocation if the there is violation(s) for the predefined triggering conditions.
2. Migration Planning phase; this is the most critical part of the processing, since it is the duty of the resource manager to produce a suitable migration plan or orders for a new placement that can eliminate or minimally violate the triggering conditions. The migration plan usually consists of sender PMs, migrated VMs and receiver PMs.
3. Migration execution phase; it is the duty of the VMM to implement the migration plan or orders produce by the resource manager. The specifics of this phase depend on the employed virtualization technology.

The above mentioned architecture is widely used in the literature. It can be described as a *single-tier* architecture, which differs from multi-tier architecture used in [19] or arbitrator architecture used in [20][21].

III. TYPES OF MANAGEMENT ALGORITHMS

In this section, the underlying algorithms that are used to the implement the surveyed live migration based resource manager are discussed. In order to classify the different algorithms properly, *design model* was adopted as classification criteria from algorithmic issues stand point. By design model, we refer to the standard procedure followed by an algorithm to solve a given problem. Taking design model into consideration, it is possible to broadly classify the surveyed algorithms into *ordering algorithms*, *Constraint Programming* (CP) based algorithms and *Genetic Algorithms* (GAs). Moreover, we discuss the differences in *key concepts or techniques* that exist among the same class

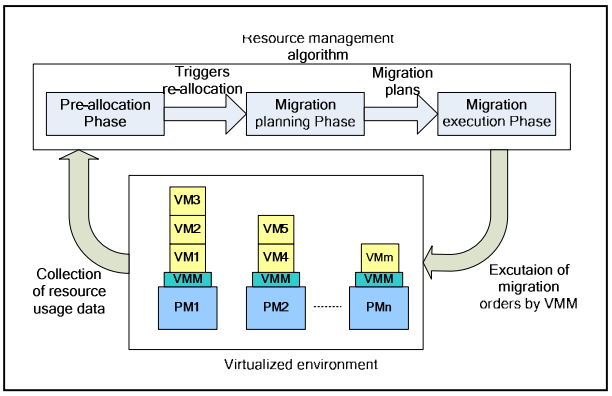

Figure 1. Architecture of a general live migration based resource manager.

of algorithms. These key techniques can be described as an *ad hoc* techniques or modifications that are introduced into a specific algorithm to enhance its performance against the weakness that may exist into the standard procedure to solve a given problem. They give a specific algorithm the unique features and merits against other algorithms. On the other hand, the existed differences among the surveyed algorithms in key techniques do not affect the above mentioned classification into three groups, as long as they follow the same underlying design model. Therefore, the discussion in this section is centered on design model, which give us a brief picture about the processing method, and algorithms features, which give us an idea about the design goals that a specific resource manager is able to satisfy. In addition, experimental results are presented whenever possible. Both experimental results and algorithm features gives an insight about suitability and the ability of a specific resource manager in solving the problems of resource allocation in virtualized environments.

A. *Ordering Algorithms*

VMs resource allocation problem is NP hard problem that is usually formulated as a generalization of knapsack or generalization of its special form of bin packing problem. Therefore, the literature is dominated by ordering algorithms, which is a popular heuristic approach to address such class of problems. The resource manager under such kind of algorithms should resort for a specific ordering model to answer the questions of; From where to migrate? Which VM to migrate? Where to migrate? These questions are the core of the migration plan discussed in Section II. The assignment of a candidate VM to a candidate PM is done only when assignment conditions are met, while, VMM receives migration orders when stopping criteria are met.

The comparison of these algorithms is summarized in Table I, while, these algorithms can be discussed as the follows.

Verma et al., in [22], proposed Dynamic Management Algorithm (DMA) that it is based on Polynomial-Time Approximation Scheme (PTAS) heuristic algorithm. The algorithm operates by maintaining two types of ordering lists, which are migration cost list (VMs on each PM is arranged in non-decreasing order according to their migration cost)

 ISBN: 978-1-61208-106-9

TABLE I. COMPARISION OF ORDERING ALGORITHMS

Resource manager	Design model				
	From where to migrate	*Which VM to migrate*	*where to migrate*	*Assignment conditions*	*Stopping criteria*
DMA [22]	Max-Min thresholds violating PM	Lowest utilization VM (migration cost)	Lowest PM (unused portion of resources)	Resource constraints satisfaction	Repeat migrations until Overload or underutilization elimination
TOPSIS Algorithm [23]	overloaded PM (threshold breaking volume value)	Best VM candidate (migration cost)	Lowest PM (volume)	Resource constraints satisfaction	Repeat migrations until Overload elimination
Andreolini et al., algorithm [24]	Overloaded PMs (CUSUM algorithm)	Highest loaded VM (Sorting algorithm)	lowest loaded PM (Sorting algorithm)	Each receiver must accept one guest VM in a greedy manner	When all guest VMs are assigned to the receivers
MFR algorithm [25]		All VMs are sorted in descending order according to resource demand forecast	Bin packing heuristic is used to map VMs onto the PMs	Resource constraints satisfaction	Construct a new placement that satisfied assignment condition or minimally violated it
pMapper [20] (power manager)	Overloaded PMs (SLA violations)	Lowest VM (application size)	Bin packing heuristic is used to map VMs onto the PMs	Resource constraints satisfaction	Construct a new placement that satisfied SLA requirements
Sandpiper [26]	Overloaded PM (threshold breaking volume value)	Highest VM (VSR)	Lowest PM (volume)	Resource constraints satisfaction	Repeat Until overloading is eliminated by migration or swap
Ruth et al., algorithm [16]		Heuristic is used to determine over and under utilization VMs	Lowest loaded PM in the same domain or other domain	Resource constraints satisfaction in same domain or other domains	Repeat migrations until overloading and underutilization is eliminated
vGreen [19] (MPC balance)	Overloaded PM (threshold breaking nMPC value)	Lowest VM (vMPC)	lower than overloaded PM (nMPC)	Resource constraints satisfaction and system balance	Repeat migrations until MPC violations are eliminated
vGreen [19] (IPC balance)	Overloaded PM (threshold breaking nIPC value)	Lowest VM (vIPC)	A node that has a lower nMPC value than the sender one	Resource constraints satisfaction and system balance	Repeat migrations until MPC violations are eliminated

and residual capacity list (all PMs are arranged in non-decreasing order according to their residual capacity, which reflects the unused portion of resources). One of the features of this management algorithm is the ability to minimize the migration cost. This is achieved by employing a migration cost function that ranks possible new placements. High rankings will be given for those placements as close as from the current one, while, low rankings will be given for far placements or placements that initiates idle PMs. Another feature is the ability to minimize power consumption by detecting underutilization in the managed system. This achieved by using Max-Min thresholds selection model. When the resource usage of a running PM violates a minimum predefined threshold value, the algorithm tries to pack the running VMs as close as possible thus trying to minimize the number of running PMs.

Comparison experiments were held between the proposed DMA and an exact or optimal DMA. The optimal DMA outperforms the exact one only in the in the optimality of solutions, i.e., mapping VMs onto fewer PMs. However, the proposed DMA has a better time performance and minimum migration cost. In addition, the proposed DMA has the ability to adapt the running resources toward workload variability, while, optimal DMA lacks this adaptability merit. The main limitation of this algorithm is the CPU-memory resource model. Under this model it is not possible to judge the performance of the resource manager under network intensive applications.

Tarighi el al in [23] formulate resource allocation problem as a multi-criteria decision problem and employed fuzzy Decision Making Model (FDM) based on TOPSIS techniques. Fuzzy TOPSIS is a technique for ordering preference by similarity to an ideal solution. Adopting this technique gives the authors the flexibility to include beside the usual deterministic information another kind of information in the form of linguistic and fuzzy parameters. For example, like Sandpiper in [26] the authors used *volume* or ordering function to detect the degree of overloading in the running PMs. However, unlike the volume in Sandpiper, which usually processes deterministic data of resource usage (CPU, memory and network), volume here is able to process for example Quality of Service (QoS) as a linguistic parameter and temperature of the PM as a fuzzy parameter. By including the temperature, the manager becomes able to forecast failure for a specific node and takes fault tolerance migration actions. This an advantage compared with other resource managers. Another point of comparison with Sandpiper [26] is VMs ordering functions. Both of them using the same kind of function, however, the function here is included further data of memory footprint size to arrange VMs. That adds flexibility to the performance of the resource

 ISBN: 978-1-61208-106-9

manager by giving priority to specified VMs during migration.

Experimental investigation validates the performance of the system under different critical scenarios. However, the results showed that method suffers from the problem of time-consuming calculations.

Andreolini et al., in [24], proposed management algorithms for cloud computing context. One of interesting part of this algorithm is that of using a selective CUSUM algorithm to detect the critical nodes instead than the widely used threshold violation selection model. An important feature of this algorithm is the ability to capturing significant changes of running PMs load. Thus, limiting the number of sender nodes and avoid needless migrations generated by false alarms due to many instantaneous spikes, non-stationary effects, and unpredictable and rapidly changing load. Another part is using a load trend-based sorting algorithm to select the candidate VMs for migrations and the receiver nodes. An important feature of this algorithm is the ability to clearly classify the trend of the running VMs load to increasing, decreasing, oscillating or stabilizing, Thus, contributes to limiting number of migrated VMs to only few critical ones. Finally, the algorithm distribute the migrated VMs by assigning each receiver only one VM. Thus, we can avoid load imbalance shifting, which is undesirable effect that generates frequent fluctuations negatively impact system performance and stability. The authors confirmed the performance of their system by experiments on traces coming from cloud platforms, however, no experimental details are provided in this paper.

Bobroff et al., in [25], proposed a dynamic server migration and consolidation algorithm introduced as Management-Forecast-Reallocation or MRF algorithm. The interesting part of this algorithm is maintaining a gain formula that it is used to combine online resource measurements with resource demand forecast. This gain formula is the core of VMs ordering list. Using of forecasting technique gives MFR the ability to adapt the available resource to the workload variability in a proactive manner, thus providing probabilistic SLA guarantees. In addition, it can minimize the number of running PMs at the time of decreasing workload behavior thus minimize power consumption. However, the downside of forecasting technique is the increased sensitivity of MFR performance toward remapping interval. For longer remapping intervals, the performance of MFR is degraded. This is showed by the experimental results. Another feature of this algorithm is the ability to deal with the critical scenario of high workload utilizations. When it is impossible to totally eliminate capacity violations, it generates a placement that minimizes violations as much as possible.

The experimental results were based on workload traces across a variety of operating systems, applications, and industries. The results were proved that MFR outperforms static allocation in term of reducing the physical resource consumption for a specific SLA violations rate by 50% and reducing SLA violations at a fixed capacity by 20%. On the other side, the main limitations is in the resource model (CPU only), thus we could not get the full picture of the resource manager performance under memory and network intensive applications.

Sandpiper is a resource manager proposed in [26]. An interesting feature of Sandpiper is that of using a score function or ***volume*** to measure the degree of overloading in the running PMs. This function is designed to capture overloading along three dimensions of CPU, memory and network. In addition, another score function or Volume Size Ratio (VSR) is designed to order VMs in a descending order according to their memory size, thus minimize the migration overhead. Moreover, Sandpiper, like [25], is designed to deal with critical scenario of high workload utilization but using a different approach. Sandpiper first tries to mitigate a hotspot by migration. If failed, it tries to swap a high VSR VM in the overloaded node with one or more of low VSR VMs in the destination node. The experimental results showed that migration overhead is less than that of swapping overhead; however, swapping increases the chances of mitigating hotspots in clusters with high average utilization. Another feature is the ability of Sandpiper to address system stability by avoiding needless, wasteful and thrashing migrations. Sandpiper avoids needless migrations generated by false alarms by triggering migrations only if thresholds or SLAs are exceeded for sustained time. In case of increasing number of hotspots, Sandpiper either implements a partial solution or gives up entirely wasteful migrations. However, monitoring techniques is the most interesting part of the paper. Here, the authors proposed black box and gray box monitoring techniques. In the gray box technique, it is possible for Sandpiper to relay on some OS level statistics beside external usage to infer SLA violations. However, in the black box technique, Sandpiper depends only on the external usage to infer the SLA violations. This ability to use some OS level statistics gives gray box based Sandpiper an edge performance over black box one. Experimental results showed that gray box based Sandpiper behaves proactively. So, it produces fewer swaps, resolve situations faster and balance system more quickly compared with black box Sandpiper. By comparing with static allocation, Sandpiper eliminates all hotspots, while static allocation failed. In addition, Sandpiper reduces the number of intervals experiencing sustained overload by 61%. the experiments showed that the system overhead has insignificant CPU and I/O requirements and has a negligible impact on performance, while, the system can scaled up to 500 VMs with computational complexity of less than 5 seconds. For very large data centers with thousands of VMs, authors proposed that computation could be split up across multiple nodes, or the center's servers can be broken up into pools, each controlled independently by its own control plane. On the other hand, results showed that the quality of Sandpiper degrades for long measurements interval.

Ruth et al., in [16], presented a resource manager for a system called Violin, which composed of virtual network of VMs. An important feature of this manager is the ability to identify and eliminate underutilization by employing max-min threshold violations detecting heuristic. Thus, reduce power consumption. However, the novel part of this paper is in the employing two different virtualization techniques of

 ISBN: 978-1-61208-106-9

VM slicing and VM migration to resolve resource allocation problem. By VM slicing, the manager first performs fine-grained control over per-host memory and CPU allocation utilizing memory ballooning and CPU scheduling techniques provided by VMware and Xen technologies. If failed to resolve resource allocation problem, it adopts the VM migration option. This hybrid approach is an efficient method for limiting the number of migrations, thus minimize migration cost.

The experimental study reported a small migration overhead and concentrated mainly on validating the performance of the system under different critical scenarios. However, the main limitation is in the consideration of only two dimensions of CPU and memory as a sources model.

Dhiman et al., in [19], presented vGreen, a multi-tiered software system, for energy efficient computing in virtualized environments. The innovative part of this system is in the using of novel hierarchal parameters processed in a novel multi-tier architecture. The authors developed the idea of the hierarchal parameters from experiments on benchmarks from SPEC-CPU 2000 suite, namely *mcf* and *perl*. Experiential results showed that co-scheduling of VMs with heterogeneous characteristics on the same PM is beneficial from both energy efficiency and performance point of view. In order to capture these characteristics, authors developed two kinds of parameters which are memory accesses per cycle (MPC) and instructions per cycle (IPC). These parameters are maintained in two *hierarchal* levels of node (nMPC, nIPC) and VM (vMPC,vIPC). On the other hand, the multi-tier architecture is shown in Fig. 2. In contrast to the single tier architecture that it is shown in Fig. 1, where the migration planning phase is implemented in a single-tier or step, the migration phase in multi-tier architecture is implemented in four sequential steps. Both of the above mentioned architecture and parameters gives vGreen the merit of addressing performance and power balancing requirements more accurately compared with other systems presented in this survey. Therefore, when the manager implements MPC balance tier (Table I), this results in a better overall performance and energy efficiency across the cluster. While balancing of IPC metric values in the IPC balance tier (Table I) results in better balance of power consumption across the PMs. The performance goals are further checked in the Uitl balance tier which eliminates overcommitted nodes from the cluster. Finally, VM consolidation tier contributes in the resource adaptation ability of the manager by eliminating underutilization from the cluster. By comparing vGreen with VM scheduler that mimics the Eucalyptus, which is a state of art strategy for cloud context, and under heterogamous workload conditions, the experimental results showed that vGreen outperforms along author`s developed metrics of energy savings, which captures energy consumption reduction, Weighted Speedup, which captures migration overhead and Reduction in Power Imbalance, which captures the power consumption variance within the machines of the cluster. An overall performance and system level energy savings by 20% and 15% improvement were achieved. However, under homogenous workload condition both performed the same. On other hand,

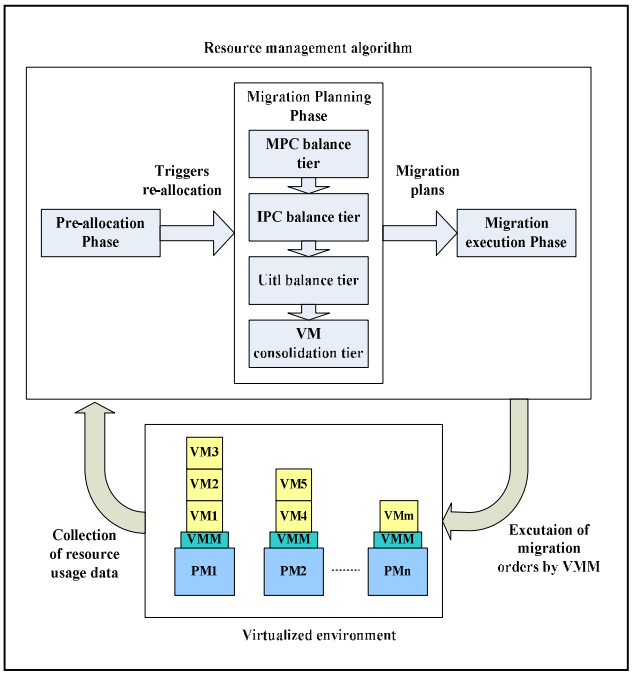

Figure 2. Multi-tier architecture of vGreen system.

the limitation of this manager is in the two dimensions CPU-memory resource model.

In [20], Verma et al. propose the power aware application placement framework or pMapper as the resource manager. The novelty in this resource manager is the architecture which consists of arbitrator that issues migration orders based on the information communicated by performance, power and migration managers. This novel architecture gives pMapper the ability to serve many of management objectives at the same time. The system operation can be described as follows. The performance manager continuously checks the performance level in the running nodes against the performance targets specified by SLA. The Power manager utilized an experimental developed CPU based power model in the generation of power-minimizing new placements (as shown in Table I). While, the migration manager utilized an experimental developed migration cost model that quantify migration cost from the decrease in throughput because of live migration and estimate the revenue loss because of the decreased performance (as given by SLA). Finally, the arbitrator constructs a new placement by picking up the optimal migrations that trade-offs power-migration targets and achieve SLA goals.

In [21] the authors modified their pMapper. The core of their modifications is proposing a new power models that related power consumption to the CPU, memory footprint and caches usage characteristics of the application. The essence of their new proposal is to sort all applications in ascending order by their memory usage. Then, classify them to three categories. Category 1 or small applications can be packed together respecting memory and CPU limits. Thus, avoid performance degrade. Category 2 or large applications can be packed only based on the CPU limit. Thus, achieve

 ISBN: 978-1-61208-106-9

maximum power savings. Finally, category 3 or medium applications can be packed either as category 1 or category 2 applications.

The experiments were performed to compare the new CPU-Cache based pMapper with the old CPU based pMapper. Both algorithms succeed in reducing the number of the running nodes from 11 to 4. However, CPU-Cache based pMapper outperforms in term of performance overhead. Then CPU-Cache based pMapper is compared with cache-oblivious strategies. For CPU-Cache based pMapper only Category 3 applications have performance impact, while for cache-oblivious strategies more than half of the applications have performance impact. However, this CPU-memory resource model is still limited since it ignores the effect of network dimension of the resources.

B. Constraint Programming

The main idea of the constraint programming based resource manger is to formulate the VM resource allocation problem as a constraint stratification problem then applies a constraint solver to solve the optimization problem. The ability of this solver to find the global optimum is the main motivation to adopt this approach. In the literature, we have identified two papers. The comparison is presented in Table II, while, the discussion can be presented as follows.

Entropy resource manager [27] utilizes Choco constraint solver to achieve the objectives of minimizing the number of the running nodes and minimizing migration cost. The operation of the algorithm can be described as follows. Entropy iteratively checks optimality constrain, i.e., the current placement uses minimum number of the running nodes. If Entropy at VM packing problem (VMPP) phase success in constructing a new optimal placement (uses fewer running nodes), it will activate the re-allocation. In addition, Entropy employs a novel experimental developed migration cost model that relates memory and CPU usage with migration context. High parallelism migration steps reduce the cost, while sequential and infeasible migration steps increases cost. Using of constraint programming techniques facilitate the task of capturing such context. However, considering only viable processing nodes (a running node can support only a single active VM, while other co-allocated VMs should be inactive) and CPU-Memory resources model are the limitations of Entropy model.

In order to speed up the computation process that can be expensive, authors used many optimization techniques. Some of the techniques used in the VMPP phase are used to detect and exclude partially constructed solutions as soon as possible if they violate the optimality and viability constraints. Others used to reduce search space, by limiting the search for the promising region near from the optimal value by imposing upper and lower bounds. In addition, the authors devise a metric of the number of active VMs divided by the number of nodes to calculate the lower bound, while the upper bound is identified by using First Fit Decreasing (FFD) heuristic. Finally, authors devised equivalence classes metric (VMs memory size to the CPU states), which is exploited to reduce the size of search tree. Moreover,

TABLE II. COMPARISION OF CONSTRAINT PROGRAMMING ALGORITHMS

Resource manager	Design model	Constraint Solver phases of processing	
Entropy [27]	Initialization	Optimality and viability constraints violations.	
	Processing	VM packing problem	
		VM replacement problem	
	Stopping criteria	Can be aborted at any time	
Van et al., algorithms [15]	Initialization	GDM receives inputs from LDM and monitoring probe	
	Processing	GDM	VM Provisioning
			VM Packing and replacement
	Stopping criteria	Can be aborted at any time	

Equivalence classes are also exploited as an optimization technique in VM replacement problem phase with more strict constraints.

Scalability experimental results showed that the system complexity is directly related to the characteristics of running VMs and the underlying PMS. More computation time is required for configurations sets that have many VMs memory requirements and many CPU states. Moreover, the scalability of the system is showed to be related to number of VMs per node. Higher the ratio, longer the time required to compute a solution. On the other hand, when compared with First Fit Decreasing heuristic (FFD), Entropy outperforms in term of minimizing the number of unsatisfied VMs and producing reconfiguration plan with better cost. In addition, Entropy outperforms static allocation by 50% and FFD by 25% in term of minimizing the number of running nodes over a collection of NASGrid benchmarks.

Van et al., in [15], proposed an architecture and management algorithms for cloud computing contexts. The management algorithms are based on Entropy resource manager [27]. However, the main advantage over Entropy resource manager is in the novel architecture that separate the management decisions among a Local Decision Module (LDM) associated with each application and a Global decision Module (GDM). The ability of the resource manager here to combine and automated application provisioning problem with resource adaptation problem is the outcome of this architecture. Another advantage is in the including of operations costs in the migration planning model. However, like Entropy the resource model is limited to the CPU-memory dimensions. The architecture and the algorithms are validated through simulation experiments. The attempt is still in the early development phase.

C. Genetic Algorithms

GAs are metaheurstics that are inspired by evolutionary biology. It starts the evolution process from a population of initial solutions and changes them very fast by applying the usual selection and recombination operators. The rate of change reduces gradually when we reach the optimal solution. This incremental behavior, besides the simplicity of implementation and the ability of parallelization makes GAs a promising approach for VMs resource allocation

 ISBN: 978-1-61208-106-9

problem. In the literature, we have identified two papers. Table III presents and compare the design model for these algorithms, which can be discussed as follows.

Campegiani in [18] used GA to find the optimal allocation of virtual machines in a multi-tier distributed environment. The innovative part of this work is the formal model that addresses beside the usual quantitative resources (CPU, memory and network), qualitative resource like physical position awareness. In addition, this model allows for multiple-SLA representation for each VM. This scheme gives the proposed GA the ability to capture multi-tier distributed infrastructure, by a signing a profit for specific SLA. Maximize the profit through evolution is translated into maximization of physical resources efficiency, while accommodating for transient workload surges. However, the challenge for GA comes from the infeasible solutions that can be appeared as a byproduct of the evolution process. The author addressed this challenge by devised penalty function that differentiates feasible solutions, while penalize unfeasible ones. These infeasible solutions are fixed using a repair operator. Simulation Experiments on arbitrary dataset were held to validate the algorithm which is still in the early development phase.

Nakada et al., in [17], implemented a prototype Virtual Machine packing optimization mechanism on Grivon, which is a virtual cluster management system for Hardware as a Service (HaaS) cloud system type. The most interesting part of this algorithm is the penalty function with the objective of packing VMs tightly onto the PMs to minimize the number of the running nodes, while attempting to minimize migration cost and respecting SLA performance levels at same time. Experiments were held to validate the performance of the system. According to the authors, experimental results showed that the GA based approach is flexible and fast enough for VM packing problems and represent a promising approach.

TABLE III. CPMPARISIOON OF GENETIC ALGORITHMS

The design model of GA	the resource manager	
	Campegiani [18]	Nakada et al [17]
Chromosome representation	Binary representation	Integer sequence
Initial population	Applying First fit, next fit and best fit heuristic on the input configuration.	Repeatedly applying mutation on the input configuration.
Selection operator	tournament selection scheme	regular normalized weighted roulette method
Crossover operator	Uniform crossover	One point cross over
Mutation operator	Fix rate mutation operator	Mutations will cause real change in the individual by reordering two randomly chosen numbers.
Type of fitness function	Penalty function	Penalty function
Replacement scheme	Inject new individual, remove lowest fitness one.	
Special operators	Repair operator	

IV. CONCLUSION AND FUTURE RESEARCH

Virtualization is a re-emerged technology that offers powerful resource management mechanisms that can cope with the challenges of modern distributed infrastructures like datacenters and cloud platforms. Live migration based resource management systems is a promising approach for efficiently manage resources and rapidly eliminate hostpots in the virtualized environments. In this paper, we surveyed state of art resource managers describing them using general resource manager architecture and presenting different types of the resource management algorithms which classified to ordering, constrain programming and genetic algorithms. We compared and discussed these algorithms from the design model and key concepts and techniques standpoints.

It is possible to highlight the below mentioned fully unexploited recent trends; we believe that they will attract a greater attention in the future directions of research by developing more formal models, trying alternative approaches, devising metrics or performing feasibility experiments.

- Qualitative resources are an important recent trend, which gives a resource manager the merit to handle qualitative requirements of resource allocation. For example, physical position awareness, which is described as a qualitative resource, is an important requirement for multi-site virtual clusters. In the survey, we have identified only one paper [18], which is still in the early development phase, which addresses such theme.
- Developing a resource manager that has the ability to combine on-the-fly (or dynamic) application provisioning with dynamic application consolidation at the same time is an important recent trend especially for cloud contexts. We have identified in the survey only one paper [15], which still in the early development stage, that address such theme.
- Developing a hybrid resources manager that uses more than one virtualization techniques (like VM migration and VM slicing) is an important trend that can combine the benefit of both techniques. VM slicing can contributes into migration cost limitation or minimization. In addition, it is useful to capture the structure of modern multi-domain or multi-tier infrastructures. On the other hand, VM migration, that finds global solutions for resource allocation problems, can enhance the performance of VM slicing technique, which is useful in finding local solutions for resource allocation problem. We have identified in the survey only one paper [16] that apply this method for a grid computing platform.
- According to the survey, ordering and CP approaches are suffers from time-consuming calculations. It requires innovative techniques and complex architectures to overcome this difficulty. We believe that the future research developments will turn toward more robust and faster metaheuristic algorithms. Multi-objective GA is a possible

 ISBN: 978-1-61208-106-9

promising candidate algorithm for future cloud systems.

- Hybriding CP and GA approaches can be seen as a possible direction of future research. The combined approach can benefit from their different aspects. From one side, constraint programming is a good approach to model complex constraints. On the other side, GA can greatly fasten CP expensive processing.
- Dataset is important for algorithm development. It is usually used to test algorithms against it. It have been noted from the surveyed papers that many researchers resort to the randomized configurations for developing their algorithms. This field of research lacks suitable dataset that captures virtualized infrastructure. Therefore, future development in this field will be largely affected by the development of suitable dataset by specialized community like operational research or integer programming.
- Finally, future development in this field is largely related to simulation and experimentation procedures. It has been noted from the surveyed papers, that there is a need to define standard benchmarks applications, standard metrics to measure the goodness of the resource managers and to perform experimental tests under large configurations of VMs and PMs and comparing among different states of art resource managers.

ACKNOWLEDGMENT

First author is supported by the Japanese Government (Monbukagakusho) Scholarship program to complete his PHD study. In addition, I would like to thank the anonymous for their efforts, which contributes much to the development of this paper.

REFERENCES

[1] C. Clark, K. Fraser, A. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," In Proc. of the 2nd conference Symposium on Networked Systems Design and Implementation, vol. 2, pp. 273 – 286, 2005.

[2] K. Appleby, S. Fakhouri, L. Fong, M. Goldszmidt, S. Krishnakumar, D. Pazel, J. Pershing, and B. Rochwerger, "Oceano - SLA-Based Management of a Computing Utility," In Proc. of IFIP/IEEE Symposium on Integrated network management, Seattle, WA , USA, pp. 855 – 868, May 2001.

[3] J. Chase, D. Anderson, P. Thakar, A. Vahdat, and R. Doyle, "Managing Energy and Server Resources in Hosting Centers," In ACM SIGOPS Operating Systems Review, vol. 35, issue 5, pp. 103 – 116, December 2001.

[4] B. Urgaonkar, P. Shenoy, A. Chandra, and P. Goyal, "Dynamic Provisioning for Multi-Tier Internet Applications," In Proc. of the 2nd IEEE International Conference on Autonomic Computing, Seattle, WA , June 2005.

[5] K. Rajamani and C. Lefurgy, "On Evaluating Request-Distribution Schemes for Saving Energy in Server Clusters," Proc. IEEE International Symp. Performance Analysis of Systems and Software, pp. 111 – 122, 2003.

[6] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi, "Dynamic Placement for Clustered Web Applications," In Proc. of the 15th international conference onWorld WideWeb, Edinburgh, Scotland, pp. 595 - 604, 2006.

[7] J. Almeida, V. Almeida, D. Ardagna, C. Francalanci, and M. Trubian, "Resource Management in the Autonomic Service-Oriented Architecture," In Proc. of International Conference on Autonomic Computing,pp. 84-92, 2006.

[8] P. Padala, K. G. Shin, X. Zhu, M. Uysal, Z. Wang, S. Singhal, A. Merchant, and K. Salem, "Adaptive Control of Virtualized Resources in Utility Computing Environments," In EuroSys '07: Proceedings of the 2nd ACM SIGOPS/EuroSys European Conference on Computer Systems, Lisbon, Portugal, pp. 289 – 302, 2007.

[9] X. Wang, D. Lan, G. Wang, X. Fang, M. Ye, Y. Chen and Q. Wang, "Appliance-based Autonomic Provisioning Framework for Virtualized Outsourcing Data Center. Autonomic Computing," Fourth International Conference on Autonomic Computing, Jacksonville, Florida, USA, pp. 29 – 29, 2007.

[10] S. Jones, A. Arpaci-Dusseau, and R. Arpaci-Dusseau, "Geiger: Monitoring the Buffer Cache in a Virtual Machine Environment," In Proc. ASPLOS' 06, pp. 13-23, October 2006.

[11] K. Govil, D. Teodosiu, Y. Huang, and M. Rosenblum, "Cellular Disco: Resource Management using Virtual Clusters on Shared Memory Multiprocessors," In Proc. SOSP'99, Dec.1999. pp. 154-169.

[12] C. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow, M. Lam, and M. Rosenblum, "Optimizing the Migration of Virtual Computers," In Proc. of the 5th symposium on Operating systems design and implementation, Vol. 36 , Issue SI, pp. 377 – 390, 2002.

[13] VMware, http://www.vmware.com. Last access on 2010.08.22.

[14] Xen, http://www.xen.org, Last access on 2010.08.22.

[15] H. Van and J. Menaud, "Autonomic Virtual Resource Management for Service Hosting Platforms," In Proc. of the 2009 ICSE Workshop on Software Engineering Challenges of Cloud Computing, pp. 1-8, 2009.

[16] P. Ruth, J. Rhee, D. Xu, R. Kennell, and S. Goasguen, "Autonomic Live Adaptation of Virtual Computational Environments in a Multi-Domain Infrastructure," In Proc. IEEE ICAC '06, pp. 5 – 14, June, 2006.

[17] H. Nakada, T. Hirofuchi, H. Ogawa and S. Itoh, "Toward Virtual Machine Packing Optimization Based on Genetic Algorithm", Proceedings of the 10th International Work-Conference on Artificial Neural Networks: Part II: Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living, Salamanca, Spain, pp. 651 – 654, 2009.

[18] P. Campegiani, "A Genetic Algorithm to Solve the Virtual Machines Resources Allocation Problem in Multi-tier Distributed Systems," Second International Workshop on Virtualization Performance: Analysis, Characterization, and Tools (VPACT'09), Boston, Massachusett, April, 2009.

[19] G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: a System for Energy Efficient Computing in Virtualized Environments," In Proceedings of the 2009 International Symposium on Low-Power Electronics and Design (ISLPED'09), pages 243-248, New York, NY, USA, 2009.

[20] A. Verma, P. Ahuja, and A. Neogi. "pMapper: Power And Migration Cost Aware Application Placement in Virtualized Systems," In Proc. of 9th ACM/IFIP/USENIX International Conference on Middleware, Leuven, Belgium, pp. 243-264, 2008.

[21] A. Verma, P. Ahuja and A. Neogi, "Power-Aware Dynamic Placement Of HPC Applications," In Proc. of the f the 22nd annual international conference on Supercomputing, Island of Kos, Greece, pp. 175-184, 2008.

[22] G Khanna, K Beaty, G Kar, A Kochut, "Application Performance Management in Virtualized Server Environments," Network Operations and Management Symposium 2006 NOMS 2006 10th IEEEIFIP, pp. 373 – 381, April, 2006.

[23] M.Tarighi, S.A.Motamedi and S.Sharifian, "A New Model for Virtual Machine Migration in Virtualized Cluster Server Based on Fuzzy

Decision Making," Journal of Telecommunications, vol. 1, issue 1, pp. 40-51, Feb. 2010.

[24] M. Andreolini, S. Casolari, M. Colajanni and M. Messori, "Dynamic Load Management of Virtual Machines in a Cloud Architectures," First International Conference on Cloud Computing (ICST CLOUDCOMP2009), Munich, Germany, October 19-21, 2009.

[25] N. Bobroff, A. Kochut and K. Beaty, "Dynamic Placement of Virtual Machines for Managing SLA Violations," IEEE/IFIP International Symposium on Integrated Network Management (IM), Munich, Germany, pp. 119 – 128,May 21-25, 2007.

[26] T. Wood, P. Shenoy, A. Venkataramani, M. Yousif, "Black-Box and Gray-Box Strategies for Virtual Machine Migration," 4th USENIX Symposium on Networked Systems Design USENIX Association & Implementation, pp. 229–242, 2007.

[27] F. Hermenier, X. Lorca, J. Menaud, G. Muller and L. Lawall, "Entropy: a Consolidation Manager for Clusters," In proc. of the 2009 International Conference on Virtual Execution Environments (VEE'09), Washington, DC, USA, pp. 41-50, Mar. 2009.

 ISBN: 978-1-61208-106-9