Dynamic Virtual Machine Placement Considering CPU and Memory Resource Requirements

Abdelkhalik Mosa and Rizos Sakellariou
School of Computer Science
The University of Manchester, UK
{abdelkhalik.mosa,rizos}@manchester.ac.uk

***Abstract*—In cloud data centers, cloud providers can offer computing infrastructure as a service in the form of virtual machines (VMs). With the help of virtualization technology, cloud data centers can consolidate VMs on physical machines to minimize costs. VM placement is the process of assigning VMs to the appropriate physical machines. An efficient VM placement solution will result in better VM consolidation ratios which ensures better resource utilization and hence more energy savings. The VM placement process consists of both the *initial* as well as the *dynamic* placement of VMs. In this paper, we are experimenting with a dynamic VM placement solution that considers different resource types (namely, CPU and memory). The proposed solution makes use of a genetic algorithm for the dynamic reallocation of the VMs based on the actual demand of the individual VMs aiming to minimize under-utilization and over-utilization scenarios in the cloud data center. Empirical evaluation using CloudSim highlights the importance of considering multiple resource types. In addition, it demonstrates that the genetic algorithm outperforms the well-known best-fit decreasing algorithm for dynamic VM placement.**

***Index Terms*—dynamic VM placement, genetic algorithm, CPU and memory resource allocation.**

I. Introduction

The Cloud computing paradigm offers computing resources in the form of elastic, on-demand, measured services over a network [1]. Cloud data centers rely on virtualization technology to consolidate multiple virtual machines (VMs) on a single physical machine (PM), which saves energy and consequently minimizes CO_2 emissions and associated costs [2].

In a cloud data center, a VM placement solution can consolidate VMs by allocating VMs to suitable PMs in a way that achieves the cloud provider's objectives [3]. VM placement is a two-fold process, which includes initial (static) placement, and dynamic re-placement of the VMs. The initial VM placement receives new VM placement requests and assigns VMs to PMs based on the requested VM size. Due to the dynamic nature of cloud data centers (*e.g.* VMs' workload changes over time [4]), initial VM placement techniques may result in suboptimal placement solutions that can lead to either *over-provisioning* or *under-provisioning* of resources. Over-provisioning indicates that the allocated resources exceed the required resource demand which leads to *under-utilization* of the PMs. Under-provisioning means that the allocated resources do not meet the demand which eventually leads to PM *over-utilization*. Under-utilization leads to energy waste and higher costs, while over-utilization increases the number of service level agreement violations (SLAVs). Dynamic VM placement uses a live migration utility to reallocate VMs and achieve an efficient VMs-to-PMs assignment. A dynamic VM placement solution can help mitigate the costs associated with under-utilization or over-utilization.

A high proportion of current VM placement solutions consider only CPU resources and ignore other types of resource such as memory, storage, or network bandwidth. However, considering different types of resources (or characteristics) including memory or network bandwidth becomes critical as many VMs may be either memory-intensive or bandwidth-intensive, which suggests that the CPU is not the most deciding resource type in such cases.

This paper explores the problem of allocating VMs along more than one of these characteristics by proposing a dynamic demand-based VM placement solution that considers both CPU and memory requirements. The proposed solution adopts a genetic algorithm (GA) aiming to minimize under-utilization as well as over-utilization of both CPU and memory to ensure better overall utilization and reduce any SLAVs.

In the remainder of this paper, Section II describes some existing solutions that use either heuristics or meta-heuristics for dynamic VM placement. Following this, Section III describes the objective of dynamic VM placement. Then, Section III briefly presents the dynamic VM placement solution using a GA. Finally, Section V describes the conducted experiments and results and the paper is concluded in Section VI.

II. Related Work

Initial VM placement has been extensively studied in the literature [3]. The dynamic reallocation of VMs is typically achieved using heuristics or meta-heuristics. Examples of dynamic VM placement solutions that use heuristics to reduce energy consumption while keeping SLAVs below a certain level include [5]–[8]. Many dynamic VM placement solutions use meta-heuristics to dynamically reallocate VMs among PMs based on a predefined objective function. Some of these solutions adopt a genetic algorithm [9], [10]; an ant colony system [11], while others use particle swarm optimization [12]. However, many of these solutions only consider a single type of resource, often CPU [5], [6], [10], [12]. Other solutions consider multiple resources using a genetic algorithm, but they only solve the initial VM placement and ignore the dynamic reallocation of the VMs [13]. There is scope for research considering different types of resources, an area explored by this paper, which proposes a CPU- and memory-aware dynamic VM placement solution based on a genetic algorithm.

III. Placement Objective

The main objective for the VM placement is to utilize the PMs efficiently. Therefore, we are introducing an objective function that seeks to utilize the PMs efficiently by minimizing the estimated under-utilization per each resource type (CPU and memory), while reducing SLAVs by minimizing the estimated over-utilization at each scheduling interval. Under-utilization is defined as the ratio of available (or unused) capacity divided by the total capacity of a given resource type. Over-utilization is defined as the demand that exceeds the capacity of a given resource type.

$$\text{Minimize } \left(\sum_{r=1}^{n}(\bar{W}^r + \bar{O}^r)\right) \tag{1}$$

In Equation 1, n represents the number of resource types (CPU and memory in this paper but this could be generalized to storage, bandwidth, etc); $\bar{W}^r$ is the average under-utilization per resource type r of all PMs running at a scheduling interval; and $\bar{O}^r$ is the average over-utilization per resource type r for all PMs running on the current scheduling interval.

IV. Dynamic VM Placement using a Genetic Algorithm

The proposed dynamic VM placement solution solves both the initial and dynamic VM placement problems. For the initial VM placement, it adopts a power-aware best-fit decreasing (BFD) heuristic [5] that allocates VMs based on the resource requirements defined by VM types. Then, dynamic reallocation of the VMs may happen during each scheduling interval (periodically every five simulated minutes) using a GA that searches for a VMs-to-PMs mapping that can reduce under-utilization and over-utilization based on the objective function defined by Equation 1.

Algorithm 1 shows the pseudo-code of the GA for the dynamic reallocation of the VMs after initial placement. The GA runs at each scheduling interval and for a specified number of times according to the number of generations. At each scheduling interval, the algorithm tries to find source and target PMs. The source and target PMs are the PMs that are either a source or a target of VMs' migrations. Source PMs are the ones that are not switched off and host running VMs, while target PMs are the ones that are not over-utilized before of after the VMs' migration. The initial population for VMs-to-PMs mapping is created by mutating the current mapping. For each generation, the GA first selects the parent using tournament selection, applies uniform crossover, mutates the VMs and evaluates the candidate VMs-to-PMs mapping based on the objective function (Section III).

Once the GA finds the best VMs-to-PMs mapping for a given scheduling interval, it will enact VM migration to reflect the new VMs-to-PMs mapping.

Algorithm 1 Dynamic reallocation of VMs using a GA

```
1:  Input: schedulingIntervals, generations;
2:  for all (schedulingIntervals) do
3:      Store current VMs-to-PMs mapping;
4:      Find source and target PMs;
5:      Build initial population of VMs-to-PMs mapping;
6:      Evaluate candidate mappings using Equation 1;
7:      for all (generations) do
8:          Select parents for crossover;
9:          Crossover based on the cross over ratio;
10:         Mutate VMs using only the source PMs (active);
11:         Evaluate candidate mappings using Equation 1;
12:         Select population for next generation;
13:     Save best allocation found so far;
14:     Build migration map from the best found allocation;
15:     Apply VM migration based on the migration map;
```

V. Experimental Evaluation

A. Simulation Settings

The CloudSim simulation toolkit [14] is used to simulate a cloud data center, which consists of two types of PMs, namely, HP ProLiant ML110 (2 cores at 1860 MHz each) and HP ProLiant ML110 G5 (2 cores at 2660 MHz each) that can host four different VM types. CloudSim generates synthetic CPU and RAM utilization traces every five simulated minutes based on a uniform normal distribution with values between zero and one. The simulation works for a whole day, and the dynamic reallocation of the VMs takes place every five minutes (scheduling interval). For testing different capabilities, we have also skewed memory utilization to be more than 50% in some experiments.

Using the baseline BFD heuristic for dynamic VM placement, this starts migrating VMs from over-utilized PMs when the utilization level of the CPU or memory is 100%. It chooses VMs with minimum memory sizes for migration from over-utilized PMs to minimize the total migration time.

B. Performance Metrics

1) *Average overall under-utilization (AOUU)*:
AOUU is the average of ACUU and ARUU, which correspond to average under-utilization per CPU and memory, respectively.
2) *Average overall SLA violations (AOSLAVs)*:
AOSLAVs estimates the average number of SLA violations and is calculated as follows:

$$AOSLAVs = \frac{1}{2}(OCSLAVs + ORSLAVs),$$

where OCSLAVs represents CPU SLA violations for all active PMs and is estimated as follows:

$$OCSLAVs = \frac{RequestedCPU - AllocatedCPU}{RequestedCPU}.$$

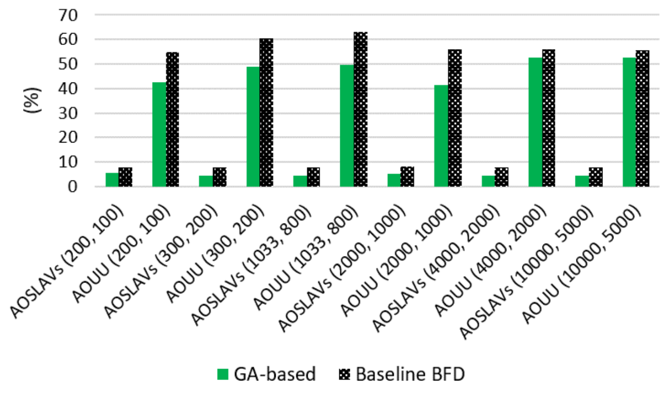

Fig. 1. AOSLAVs and AOUU using different data center sizes (#VMs,#PMs)

The ORSLAVs represents memory SLA violations for all active PMs and is estimated as follows:

$$ORSLAVs = \frac{RequestedRam - AllocatedRam}{RequestedRam}.$$

Experiment 1: Normal CPU and memory utilization: Figure 1 summarizes the results of the proposed GA-based solution as opposed to the baseline BFD heuristic using different data center sizes ranging from 100 to 5000 PMs that host 200 to 10,000 VMs. The X-axis shows the AOSLAVs and AOUU (for different sizes of the data center) with the Y-axis showing their values as a percentage. The experiments run for a whole day of simulation time except the instances with 4,000 and 10,000 VMs that run for only 6 hours due to memory constraints.

Experiment 2: Skewed memory utilization: This experiment skews memory utilization (over 50%) to assess the reaction of both solutions to the increase in memory requirements. Figure 2 shows that the ACUU of the proposed GA-based solution is 45.95% compared to 57.97% of the baseline BFD solution, which means around 20% improvement in CPU utilization. Moreover, the ARUU is 22.56% and 38.69% for the proposed GA-based solution and the baseline BFD solution, respectively, which means memory utilization is improved by around 40%. In Figure 2, the size of the circle defines the AOSLAVs. The proposed GA-based solution managed to reduce the AOSLAVs by around 70%.

VI. Conclusion and Future Work

This paper addresses the dynamic VM placement problem by considering both CPU and memory resource requirements of VMs. The paper makes use of a genetic algorithm for the dynamic reallocation of the VMs. The candidate solutions are evaluated against an objective function that aims to reduce over-utilization as well as under-utilization of PMs to minimize the resulting SLAVs and improve overall utilization. Possible directions for future work include: (1) considering other types of resources such as network bandwidth or storage; (2) conducting more experiments using real workload traces.

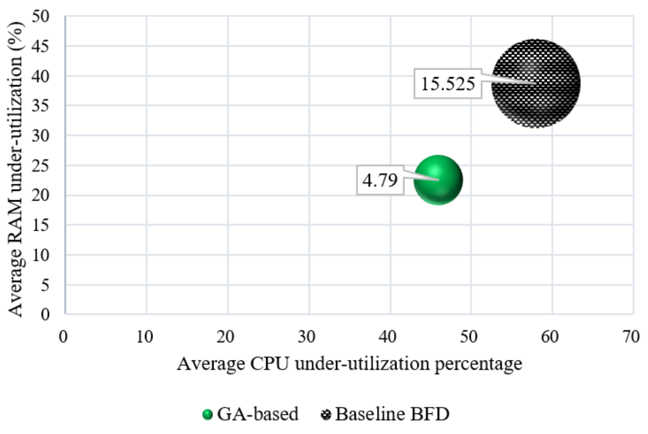

Fig. 2. Average CPU and memory under-utilization along with AOSLAVs in an increased memory utilization scenario

References

[1] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

[2] VMWare, "Server Consolidation and Containment With Virtual Infrastructure." [Online]. Available: http://www.vmware.com/pdf/server_consolidation.pdf

[3] I. Pietri and R. Sakellariou, "Mapping virtual machines onto physical machines in cloud computing: A survey," *ACM Computing Surveys (CSUR)*, vol. 49, no. 3, p. 49, 2016.

[4] C. Hyser, B. McKee, R. Gardner, and B. J. Watson, "Autonomic virtual machine placement in the data center," *Hewlett Packard Laboratories, Tech. Rep. HPL-2007-189*, vol. 189, 2007.

[5] A. Beloglazov and R. Buyya, "Optimal online deterministic algorithms and adaptive heuristics for energy and performance efficient dynamic consolidation of virtual machines in cloud data centers," *Concurrency and Computation: Practice and Experience*, vol. 24, no. 13, pp. 1397–1420, 2012.

[6] X. Wang, X. Liu, L. Fan, and X. Jia, "A decentralized virtual machine migration approach of data centers for cloud computing," *Mathematical Problems in Engineering*, vol. 2013, 2013.

[7] Y. Liu, X. Sun, W. Wei, and W. Jing, "Enhancing energy-efficient and qos dynamic virtual machine consolidation method in cloud environment," *IEEE Access*, vol. 6, pp. 31 224–31 235, 2018.

[8] T. H. Nguyen, M. D. Francesco, and A. Yla-Jaaski, "Virtual machine consolidation with multiple usage prediction for energy-efficient cloud data centers," *IEEE Transactions on Services Computing*, pp. 1–1, 2018.

[9] C. T. Joseph, K. Chandrasekaran, and R. Cyriac, "A novel family genetic approach for virtual machine allocation," *Procedia Computer Science*, vol. 46, pp. 558 – 565, 2015, proceedings of the International Conference on Information and Communication Technologies, ICICT 2014.

[10] A. Mosa and N. W. Paton, "Optimizing virtual machine placement for energy and sla in clouds using utility functions," *Journal of Cloud Computing*, vol. 5, no. 1, p. 17, 2016.

[11] F. Farahnakian, A. Ashraf, T. Pahikkala, P. Liljeberg, J. Plosila, I. Porres, and H. Tenhunen, "Using ant colony system to consolidate vms for green cloud computing," *IEEE Transactions on Services Computing*, vol. 8, no. 2, pp. 187–198, 2015.

[12] S. Wang, Z. Liu, Z. Zheng, Q. Sun, and F. Yang, "Particle swarm optimization for energy-aware virtual machine placement optimization in virtualized data centers," in *2013 International Conference on Parallel and Distributed Systems*, Dec 2013, pp. 102–109.

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

[14] R. N. Calheiros, R. Ranjan, C. A. F. De Rose, and R. Buyya, "Cloudsim: A novel framework for modeling and simulation of cloud computing infrastructures and services," *arXiv preprint arXiv:0903.2525*, 2009.