Dynamic optimization of VM to Server mapping

B.Pavithraselvi*[1], B.Sandhya[2]

*[1,2]Kathir College of engineering, Coimbatore, Tamil Nadu 641062, India
*[1]pavithraslv001@gmail.com

Abstract

Background/Objectives: To dynamically optimize the Virtual machines for server mapping in the cloud environments to avoid server underutilization and overloading of the organizations utilizing virtualized data centresso that the overall power consumption and carbon emissions are minimized.
Methods/Statistical analysis: To avoid server underutilization, server consolidation plans at minimizing the number of server machines utilized in the data centres by consolidating load and improving resource utilization of physical systems. Server consolidation of virtual machines (VMs) via live migration and exchanging idle nodes to the sleep mode permit Cloud providers to improve resource usage and minimize energy consumption. Server consolidation during live migration is an effective method in the direction of energy conservation in cloud data centers. Even though a batch of research and review has been performed on server consolidation, a variety of problems involved have mainly been offered in isolation of each other. In this paper, initiate with a set of heuristic approaches for dynamic optimization of the VM-to-server mapping based on grouping of fundamental management actions, such as removing and restarting physical machines, VM migration, and removing and restarting VMs.
Findings: The proposed approach utilizes the service consolidation to avoid server underutilization and overloading to enhance the usage of several-utilized servers in data centres acquiring reduced system management cost. Thus the server is utilized efficiently in the organization with low management cost.
Improvements/Applications: Overall utilization of the datacenter is improved using our approach and the power consumption and carbon emissions are minimized.

Keywords: Cloud computing,dynamic optimization, virtual machines, server mapping

1. Introduction

Cloud computing permits users to extent up and down their resources based on requires. An additional important role of cloud computing platform is to dynamically balance the load between different servers in order to avoid hotspots and improve resource utilization. Load balancing of the whole system can be switched dynamically by using virtualization technology where it becomes promising to remap virtual machine and physical resources according to adjust in load. Due to these benefits, virtualization technology is being systematically implemented in cloud computing. Moreover, in order to accomplish the best performance, the virtual machines have to completely utilize its services and resources by adapting to the cloud computing environment dynamically. Virtual machine live migration knowledge builds it feasible to assigning between the virtual machines (VMs) and the physical machines (PMs) even as applications are running. Live migration enhances the resource utilization and offers the improved performance result.

To date, most research on Virtual Machine (VM) provisioning for cloud datacenters has focused on deploy time scheduling, typically formulated as assignment problems where VMs are mapped to Physical Machines (PMs). Common objectives for these formulations are to optimize criteria such as Service Level Agreements (SLAs), provider revenue, performance, utilization etc. or a combination thereof. Notably, there are several factors that complicate this problem. First of all, VM scheduling is an online problem as both the arrival rate of new VM requests and the completion time for provisioned VMs is unknown. In addition, resource usage of individual VMs also varies over time. Changes to the server pool, due to failures or energy-management actions such as power-off and frequency-scaling, can also impact the performance of deployed VMs.

These factors imply that any scheduling solution may become suboptimal over time. To address this, we propose a Dynamic VM migration and Initialization approach to optimize VM provisioning as a balance to VM scheduling. Our method consists of a set of approaches that enable cloud infrastructure providers to dynamically reconfigure the mapping of VMs to PMs and adjust to the changes in workloads and the physical framework. These methods are based on a grouping of management actions, i.e., remove and restart of PMs, live VM migration, and remove and

restart of VMs. This approach goals to maximize cloud provider profits over time by reducing power consumption, improving PM utilization, and prioritizing significant VM requests.

Optimal mapping of acknowledged VMs to a set of PMs in order to gain improve profit whereas submitting to all SLAs specified by customers is difficulty for cloud providers as it is in common a NP-hard problem [1], [2]. Different algorithms have been proposed to generate near-optimal placement methods[3], who present an improved genetic algorithm planed to optimize probably conflicting objectives, together with making effective usage of multidimensional resources, avoiding hotspots, and reducing power consumption.

Live migration of running VMs is therefore a necessity. A complete study on principles and performance of live migration methods (with pre-copy, post-copy and hybrid) is existing in [4], which also examines how migration downtime can be minimized. [5] describe a model for joint optimization of data center deployment, VM assignment, and migration. Based on fluctuations in network performance (latency), they propose a technique based on network flow maximization to estimation VM migration cost by amortizing it to the latency of each access. [6]Escribe a bin packing technique to allocation and migration of VMs in data centres and likewise, [7] merge bin packing with resource usage prediction to dynamically optimize VM placement. Autoregressive methods are exercised for predictions and the number of VM migrations is reduced (avoiding ping-pong effects) supported on the predictions. [8] plan to reduce VM execution time via a knapsack formulation for VM placement. A hybrid on-line off-line method is utilized where VM migrations are merged with the knapsack placement algorithm. A defragmentation method by [9] involves two algorithms while [10] propose simple destination exchange scheme for VMs in order to minimize network traffic.

A huge volume of attempt has been devoted to server consolidation processes based on workload analysis, planing at enhancing efficiency in cloud infrastructures [11], [12], [13]. Additional methods for isolation of resources in hardware [14], [15], or software [16] have been implemented to minimize the performance degradation initiated by consolidation of multiple VMs on a same server. Moreover, these approaches generally operate in off-line ways which are not able to dynamically and efficiently adapt the cloud to adjust (including workload variations, system failures, etc). They neither take VM pricing methods and monetary penalty for SLA violation into consideration. In [17] proposed a framework to detect application performance deviations and a VM migration mechanism to handle these. They proposed a set of server consolidation heuristics based on VM migration costs and server residual capacity.

Autonomic Computing [18] initiative was initiated by IBM, who also introduced the MAPE-K reference model. Its goal is to build computing systems that can manage themselves given high-level objectives from administrators [19]. In the MAPE-K model, with the support of a Knowledge base, the system Monitors the managed elements, analyzes the data monitored, and finally Plans and Executes suitable actions to ensure the system is in a desired state. Although considerable progress has been achieved in the past few years, the original vision remains unfulfilled as even more complexity is added to the system due to the convergence of new technologies and new applications [20]. The main goal of this work is to maximize the monetary profit of running a datacenter by automatic adaption to both internal and external changes, and thus it is within the scope of autonomic computing.

2. Dynamic Optimization Of The VM To Server Mapping

At any point in time, many events can acquire place. We describe four actions and prioritize them in downwards order as PM crash, VM turn-off, VM arrival and VM initialize. Our proposed method dynamically manages these actions following to their priorities, and adjusts the cloud infrastructure to the modified in a proactive method. Easy management events are utilized for improve the datacenter, i.e., (i) remove/restart VMs, (ii) VM migration, (iii) remove/restart PMs and (iv) VM initialization. For an event of PM crash, a crashed PM not only changes all VMs hosted, but it should also be excluded as a potential destination for VMs. Upon an action of PM crash, we easily remove all VMs hosted on that PM. A VM turn-off event has a higher priority than a VM arrival event, as capacity released by a terminated VM can be used to accept more VMs into the datacenter. When a VM terminates, the taken resources are released, improving the remaining capacity of a PM. All VMs arrival is added to a list such as VMList, which also may contain VMs removed in the past. VMs in VMList can be potentially executed depending on the decision by the optimization process. VMs initialization event taken place for migration is not possible with used deadline create a new VM and add them to VMList. An actions taken on the occurrence of events is shown in Algorithm 1.

Algorithm 1: ManageAction(Actions)

1. for a∈ Actions do
2 if PM pm crash then
3 Eliminate pm and all VMs hosted;
4 else if VM vm turn-off then
5 Release capacity occupied by vm;
6 else if VM vm arrive then
7 Add vm to VMList;
8. else if VM vm initialize then
9. Create a VM in PM and add vm to VMList

Once the action managing process is completed, a consolidation action offered in Algorithm 2 is generated to improve the profit achieved by VM provision. In general, if an infrastructure provider has too limited capacity, the profit can be improved by choosing which VMs to run.

Algorithm 2: consolidation ()

1 if remove/restart VM is allowed then
2 Add all removed VMs to VMList;
3 Sort VMs in VMList by (price + penalty) in descending order;
4 for vm∈ VMList do
5 manageVM(vm);
6 if remove/restart PM is allowed then
7 if VM migration is allowed then
8 releasePMsbyMigration(); // Release PMs via VM migration.
9 else
10 removeIdlePMs(); // Remove PMs without VM running.
11. if VM migration is not allowed then
12. InitializeVM(); // Initialize a new VM

In order to improve the datacenter operation, our method is to create a list of consolidation actions following to Algorithm 2. If the algorithm permits for remove/restart of VMs, the initial step is to reprocess all the presently removed VMs and allow them to be probably restarted by adding them to VMList (see Line 2). All VMs (also referred to as object VMs) in VMList are to be managed sequentially, in a downward order that they are ranked by (price + penalty). There are two feasible results of the action manageVM, i.e., either remove the current VM, or place and start VM in some physical machine. The final step in a round of optimization is to remove idle PMs if the feature is allowed (see Line 6–10). More PMs may be freed and then removed, depending on whether VM migration is allowed. If VM migration is not allowed new VM is initialized and add them in VMList (see Line 11-12).

Algorithm 3: manageVM(vm)

/* this task is for managing newly arrival VMs and VMs that were removed in the preceding period. */
1 pms→ active PMs;
2 pm← best-fit(vm, pms); // locate a PM for vm via the best-fit strategy.
3 if pm not found and VM migration is allowed then
4 pm←findPMbyMigration(vm);
5 if pm not found then
6 pm ←attempt to start a new PM;
7 if pm not found and remove/restart VM is allowed then
8 pm ←findPMwithVictimVM(vm); // locate a victim VM and replace it with vm.
9 if pm found then
10 placeVM(vm,pm);
11 else
12 removeVM(vm);

As described in Line 2 in Algorithm 3, we apply best-fit as the baseline approach to discover an active PM for a VM. The motivation for this is to load every PM as much as feasible, improving the utilization of the PMs and thus

reducing the residual capacity of the entire infrastructure. If this is not sufficient (i.e., no PM can host the VM), an easy solution is to try starting a removed (or a new) PM (see Line 6) and put the VM there. Moreover, in order to reduce the total number of active PMs, prior to beginning a new PM, the proposed algorithm attempts to readjust the placement of VMs, to see if there exists a PM that can host the VM following migrating some VMs to other PMs (see Line 4). The information of this can be set up in Algorithm 4).

At last, if no suitable PM is found after attempting all of the above methods, an aggressive method is functional to select one of the running VMs as the victim, remove it, and replace it with the object VM (see Line 8, as described in Algorithm 5). This step is carrying out only if replacing the victim with the object VM is possible and more gainful. Note that our algorithm presently only chooses one VM as victim; it is moreover feasible to expand this to allow selection of multiple VMs as victims instead.

To discover if re-arranging the mapping of VMs by live migration they can build room for vm on some PM, all active PMs are sorted by residual capacity in downward order in Algorithm 4. The PMs are then estimated by looking at the possibility of migrating a set of VMs to other PMs. When estimating a PM, we only provide migrating VMs that are smaller than vm (see Line 6), as testing a VM larger than vm is meaningless (such as, if a PM can be found for this case, immediately put vm there without adjusting any VM placement). In addition, note that as a system can be characterized by various dimensions (CPU, memory, storage, etc), the size function in Algorithm 4 can have various descriptions depending on the application situations. In this work, it is described in terms of CPU cores as other dimensions (memory, storage, etc.) are used as constraints when evaluating the possibility of placement on PMs. The algorithm also attempts to reduce the number of migrated VMs, as migration obtains time and consumes resources. As well as, to further minimize the number of VMs migrated; all potential VMs are sorted by size in downward order (see Line 7). VMs to be migrated are added to a plan by function addToMigratitonPlan (see Line 13). The evaluation process stops when the initial suitable PM is found, and the migration plan is performed by commitMigratitonPlan (see Line 20). If a PM is not suitable, the migration plan is canceled by cancelMigratitonPlan (see Line 18).

Algorithm 4: findPMbyMigration(vm)

```
/* Find a PM that can host vm after migrating some VMs to other PMs. */
1 pms← all active PMs;
2 Sort pms by residual capacity in downward order;
3 for p ∈ pms do
4 feasible ←FALSE;
5 vmSet←VMs hosted in p;
6 vms← {v ∈vmSet | size(v) < size(vm)};
7 Sort vms by capacity in downward order;
8 pmset← pms\{p};
9 for v ∈ vms do
10 pm best-fit(v, pmset);
11 if pm not found then
12 break;
13 addToMigratitonPlan(v;pm);
14 if p can host vm then
15 feasible ←TRUE;
16 break;
17 if not feasible then
18 cancelMigratitonPlan();
19 continue;
20 commitMigratitonPlan();
21 return p;
```

Algorithm 5 initiates the approach of finding a victim VM to be replaced by a VM in VMList. The fundamental concept is to choose the VM with the minimum value of (price + penalty) between all choose able VMs (see Line 9-13). Once a VM is choosed as a victim VM, it is removed and moved to a waiting list and may possibly be restarted depending on the future optimization decision.

ISSN (online) : 2277-5390
ISSN (Print): 2277 - 5382

Algorithm 5: findPMwithVictimVM(vm)

```
1 destination ←null; /* Find a PM that can host vm after removing a vm hosted. */
2 minRF←price(vm) + penalty(vm);
3 pms←all active PMs;
4 for p ∈ pms do
5 vmSet← VMs hosted in p;
6 vms← {v 2 vmSet | price(v) + penalty(v) < minRF};
7 Sort vms by (price + penalty) in ascending order;
8 for v ∈vms do
9 if p can host vm after removing v then
10 victim ←v;
11 destination ←p;
12 minRF←price(v) + penalty(v);
13 break;
14 if destination is not null then
15 removeVM(victim);
16 return destination;
```

The third action in every consolidation procedure is to attempt to minimize the power consumption of the infrastructure by removing all idle PMs, or by releasing more PMs by VM migration. To vacant an active PM, all hosted VMs required to be migrated to other PM(s). Spontaneously, PMs with higher residual capability are more likely able to be emptied, and thus PMs are calculated in the order of residual capability (see Line 2 in Algorithm 6). Once again, we utilize a best-fit strategy at any time discovering a new location for a VM (see Line 8).

Algorithm 6: releasePMsbyMigration()

```
/* Release PMs through VM migration */
1 pms←all active PMs;
2 Sort pms by residual capacity in descending order;
3 for p ∈ pms do
4 feasible ←TRUE;
5 vms←VMs hosted in p;
6 pmset←pms \ {p};
7 for vm∈ vms do
8 pm ←best-fit(vm, pmset);
9 if pm not found then
10 feasible←FALSE;
11 break;
12 addToMigratitonPlan(vm,pm);
13 if not feasible then
14 cancelMigratitonPlan();
15 continue;
16 commitMigratitonPlan();
17 suspendPM(p);
```

Final action of this work is migration is not possible we can initialize a new VM with a suitable PM. Based on the user requirements create a VM and select the feasible PM for place the VM. Choose the PM based on the capacity for hold the VM.

Algorithm 7: InitializeVM()

```
// create a new VM if migration is not possible
1. Get the requirements for VM;
2. minprice←null
3. minpenalty←null, allocatehost←null
```

4. pms← all active PMs;
5. Sort pms by residual capacity in descending order;
6. for p ∈ pms do
7. if p has residual capacity for vm then
8. price←calculate(p,vm)
9. penalty←calculate(p,vm)
10. if price<minprice&& penalty<minpenalty then
11. minprice←price
12. minpenalty←penalty
13. feasible←TRUE;
14. ifallocatedhost≠null then
15. allocatevm to p
15. return allocation

3. Conclusion

We present a Dynamic VM migration and Initialization approach for cloud infrastructure providers to improve their PM utilization and increase profits. Based on a set of essential management actions, our approach can rearrange the VM to PM mapping during operation to improve the resource utilization of the cloud infrastructure, thereby maximize the revenue by prioritizing more profitable workloads and reducing energy consumption. In this work based on the user requirements initialize the VM and Balance the work load among PMs. In the proposed work we calculate the migration cost based on communication time and traffic of the network from source PM to destination PM. Based on the migration cost, VM are migrated or else VM is initialized. The feasibility and performance of our work, consisting of optimization algorithms and dynamic datacenter consolidation software, is evaluated by simulations on a cloudsim. Outcome of this research shows that overall utilization of the datacenter is improved while minimizing the power consumption.

4. References

1. W. Li, J. Tordsson, E. Elmroth. Virtual Machine Placement for Predictable and Time-Constrained Peak Loads. In Proceedings of the 8th International Conference on Economics of Grids, Clouds, Systems, and Services (GECON'11). Lecture Notes in Computer Science, Springer-Verlag, 2011; 7150, 120-134.
2. A. Roytman, A. Kansal, S. Govindan, J. Liu, S. Nath. PACMan: Performance Aware Virtual Machine Consolidation. In Proceedings of the 10th International Conference on Autonomic Computing, Berkeley, CA, USENIX. 2013; 83–94.
3. J. Xu, J. A. Fortes. Multi-objective Virtual Machine Placement in Virtualized Data Center Environments. In Proceedings of the 2010 IEEE/ACM International Conference on Green Computing and Communications & 2010 IEEE/ACM International Conference on Cyber, Physical and Social Computing, IEEE, 2010;179-188.
4. P. Svard, J. Tordsson, E. Elmroth, S. Walsh, B. Hudzia. The Noble Art of Live VM Migration - Principles and Performance of Precopy and Postcopy Migration of Demanding Workloads. 2014.
5. Y. Li, M. Yao, C. Lin. Joint Study on Optimizations of Data Center Deployment, VM Assignment and Migration. In Proceedings of the IEEE/ACM 21st International Symposium on Quality of Service (IWQoS), IEEE, 2013; 1-10.
6. W. Song, Z. Xiao, Q. Chen, H. Luo. Adaptive Resource Provisioning for the Cloud using Online Bin Packing. IEEE Transactions on Computers, 99(PrePrints), 2013.
7. K. Sato, M. Samejima, N. Komoda. Dynamic Optimization of Virtual Machine Placement by Resource Usage Prediction. In Proceedings of the 11th IEEE International Conference on Industrial Informatics (INDIN), IEEE, 2013; 86-91.
8. K. Li, H. Zheng, J. Wu. Migration-based Virtual Machine Placement in Cloud Systems. In Proceedings of the IEEE 2nd International Conference on Cloud Networking (CloudNet), IEEE, 2013; 83-90.
9. G. Shanmuganathan, A. Gulati, P. Varman. Defragmenting the Cloud using Demand-based Resource Allocation. In Proceedings of the ACM SIGMETRICS/international Conference on Measurement and Modeling of Computer Systems, ACM, 2013; 67-80.
10. C. Avin, O. Dunay, S. Schmid. Simple Destination-Swap Strategies for Adaptive Intra-and Inter-Tenant VM Migration. CoRR, 2013.

11. G. Jung, K. R. Joshi, M. A. Hiltunen, R. D. Schlichting, C. Pu. A Cost-sensitive Adaptation Engine for Server Consolidation of Multitier Applications. In Middleware 2009, Springer, 2009; 163-183.
12. S. Srikantaiah, A. Kansal, F. Zhao. Energy aware Consolidation for Cloud Computing. In Proceedings of the 2008 Conference on Power aware Computing and Systems, volume 10. USENIX Association, 2008.
13. A. Verma, G. Dasgupta, T. K. Nayak, P. De, R. Kothari. Server Workload Analysis for Power Minimization using Consolidation. In Proceedings of the 2009 Conference on USENIX Annual Technical Conference, USENIX Association, 2009; 28-28.
14. R. Iyer, R. Illikkal, O. Tickoo, L. Zhao, P. Apparao, D. Newell. VM: Measuring, Modeling and Managing VM Shared Resources. Computer Networks, 2009; 53(17), 2873–2887.
15. B. Santhosh Kumar, LathaParthiban,An Energy Efficient Data Centre Selection Framework for Virtualized Cloud Computing Environment. Indian Journal of Science and Technology, 2015; 8(35),1-6.
16. A. Verma, P. Ahuja, A. Neogi. Power-aware Dynamic Placement of HPC Applications. In Proceedings of the 22nd Annual International Conference on Supercomputing. ACM, 2008; 175-184.
17. B. Lakshmipriya, R. Leena Sri, N. Balaji. A Novel Approach for Performance and Security Enhancement during Live Migration. Indian Journal of Science and Technology. 2016; 9(4). 1-8.
18. T.Thiruvenkadam, P. Kamalakkannan. Energy efficient multi dimensional host load aware algorithm for virtual machine placement and optimization in Cloud environment. *Indian Journal of Science and Technology*. 2015; 8(17), 1-11.
19. R. Manjusha, R. Ramachandran. Secure authentication and access system for cloud computing auditing services using associated digital certificate. Indian Journal of Science and Technology. 2015; 8(S7), 220-227.
20. R. Nagaraj, Dr.V. Thiagarasu, B. Jeevithapriya. Optimization and Scalable Constrained Clustering Performances. Indian Journal of Innovations and Developments. 2015; 4(7), 1-7.

The Publication fee is defrayed by Indian Society for Education and Environment (iSee). www.iseeadyar.org

Cite this article as:

B.Pavithraselvi, B.Sandhya. Dynamic optimization of VM to Server mapping. *Indian Journal of Innovations and Developments.* 2016; 5(1), January.