*I.J. Modern Education and Computer Science*, **2015, 4, 61-66**
Published Online April 2015 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijmecs.2015.04.07

# A New Approach for Dynamic Virtual Machine Consolidation in Cloud Data Centers

**Esmail Asyabi**
School of Computer Engineering, Iran University of Science and Technology, Iran, Tehran
Email: e_asyabi@comp.iust.ac.ir

**Mohsen Sharifi**
School of Computer Engineering, Iran University of Science and Technology, Iran, Tehran
Email: msharifi@iust.ac.ir

***Abstract*—**Cloud computing environments have introduced a new model of computing by shifting the location of computing infrastructure to the Internet network to reduce the cost associated with the management of hardware and software resources. The Cloud model uses virtualization technology to effectively consolidate virtual machines (VMs) into physical machines (PMs) to improve the utilization of PMs. Studies however have shown that the average utilization of PMs in many Cloud data centers is still lower than expected. The Cloud model is expected to improve the existing level of utilization by employing new approaches of consolidation mechanisms. In this paper we propose a new approach for dynamic consolidation of VMs in order to maximize the utilization of PMs. This is achieved by a dynamic programing algorithm that selects the best VMs for migration from an overloaded PM, considering the migration overhead of a VM. Evaluation results demonstrate that our algorithms achieve good performance.

***Index Terms*—**Cloud Computing, Virtual Machine, Dynamic Consolidation, Migration.

## I. Introduction

Cloud data centers host a variety of applications such as Internet applications whose workloads continuously change. These kinds of applications are the true beneficiaries of the elasticity property offered by Cloud computing environments. Using elasticity, resources allocated to virtual machines (VMs) based on their application demands, can be dynamically scaled up or down. In fact, after uploading applications onto VMs, the Cloud service provider can properly allocate resources based on demands of applications on VMs. Therefore, users are only charged for what they actually use, reducing their cost significantly [1][2].

Data centers often provide resources for the peak demand so that they can make sure that sufficient resources are available; in addition, the performance of VMs applications are guaranteed .Needless to say that applications are not always in their peak demand; therefore, physical machines (PMs) are often underutilized since their resources are overprovisioned. Studies have found that the average utilization of PMs in many Cloud data centers is very low. Real world estimates range from 5% to 20%. Using dynamic and automatic VM consolidation, the Cloud model is expected to increase the overall utilization of physical resources in existing data centers [3][4][5].

Dynamic VM consolidation approaches leverage dynamic nature of Cloud model, both PMs and their VMs are periodically monitored. In order to minimize the number of active PMs and maximize the quality of delivered services, whenever a PM becomes a hot or cold spot, its VMs are reallocated using live VM migration. According to [6], dynamic VM consolidation problem is divided into the following four sub-problems:

1. Deciding what to do when a PM becomes over-loaded (hot spot); to avoid QOS degradation, some of its VMs should be migrated away.
2. Deciding what to do when a PM becomes under-loaded (cold spot); to save energy, all of its VMs must be migrated to other PMs so that the PM can be switched off.
3. Selecting the best VMs for migration from an overloaded PM.
4. Selecting the best destination PM for migrated VMs.

In this paper we focus on the last two sub-problems. We aim to select the best VMs for migration from an overloaded PM. To achieve this goal, we first introduce an unevenness formula. Using unevenness, memory size and granularity of VMs, we quantify the migration cost of each VM on an overloaded PM. We then propose a new dynamic programming algorithm for selecting the best VMs for migration from an overloaded PM. Finally, we present an algorithm for selecting the best destination for a VM that is a candidate for migration based on our unevenness formula.

## II. Related Work

In recent years, a lot of work has been done in the area of VM placement in Clouds. The goal has been to optimally exploit available resources, while avoiding VM

performance degradation. This problem has usually been formulated as a multi-dimensional bin packing problem. In this regard, several algorithms have been proposed with different objectives such as minimizing the number of running PMs [6][7][8][9]. Konstantinos et al. [8] use intelligent placement of VMs on PMs by employing user provided placement hints. Hints offer desired VM deployments for consumer workloads. Their framework, however, may ultimately ignore part or all of hints based on the overall available physical resources. They did not define any trust model. Also, the users might provide hints that are not compatible with the Cloud infrastructure characteristics.

Ofer Biran et al. [1] have focuses on network aware VM placement and proposed a new solution, Min Cut Ratio Aware VM Placement (MCRVMP). They consider not only the constraints of local resources such as CPU and memory but also the constraints of network resources.

Michael Cardosa et al. [10] have introduced VM placement algorithms to reduce the overall energy consumption in virtualized MapReduce clusters. Their algorithms co-place MapReduce VMs based on their complementary resource needs in order to fully utilize available resources. In addition, they have proposed algorithms for co-locating MapReduce VMs with similar runtime so that a PM could run at a high utilization throughout its uptime. In their approach, once all co-located VMs have finished, the Cloud operator can hibernate the PM to conserve energy. They have made some simplifying assumptions. First, they have assumed that the completion time of workloads can be estimated that does not necessarily hold in real world scenarios. They also use a small set of types of VMs. In our approach, there are no assumption about VM types and sizes.

Anton Beloglazov et al. [6] have presented interesting algorithms only for PM overload detection and proved their optimality. They allow system administrator to explicitly set QoS goals in terms of OTF parameter, which is a workload independent QoS. In contrast, we use a simple heuristic algorithm in our approach to detect overloaded PMs. In addition, we propose algorithms for selecting the best VMs from an overloaded PM for migration. Moreover, we select the best destination PM for migrated VMs.

Work in [4] presents a dynamic resource allocation system that strives to avoid the overload of PMs while minimizing the number of used PMs. In order to detect the potential overloaded PM, they periodically monitor the overall status of data center. They also introduce a load prediction algorithm that can capture the future resource usage of applications and decide how to place VMs based on this prediction. In a similar spirit, we continuously monitor the existing PMs to determine the overloaded PMs but we do not use the prediction approaches which may cause wrong placement.

## III. Migration Cost Of Vms

Current virtualization technology offers the ability to easily relocate a VM from one PM to another without shutting it down called VM live migration. It gives the opportunity to dynamically optimize the placement of VMs with small impact on their performance [9]. As mentioned, live migration opens opportunities for dynamic consolidation of VMs, nevertheless; it can introduce a significant overhead in the network infrastructure and deteriorate the performance of service delivered by cloud service provider.

As mentioned before, our main goal is to keep the utilization of existing PMs in the highest level possible. Moreover, the performance degradation of VMs must be in lowest level possible. To achieve these goals, we will quantify the migration cost of VMs so that we can measure migration overhead of them. To do that, we consider three criterions which have a significant impact on migration overhead. These criteria are listed in below.

a) Memory size: we assume all VMs are connected to a storage area network (SAN) and each VM image is stored on the SAN; hence, the cost of VM live migration is mostly determined by its memory footprint. Therefore, it can be said, migration time is approximately equal to the memory size of a VM divided by the network band width. As a result, memory size of VMs is a good measure for the cost of migration. Thus, in case there are options for migration, the VM which has the lowest memory footprint is the best.

b) In order to improve the overall utilization of PMs, it is essential to assign complementary workloads to a PM In other words, those VMs which their resource demands are complement, will be consolidated in the same PM. To achieve this goal, we introduce unevenness formula that quantifies how much the VMs that are consolidated in a PM are complement. Equation (1) calculates the unevenness of PM*p*.

$$unevenness(p) = \frac{1}{n}\sum_{i \neq j}\sqrt{(r_i - r_j)^2} \qquad (1)$$

where *n* is the number of resource types in PM*p* and $r_i$ is the overall resource usage for resource type *i* in PM*p*. Note, in the above calculation, we only consider bottleneck resource types such as processor, memory and network bandwidth. Actually the major design goal of our model is to keep the utilization of physical resources on each PM at the highest level while the unevenness of them is the lowest possible. In case there is a PM with a VM on it that candidate for migration, we select the VM whose unevenness of the PM can be reduced the most by migrating it.

c) Since we want to keep the utilization of PMs in the highest level, if we have to migrate a VM from the PM, a VM with lowest granularity is the best case. To achieve this goal, we calculate the granularity of a VM by (2). Where $c(v)$ is the VM CPU utilization, *m(v)* is its memory utilization and *n(v)* is its network utilization. Therefore in case there are multiple VMs

for migration, we select the one which has the lowest granularity.

$$Granularty(v) = c(v) * m(v) * n(v) \quad (2)$$

Finally, the migration cost of a VM is calculated by the (3).

$$Migration\ cost(v) = \alpha * memory\ size(v) + \beta * unevenss(v) + \delta * Granularty(v) \quad (3)$$

## IV. Selecting Best Vms For Migration

Overloaded PM directly influences the delivered QOS because when the resource capacity is completely utilized, it is high likely that the applications are experiencing resource shortage [6]. In order to detect the overloaded and under loaded PM we use hot and cold threshold for each resource type in a PM. Whenever the utilization of a resource reaches hot threshold, the PM is considered as a hot spot. This indicates that the PM is overloaded and some of its VMs should be migrated away. In the same way, if the utilization of all resources is below the cold threshold, the PM is a cold spot, this indicates the PM is underutilized and mostly idle; therefore, all of its VMs should be migrated somewhere else to turn it off and save energy.

We consider a data center infrastructure composed by n distinct physical machines. Each PM is characterized by its resources (i.e. CPU, memory and network bandwidth). Suppose the utilization of resource r in PM p, is above the hot threshold that specified for resource r. Let x be the difference between current utilization of resource r and the hot threshold. As said earlier, some VMs of PMp should be migrated and the aggregate utilization of migrated VMs must be greater than or equal to x.

One solution is to sort the VMs of PMp in descending migration cost (note that migration cost of VMs is calculated via (3)). Afterward for each VM in the sorted list, we see if its utilization from resource r is greater or equal to x; if so, it will be selected for migration; Otherwise, if we could not find the VM whose removal can reduce the utilization of PM p as much as x, we will migrate VMs respectively as long as the utilization of PM p for resource r becomes less than the hot threshold.

Although seemingly satisfactory, this solution sometimes may not be optimal. It is possible that the VM will be selected while there are some other VMs whose aggregate utilization is greater than x and the aggregate cost of them is less than the one which is selected by this solution. Instead of the aforementioned approach, we propose an optimal algorithm for this problem that works the best in all situations.

The challenge here is to find the subset of VMs in which their aggregate utilization of resource r is greater than x and the aggregate migration cost of them is minimal. The problem we want to solve is detailed as follows: suppose we have PM p that its utilization of resource r is greater than the hot threshold specified for resource r, the difference between current utilization of resource r and the hot threshold is x, the migration cost of VM i is ci, the resource usage of VM i is ui and the number of VMs running in PM p is n where vi=1 if VM i is selected for migration, 0 otherwise. In the following the problem is summarized into formulation.

$$\begin{cases} v_i \quad 1\ if\ VM\ i\ is\ selected\ ,0\ otherwise \\ \sum_{i=0}^{n} v_i u_i \geq x \\ \sum_{i=0}^{n} v_i c_i \quad must\ be\ minmal \end{cases} \quad (4)$$

We present a dynamic programing algorithm to solve this problem. Tow-dimensional matrix, *M [0..n, 0..x]*, is used to hold migration costs. Where *M [i, j]* is the minimum migration cost when there are VMs from *1* to *i* (the VMs are numbered from *1* to *n*) such that the aggregate utilization of selected VMs is greater than or equal to *j*. We need to calculate *c[n, x]* to find the result. The optimal migration cost can be recursively calculated using fallowing formula. (Note that the columns of the matrix corresponding to the unit of utilization can be changed from *1* to *x*, and rows corresponding to a virtual machine, can be changed from 1 to n).

$$\boldsymbol{M}[i,j] = \begin{cases} \infty & i = 0\ or\ j = 0 \\ min\{M[i-1,j]\ , ci + M[i-1, j-u_i]\} & j > u_i \\ min\{M[i-1,j]\ ,\ ci\} & j \leq u_i \end{cases} \quad (5)$$

We use an auxiliary array *Items [1..n]*, to store the selected VMs. At the end of the algorithm run, Items will hold the subset of VMs which selected for migration. Time complexity of our algorithm is *O(x.n)*. Where n is the number of VMs running on PM p and x is the difference between current utilization of resource r and the PM hot threshold. The algorithm is shown in below.

*Algorithm1: SelectTheBestVmsForMigration(PM p, p[1..n] VMs Costs , U[1..n] VMs utilization of resource r, the_overload)*

```
Let c[0 ..n][0.. the overload] be a empty matrix
Let Items[1.. the overload] be a empty array of list type
For u ← 0 to the_overload steps the_overload /10
        c[0][u] ←∞
        End for
    For i ← 1 to n do
c[i][0] ←∞
For u ←1 to the_overload
    If (U[i]<u)
        If (c[i-1,u] ≤ P[i]+c[i-1,u-U[i]])
            c[i,u]=c[i-1,u]
        End if
        Else
            c[i,u] =p[i]+ c[i-1, u-U[i]]
            items[u] ← list of items [u] + item i
        End else
    End if
                                Else
        If c[i-1,u] ≤ P[i]
            C[I,u]=c[i-1,u]
        End if
        Else
            C[I,u] =P[i]
            items[u] ← item i
        End else
    End else
  End for
End for
Return Items[the_overload]
```

## V. Allocating VMs To PMs

In this section we present an algorithm for VM placement. VM placement is the topic of many recently researches [16][19]. When a VM is selected for migration or is initially placed on a PM, we must select the best destination based on VM characteristics and overall policy of the data center.

Data centers follow different consolidation polices. For example, a data center may consolidate VMs on the server with high level of resources utilization to save energy. However, this policy may lead to QoS violations since VMs may not have access to the sufficient resources. On the other hand, a data center may consolidate VMs on servers with low level of resources utilization to make sure that the VMs have access to the sufficient resources and delivered services do not violate the predefined service level agreement.

As mentioned before, we follow energy efficient policy which means if there are some PMs as a destination for a migrated VM, we choose the one which has high level of utilization to reduce the running PMs and save the overall energy consumption of the data center. We also consolidate VMs in the PMs which have sufficient resources so that we can make sure of QoS of delivered services.

As said before, The destination PM must obviously have sufficient resources. Among all such severs, we select the one which its unevenness reduced the most by accepting the VM. (Recall that we calculated the unevenness of the PM in section 2 based on (1)). We also support green cloud computing by putting idle machine into standby or low power mode. Time complexity of our algorithm is *O(n)*. Where *n* is the number of PMs that exist in the data center. The algorithm is shown in below.

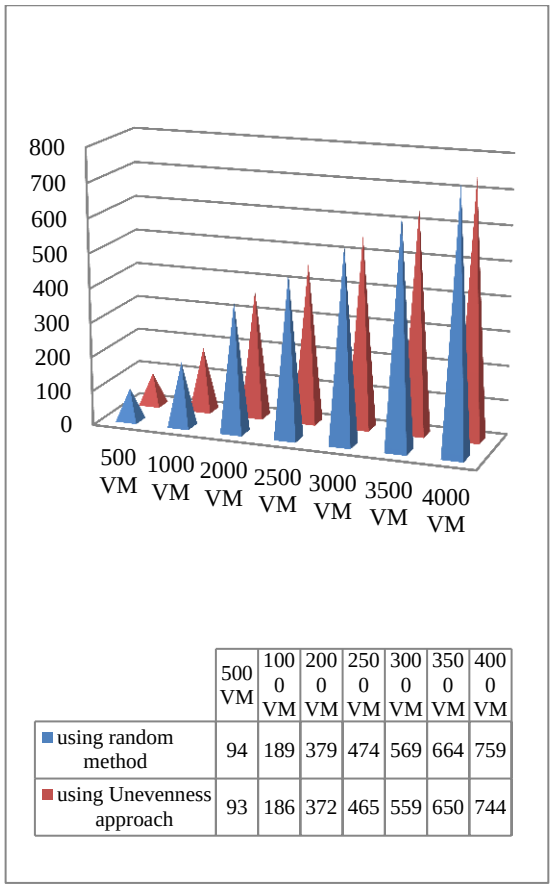
*Algorithm2: BestPM_ForMigration(VM v,list of PMs)*

```
Max=0;
Selected_PM ← NULL
Foreach PM p in list of PMs
    If( p has adequate resources for v)
        Current_unevenness=compute  unevenness of p
        Let new_unevenness be an unevenness when v is added to p
        If(new_unevenness- current unevenness ≥max)
            Max ← new unevenness- current unevenness
            Selected_PM ← p
        End if
    End if
End for
If(Selected_PM is NULL)
    Selected_PM ←Turn on new physical machine and add v to it
                End if
            Return Selected_PM
```

## VI. Evaluation

We first evaluate the effectiveness of our placement algorithm. We must place VMs on the data center PMs in which the VMs meet their resource needs while the number of used PMs is minimum possible. We use our unevenness formula to place VMs on the PMs. In this way, if there are multiple PMs that have sufficient resources for a VM, We select the one which its unevenness reduced the most by accepting the VM.

In this experiment we first created a number of VMs with different sizes and then placed them onto the PMs. Then we increased the number of VMs from 500 to 4000 and repeated the experiment. The number of used PMs every time was kept. In the next step we used random VM to PM mapping. In this case if a PM has sufficient resources for a VM is selected as a destination PM. We did the previous experiment with random method. The results of these experiments are shown in Fig. 1, As it can be seen, our method in comparison with random method decreases the final active PMs.



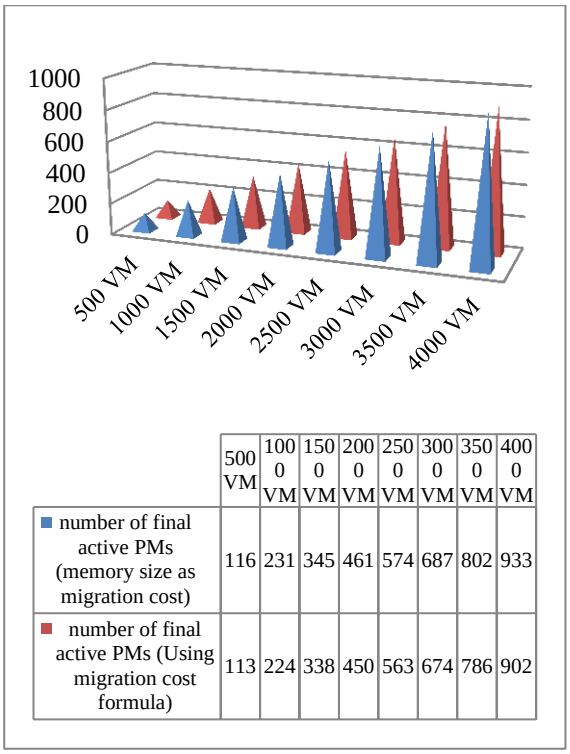

| | 500 VM | 1000 VM | 2000 VM | 2500 VM | 3000 VM | 3500 VM | 4000 VM |
|---|---|---|---|---|---|---|---|
| using random method | 94 | 189 | 379 | 474 | 569 | 664 | 759 |
| using Unevenness approach | 93 | 186 | 372 | 465 | 559 | 650 | 744 |

Fig. 1. Number of final active VMs using unevenness formula.

Next we evaluate the effectiveness of our approach when a PM is overloaded. Recall that our scheduler is invoked periodically, and whenever a PM is overloaded, some of its VMs should be migrated away. In this experiment we first placed a certain number of VMs on the PM according to our unevenness formula. Afterword we gradually increased the CPU demand of the half of existing VMs to emulate a situations such as flash crowd in the internet applications. The CPU demand of VMs was increased by 10%. Typically some PMs became hot spot. Consequently; some of their VMs must have migrated somewhere else. At first, we used memory size as the migration cost for VMs. later our migration cost formula was used to calculate the migration cost for each VM.

In this experiment we defined 80% as the hot threshold for the CPU resource type. Similar to previous experiment, we increased the number of VMs form 500 to 4000. Each time after demand increasing, the number of active PMs was kept. The results of the experiment are shown in Fig. 2, as mentioned, we did the previous experiment with the same VMs. but, we only considered the memory size of VMs as a migration cost. As it can be seen from Fig. 2, our approach reduces the number of final active PMs.




| | 500 VM | 1000 VM | 1500 VM | 2000 VM | 2500 VM | 3000 VM | 3500 VM | 4000 VM |
|---|---|---|---|---|---|---|---|---|
| number of final active PMs (memory size as migration cost) | 116 | 231 | 345 | 461 | 574 | 687 | 802 | 933 |
| number of final active PMs (Using migration cost formula) | 113 | 224 | 338 | 450 | 563 | 674 | 786 | 902 |

Fig. 2. Number of final active VMs using migration cost formula.

In the next experiment we used our dynamic programing algorithm to evaluate its effectiveness. As said before, when a PM is overloaded, some of its VMs must be migrated to other PMs to avoid QoS degradation.

In the previous experiment, we use our migration cost formula to select candidate VMs for migration. In this experiment we use our dynamic programing algorithm to select candidate VMs. The experiment is detailed in below.

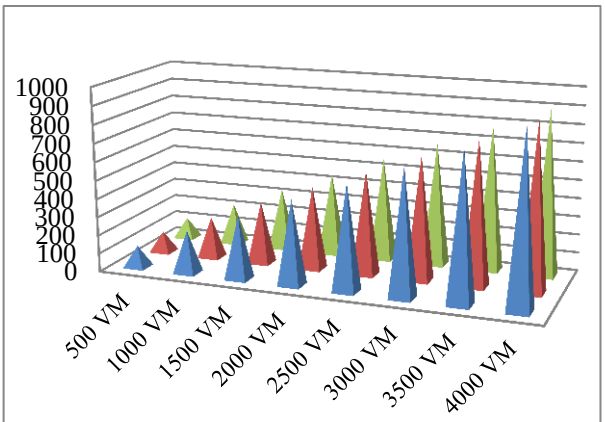
We first placed a certain number of VMs on the PMs and then increased the CPU demand of half of existing VMs. Like other experiment we increased the CPU demands of VMs by 10%. As expected some of the existing PMs became hot spot. Therefore some of their VMs should be migrated. Note that we defined 80% as the hot threshold which means if the CPU utilization of a server is more than 80%, the server is considered as a hot spot.

We use our dynamic programing algorithm presented in section 2 to select the best VMs for migration. We also use the migration cost formula to calculate migration cost of VMs in an overloaded PM. Similar to the previous experiment, we increased the number of VMs form 500 to 4000. Each time after demand increasing, the number of active PMs was kept

As expected we could improve the earlier results. Finally, the number of active PMs was less than the number of active PMs in the previous employed methods. The results are shown in Fig. 3. As it can be seen from Fig. 3, we decrease the average final active PMs of data center by 2.3%. This also indicates that the overall energy consumption of the data center is reduced by 2.3%.

## VII. Conclusion

In this paper, we presented a new approach for the problem of dynamic VM consolidation of virtual machines in Cloud computing environments. We first defined the migration cost of each VM based on their memory, CPU and network utilization. Afterward we presented a dynamic programming algorithm to select the best VM for migration based on VM migration cost from an overloaded PM. Moreover, we presented an algorithm to select the best destination for migrated VMs. Simulation results demonstrate we achieve good performance and significantly reduce the overall energy consumption of the cloud data center. The research work is planned to be followed by development of our algorithms in support of elasticity property of Cloud Computing Environments.




| | 500 VM | 1000 VM | 1500 VM | 2000 VM | 2500 VM | 3000 VM | 3500 VM | 4000 VM |
|---|---|---|---|---|---|---|---|---|
| number of final active PMs (memory size as migration cost) | 116 | 231 | 345 | 461 | 574 | 687 | 802 | 933 |
| number of final active PMs (Using migration cost formula) | 113 | 224 | 338 | 450 | 563 | 674 | 786 | 902 |
| number of final active PMs (Using Dynamic programing algorithm) | 112 | 223 | 337 | 449 | 560 | 672 | 784 | 898 |

Fig. 3. Number of final active VMs using unevenness dynamic programming algorithm.

## VIII. References


[1] Ofer Biran, Antonio Corradi, Mario Fanelli, Luca Foschini, Alexander Nus, Danny Raz and Ezra Silvera, "A Stable Network-Aware VM Placement for Cloud Systems", 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2012.

[2] Daniel Warneke and Odej Kao, "Exploiting Dynamic Resource Allocation for Efficient Parallel Data Processing in the Cloud," IEEE Transactions on Parallel and Distributed Systems, vol. 22, no. 6, pp. 985-997, June 2011.

[3] Zhen Xiao, Qi Chen and Haipeng Luo, "Automatic Scaling of Internet Applications for Cloud Computing Services," IEEE Transactions on Computers, 30 Nov. 2012.

[4] Zhen Xiao, Weijia Song and Qi Chen, "Dynamic Resource Allocation Using Virtual Machines for Cloud Computing Environment," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 6, pp. 1107-1117, June 2013.

[5] Shekhar Srikantaiah, Aman Kansal and Feng Zhao, "Energy Aware Consolidation for Cloud Computing", Proceedings of the 2008 conference on Power aware computing and systems, p.10-10, December 07, 2008, San Diego, California.

[6] Anton Beloglazov, Rajkumar Buyya, "Managing Overloaded PMs for Dynamic Consolidation of Virtual Machines in Cloud Data Centers under Quality of Service Constraints," IEEE Transactions on Parallel and Distributed Systems, vol. 24, no. 7, pp. 1366-1379, July 2013.

[7] Wenting Wang, Haopeng Chen and Xi Chen, "An Availability-aware Approach to Resource Placement of Dynamic Scaling in Clouds", IEEE Fifth International Conference on Cloud Computing, 2012, Honolulu, USA.

[8] Konstantinos Tsakalozos, Mema Roussopoulos, and Alex Delis, "Hint-based Execution of Workloads in Clouds with Nefeli" , ieee transactions on parallel and distributed systems, 2012.

[9] Nicolo Maria Calcavecchia, Ofer Biran, Erez Hadad and Yosef Moatti, "VM Placement Strategies for Cloud Scenarios," cloud, pp.852-859, 2012 IEEE Fifth International Conference on Cloud Computing, 2012.

[10] Michael Cardosa, Aameek Singh, Himabindu Pucha, Abhishek Chandra, "Exploiting Spatio-Temporal Tradeoffs for Energy-Aware MapReduce in the Cloud," IEEE Transactions on Computers, vol. 61, no. 12, pp. 1737-1751, Dec. 2012.

[11] Fan Chung, Ronald Graham, Ranjita Bhagwan, Stefan Savage and Geoffrey M. Voelker, "Maximizing Data Locality in Distributed Systems", Journal of Computer and System Sciences archive, Volume 72 Issue 8, December, 2006.

[12] Anton Beloglazov, and Rajkumar Buyya, "Optimal Online Deterministic Algorithms and Adaptive Heuristics for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Data Centers", journal of Concurrency and Computation: Practice and Experience, Volume 24, Issue 13, pages 1397–1420, 10 September 2012.

[13] Sherif Sakr, Anna Liu, Daniel M. Batista and Mohammad Alomari, "A Survey of Large Scale Data Management Approaches in Cloud Environments", IEEE Journal of Communications Surveys & Tutorials, Vol. 13, pp. 311 – 336, 2011.

[14] Sivadon Chaisiri, Bu-Sung Lee, Dusit Niyato, "Optimization of Resource Provisioning Cost in Cloud Computing," IEEE Transactions on Services Computing, vol. 5, no. 2, pp. 164-177, Second 2012.

[15] Cui Lin and Shiyong Lu, "Scheduling Scientific Workflows Elastically for Cloud Computing", IEEE 4th International Conference on Cloud Computing, 2011, Washington, DC, USA.

[16] Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel, "Elastic Virtual Machine for Fine-grained Cloud Resource Provisioning", Springer 4th International Conference on Computing and Communication Systems, 2012.

[17] Ui Han, Li Guo, Moustafa M. Ghanem and Yike Guo, "Lightweight Resource Scaling for Cloud Applications," IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012), 2012.

[18] Bahman Javadi, Jemal H. Abawajy and Rajkumar Buyya, "Failure-aware resource provisioning for hybrid Cloud infrastructure", Journal of Parallel and Distributed Computing archive, Volume 72 Issue 10, October, 2012.

[19] Marco Guazzone, Cosimo Anglano, and Massimo Canonico, "Exploiting VM Migration for the Automated Powerand Performance Management of Green Cloud Computing Systems", springer *1st International Workshop on Energy-Efficient Data Centres (E2DC 2012)*, Madrid, Spain, May 2012.

[20] Hong Xu and Baochun Li, "Anchor A Versatile and Efficient Framework for Resource Management in the Cloud", IEEE Transactions on Parallel and Distributed Systems, 2012.

[21] Anton Beloglazov, Jemal Abawajy and Rajkumar Buyya, "Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing", Future Generation Computer Systems, 2012.


## Authors' Profiles

**Esmail Asyabi** is a Phd student in software engineering in the School of Computer Engineering of Iran University of Science and Technology.
He is mainly interested in cloud computing environments and distributed systems. He also received MSc, in software engineering from Iran University of Science and Technology. His website is http://webpages.iust.ac.ir/e_asyabi

**Mohsen Sharifi** is a professor of software engineering in the School of Computer Engineering of Iran University of Science and Technology. He directs a distributed systems research group and laboratory. He is main ly interested in the engineering of distributed systems, solutions, and applications, particularly for use in various fields of science. The development of a true distributed operating system is on top of his wish list. He received his BSc, MSc, and PhD in co mputer science fro m Victoria University, Manchester, UK, in 1982, 1986, and 1990, respectively. His website is http://webpages.iust.ac.ir/msharifi.