# Achieving Intelligent Traffic-aware Consolidation of Virtual Machines in a Data Center Using Learning Automata

Akaki Jobava
Oslo and Akershus University College of Applied Sciences
Departement of Computer Science
Oslo, Norway

Anis Yazidi
Oslo and Akershus University College
Oslo, Norway
Departement of Computer Science
Email: anis.yazidi@hioa.no

B. John Oommen
Carleton University
School of Computer Science
Ottawa, Canada
Email: oommen@scs.carleton.ca

Kyrre Begnum
Oslo and Akershus University College
Oslo, Norway
Departement of Computer Science
Email: Kyrre.Begnum@hioa.no

***Abstract*—Cloud Computing (CC) is becoming increasingly pertinent and popular. A natural consequence of this is that many modern-day data centers experience very high *internal* traffic within the data centers themselves. The VMs with high mutual traffic often end up being far apart in the data center network, forcing them to communicate over unnecessarily long distances. The consequent traffic bottlenecks negatively affect both the performance of the application and the network in its entirety, posing non-trivial challenges for the administrators of these cloud-based data centers. The problem can, quite naturally, be compartmentalized into two phases which follow each other. First of all, the VMs are consolidated with a VM clustering algorithm, and this is achieved by utilizing the toolbox involving Learning Automata (LA). By mapping the clustering problem onto the Graph Partitioning (GP) problem, our LA-based solution successfully reduces the total communication cost by amounts that range between 34% to 85%. Thereafter, the resulting clusters are assigned to the server racks using a cluster placement algorithm that involves a completely different intelligent strategy, i.e., one that invokes Simulated Annealing (SA). This phase further reduces the total cost of communication by amounts that range between 89% to 99%. The analysis and results for different models and topologies demonstrate that the optimization is done in a fast and computationally-efficient way. Indeed, as far as we know, this paper pioneers the application of LA in the traffic-aware consolidation of virtual machines in data centers, and also pioneers a strategy which serializes the tools in LA and SA to optimize CC.**

## I. Introduction

Cloud Computing (CC) is a relatively new phenomenon. It refers to an environment and computational model in which physical and virtualized computing resources are distributed and accessed over the network. CC is maturing to become a very central paradigm within the theory and applications of computation. Its robustness, increasing user-friendliness, high flexibility and scalability, combined with its cost efficiency [3], [9], [12], make it an increasingly popular model in real-life enterprises.

One of the main reasons behind the success of CC is that the concept of "virtualization", central to this computational model, allows the overall system to create, clone, migrate, restore, etc. Virtual Machines (VMs) in a time-effective manner with minimal effort from the system administrator. Live migration allows VMs to be moved from one physical host to another without the client/customer noticing it. Consequently, CC is becoming one of the major driving forces behind the rapid growth of data centers around the world [7]. The goal of this paper is to see how the VMs can be optimally placed within a data center in a traffic-aware manner. Viewed from a traffic-aware perspective, the resource of bandwidth becomes a bottleneck in the higher layers of the network, decreasing the performance when it concerns communication [17] between the applications. This also increases the workload for the network elements on the aggregation and core layers, which, in turn, often results in higher power consumption within the data center [7], more greenhouse emissions, and the increased business costs. First of all, it is clear that, in most cases, the applications communicating extensively with each other in the cloud environment will belong to the same tenant. It would thus be beneficial for the whole network if the VMs hosting applications with high mutual traffic were deployed in the close proximity with each other. To accomplish this, we would like the VMs that communicate much with each other to be "clustered" together. Such a placement would relieve the network elements in the upper layers

Third author status: *Chancellor's Professor*, *Fellow: IEEE* and *Fellow: IAPR*. The author is also an Adjunct Professor with University of Agder, Grimstad, Norway.

978-1-5090-2914-3/16/$31.00 ©2016 IEEE

of the networking infrastructure where the most expensive equipment usually operate, and fully utilize the links at the lower levels of the network. However, this, in and of itself, is far from trivial because the traffic patterns are not known *a priori*. Secondly, once we have identified the VMs that really should be in the close proximity of each other, the task is to assign them to the available server racks. This paper addresses both these issues. Firstly, it investigates how the VMs with high mutual communication can be consolidated into clusters in order to reduce the total communication cost. It then explains how these clusters can be assigned to the racks.

One approach to resolve this problem could be to attempt all possible combinations of VM placements and choose the most optimal configuration. However, since data centers usually host hundreds/thousands of VMs, this would require us to test an astronomical number of different permutations in order to find the best possible placement when the number of VMs is greater than 20 – the task would be computationally infeasible. The *modus operandus* suggested in this paper breaks down the problem in two main parts - each associated with one of the above distinct phases of solving the problem. We first determine the VM clusters using a Graph Partitioning (GP) algorithm. This is achieved by utilizing the toolbox involving Learning Automata (LA). By mapping the clustering problem onto the GP problem, our LA-based solution successfully reduces the total communication cost since it succeeds in consolidating VMs with high mutual traffic into distinct clusters. We then address the second phase of assigning the resulting clusters to the physical hosts in the server racks in the data center. This problem is not as computationally hard as the previous phase as any algorithm that resolves the quadratic assignment problem should be able to handle it. We have opted to solve this phase by invoking the tools in the toolbox of Simulated Annealing (SA).

## II. Three-tier network architecture

A data center network is traditionally based on a *layered* [16] or a three-tier approach. Such a three-tier network architecture consists of three layers of switches and routers (see Fig.1). The layered approach is designed to enhance scalability, high performance and flexibility and to also improve the maintenance associated with data center networks. These layers are explained below.

Access layer: This is where the servers are physically connected to the network by connections to the Layer 2 switches, also called the Access or Edge switches.

Aggregation layer: This layer provides functions such as service module integration, Layer 2 domain definitions, spanning tree and default gateway redundancy.

Core layer: This layer handles all the incoming and outgoing traffic that comes in and leaves the data center. This layer provides the connectivity required to various aggregation modules. It handles the Layer 3 networking with the access and border routers.

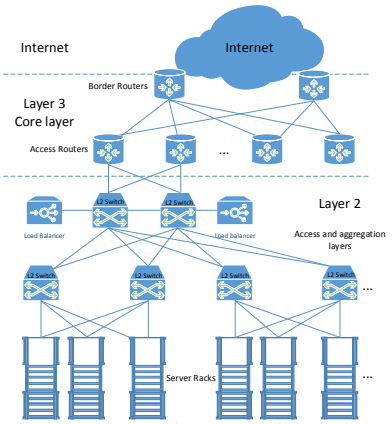

Fig. 1: The architecture of a traditional layered data center.

## III. Data Center Network Architectures

Due to the exponential growth of the cloud in data centers and the evolution of the computers in an of themselves, computing power is no longer the constraining factor in the data centers. The servers are becoming increasingly powerful and as the phenomenon of CC grows, the number of VMs correspondingly explodes. Thus, data centers are faced with inherent problems in the traditional data center network (DCN) architecture. This leads to real problematic issues such as bandwidth bottlenecks, oversubscription in the higher layers and the under-utilization of the lower layers of the data center network are becoming [1]. To resolve this, several new approaches to designing data center network topologies have been proposed in the recent years, one of which is the "tree topology" discussed below.

**A tree topology**: As mentioned previously, modern-day data centers usually follow traditional three-tier (or three-layer) network architectures. At the lowest level, referred to as the *access tier*, hosts connect to one or multiple access switches. Each of the access switches is connected to one or multiple aggregate switches at the aggregation layer. The aggregation switches, in turn, connect to multiple core switches at the core layer. This design creates a tree-like topology where packets are forwarded according to a Layer 2 logical topology [13]. The higher level network elements are usually enterprise-level devices and are often highly oversubscribed.

### A. Cost matrix

A cost matrix (or a distance matrix) is a two-dimensional array which contains information about the communication cost (or the distance) between the pairs of nodes in a set of nodes. The matrix usually has a size of $N \times N$, where $N$ is the number of the nodes in the set of nodes. Each row in the matrix corresponds to a single node denoted by $i$ and each column also represents a single node, denoted by $j$.

$$C_{ij} = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,N} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,N} \\ \vdots & \vdots & \ddots & \vdots \\ c_{N,1} & c_{N,2} & \cdots & c_{N,N} \end{bmatrix} \quad (1)$$

In the example matrix displayed above, each element of the matrix represents cost of communication from node $i$ to node $j$, or quantifies the "distance" from node $i$ to node $j$.

## IV. Proposed VM Clustering Algorithm

We shall now explain the strategy that we use to resolve the assignment of VMs. The reader must first of all appreciate that the assignment of VMs is essentially a clustering exercise. Indeed, since the traffic patterns are not known *a priori*, the assignment algorithm must learn the best assignment by inferring this from the real-time traffic. In other words, the VMs that communicate much with each other must be in the close proximity of each other, while those that communicate less could be, potentially, placed further apart. With a little insight, one can see that this is precisely equivalent to the problem of partitioning the nodes of a graph into subsets based on some pre-defined similarity criteria. This is exactly the paradigm that we invoke. Our proposed VM clustering algorithm is based on Oommen's Graph Partitioning Using Learning Automata (GPLA) [15] algorithm. That being said, the GPLA, in and of itself, is not directly applicable to our application domain. Rather, we shall see that it has to be adapted to resolve VM assignment. The GPLA attempts to solve the Graph Partitioning Problem (GPP) [2], [5], [8] by using the toolbox that incorporate stochastic Learning Automata (LA), which learn the optimal action offered by a random environment. Learning is achieved by interacting with the environment as it constantly changes and by processing the response of the environment to the actions taken. In this paper we deal with a version of the GPP in which all the sub-partitions are of equal size, and this is precisely the so-called Equi-Partitioning Problem (EPP). The best solution to the EPP is the so-called Object Migrating Automaton (OMA) proposed by Oommen and Ma [14]. This technique will be adapted for the GPP and used in the proposed VM clustering algorithm. As we will explain later in the section explaining the experimental results, the algorithm adapted for this work will read the set of 1,600 nodes or vertices distributed over 16 sub-partitions, also referred to as groups or arms, and deliver as its output the final solution of the corresponding graph partitioning problem. This will be achieved by adopting the OMA used in Oommen's algorithm. The strategy will involve checking pairs of vertices that are randomly selected by the algorithm in order to determine whether they are connected "significantly", based on which they will be either rewarded or penalized depending on the corresponding conditions ofconnectivity. In order to determine whether the nodes are connected "significantly", we specify two important thresholds, *SimilarityThreshold* and *DissimilarityThreshold*, calculated by the following formulae:

$$SimilarityThreshold = (1 + \rho) MeanEdge$$
$$DisimilarityThreshold = (1 - \rho) MeanEdge$$

where $\rho$ will be set to the fixed value of 0.25 and the *MeanEdge* value will be calculated by computing the average edge value based on all the nonzero elements (or edges between the nodes) of the symmetric VM traffic matrix $D$.

The pseudo-code for the partitioning [14] is given in Algorithm 1. For more details, we refer the reader to [14].

When two random vertices $V_i$ and $V_j$ are picked and their corresponding edge $D_{ij}$ is higher than the *SimilarityThreshold* the two nodes will be regarded as *similar*. If the nodes are found to be in distinct sub-partitions they will be penalized since this state is unfavorable. If, however, the nodes are found in the same sub-partitions they will be rewarded since this scenario is favorable. The penalize action will move the nodes closer to the *MinimumCertainty* state towards the outer boundary of the sub-partition while the reward action will push the nodes deeper into their sub-partitions, i.e., towards the *MaximumCertainty* state. When the nodes reach the outer boundaries of their sub-partitions they could be made to migrate from their current sub-partitions and moved to a better one. This process will be repeated until the maximum number of iterations is reached.

## V. Enhancement of the OMA algorithm for solving our problem

Our primary objective is to assign the VM clusters to the server racks in a manner that decreases the total cost of communication. *This* assignment problem will be treated as a Quadratic Assignment Problem (QAP) [11], [13], known to be one of the most difficult combinatorial optimization problems. The assignment of the 16 clusters to the available 16 server racks that gives the lowest total communication cost will be considered as the best assignment. The task of the cluster placement algorithm will be to conduct a search of the best assignment in the possible solution space. Since the solution space for 16 groups is an astronomically large number (i.e., 16!) the exhaustive search approach in order to find the best solution is computationally infeasible. Instead, we seek a solution that is "most optimal" from among a specific pool of solutions. In order to find such an optimal solution to QAP, we will invoke a simulated annealing (SA) phase [4], [10]. SA ensures that the algorithm does not get trapped in a local minimum and that it will be given a chance to explore a wider range of possible solutions by visiting even the inferior solutions with constantly decreasing probability [6].

**Input:** The set $V = \{v_1, v_2, ..., v_{KN}\}$ to be partitioned into K sub-partitions.
D is adjacency traffic matrix and $V_1$, $V_2$ ... $V_K$ are current feasible sub-partitions.
$\rho$ is a parameter used to determine the similarity or dissimilarity of the vertices.
M=100.
**Output:** The final partitions $\{V_1, V_2, ..., V_K\}$
**Preprocess:**
Compute Mean_Edge. Randomly partition V into $\{V_1, V_2, ..., V_K\}$
Assign all nodes to the boundary state of the actions
**Data**: Set of nodes to be partitioned: $V = \{v_1, v_2, ..., v_{KN}\}$
**Result**: The final solution to the GPP
**Method**:

```
for Iteration :=1 to Max_Iterations do
    for a random edge E_ij do
        if C_ij > (1 + ρ) · Mean_Edge then
            if v_i and v_j are in same sub-partition then
                RewardSimilarNodes(i,j)
            end
            else
                PenalizeSimilarNodes(i,j)
            end
        end
        else
            if C_ij < (1 − ρ) · Mean_Edge then
                if v_i and v_j are in same sub-partition then
                    PenalizeDissimilarNodes(i,j)
                end
            end
            else
                Pass
            end
        end
    end
end
```

**Algorithm 1: The Pseudocode for the Function ClusterVMs**

*1) Setting the Initial Cluster Placements:* The cluster placement algorithm will read the set of nodes previously partitioned by the VM clustering algorithm and the VM cluster traffic matrix $S$ in order to check all the possible cluster pairs and sort them by the corresponding edge values $\{S_{ij}\}$ in the descending order. Subsequently, the total cost of communication will be calculated using the VM cluster traffic matrix $S$ and the communication cost matrix $C$. The result of this step will be set as the initial and the current best states of the VM clusters. Observe that the initial placement will be an already-improved placement when compared to randomly-aligned VM clusters, and this helps the cluster placement algorithm to find an even more superior solution. In this regard. the total cost of communication will be calculated by summing all the edges multiplied by their corresponding communication costs using the following formula:

$$Comm_{Total} = \sum_{i,j=\cdots,n} D_{ij} \cdot C_{\pi(i)\pi(j)}, \qquad (2)$$

where $D_{ij}$ denotes a traffic rate between nodes $V_i$ and $V_j$, and $C_{\pi(i)\pi(j)}$ denotes the cost of communication between the server racks that the nodes $V_i$ and $V_j$ are assigned to.

## VI. The Simulated Annealing Process

Once the initial placement has been established and the initial total cost of communication has been calculated the algorithm will start executing the $N$ number of iterations by starting at a predefined value $T$ (temperature) and decreasing the temperature gradually. During each iteration two distinct clusters will be chosen and they will swap with places. After each swap the total cost of communication will be calculated and the new state will be stored temporarily. If the new state yields total cost of communication which is superior to the previous (or the initial) total cost of communication the algorithm will set is as the current best state. If the new state is inferior to the previous state the algorithm will move to it with a certain probability, P, calculated as below $P = e^{-\frac{\Delta}{T}}$, where $\Delta = TotalCost_{new} - TotalCost_{old}$, is the difference between the total communication cost yielded by the new state and the total communication cost of the old state, and $T$ is the temperature. This process will ensure that the algorithm does not get stuck in the local minimum and falsely assume that the optimal result has been obtained. Initially, the probability $P$ will have a higher value implying that the algorithm will accept inferior results more frequently. However, as the temperature $T$ decreases over time, the value of $P$ will gradually decrease and the algorithm will be less and less likely to accept inferior results. The simulated annealing technique will render to the cluster placement algorithm the potential of exploring a wider range of the possible solutions space. Ultimately, it will yield the most superior solution encountered.

## VII. Experiment Settings

In order to test the proposed algorithms on various *kinds* of data sets and to be able to retrieve reliable results, we performed two sets of experiments. They were conducted with two different sets of 1,600 VMs selected from the obtained traffic traces. We also understood the importance of having a plan by which one could perform the measurement and evaluation of the experimental results. We conducted three experiments on each of the simulated data center networking architectures. In each case, we conducted a separate experiment in order to observe changes in the intracluster and the intercluster

traffic caused by the VM clustering algorithm with the use of graph partitioning. In the experiments titled "Set A", we randomly selected the set of 1,600 VMs from the collected traffic traces, with the expectation that this set of 1,600 VMs will contain several VMs who have rather high mutual traffic while most of the VMs communicate with each other at a significantly lower rate. As shown in the previous sections, due to the VM clustering algorithm consolidating VMs with high mutual traffic in the same clusters, the intracluster communication increased by 1,369.28% while the intercluster traffic decreased by 84.92% at the same time. These changes caused the decrease of the total communication cost by 97.17% in the Tree set-up, by 96.82% in the Fat-tree set-up, and by 97.02% in the VL2 set-up. The smart assignment of the clusters to the server racks with the use of the simulated annealing implemented in the cluster placement algorithm further decreased the total communication cost by 99.58% in Tree set-up, by 99.52% in the Fat-tree set-up, and by 99.56% in the VL2 data center network architecture models. Figure 2 illustrates the total cost of communication with randomly assigned VMs, after the VM clustering phase and after cluster placement in all three data center network architecture models.

| | mean | st.dev | $\Delta_{Prev.mean}$ | $\Delta_{Overall}$ |
|---|---|---|---|---|
| $T_{RandTreeB}$ | 1601453698.57 | 18116631.36 | — | — |
| $T_{GpTreeB}$ | 1061026520.0 | 12608363.74 | -33.75% | -33.75% |
| $T_{QapTreeB}$ | 24244865.90 | 373481.77 | -97.71% | -98.49% |
| $T_{RandFtreeB}$ | 1950752236.0 | 17762337.15 | — | — |
| $T_{GpFtreeB}$ | 1288877475.49 | 12621967.57 | -33.93% | -33.93% |
| $T_{QapFtreeB}$ | 30825934.92 | 361225.03 | -97.61% | -98.42% |
| $T_{RandVl2B}$ | 1835909748.69 | 22100449.20 | — | — |
| $T_{GpVl2B}$ | 1211796514.7 | 10158873.22 | -33.99% | -33.99% |
| $T_{QapVl2B}$ | 28061769.69 | 410242.89 | -97.68% | -98.47% |

TABLE I: Changes in the total cost of communication for the various set-ups in the case of the data in Set B.

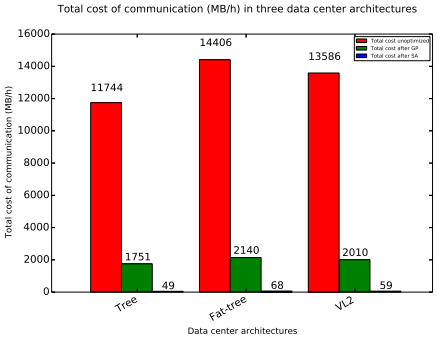


Fig. 2: Total cost of communication in all three experiments in Set A.

## VIII. Conclusion

The aim of this paper was to demonstrate how a Learning Automaton-based Graph Partitioning (GP) algorithm could be used to consolidate VMs in a traffic-aware manner, and to also show how a subsequent solution to a quadratic assignment algorithm could help in assigning the produced VM clusters to the server racks in order to reduce the total communication cost in a data center.

The analysis showed that the VM clustering algorithm was fast, resource-effective and extremely capable of consolidating the VMs with high mutual traffic in clusters while the cluster placement algorithm managed to find a significantly improved placement for the resulting clusters in all the data center network topologies tested.

## References


[1] K. Bilal, S. U. R. Malik, O. Khalid, A. Hameed, E. Alvarez, V. Wijaysekara, R. Irfan, S. Shrestha, D. Dwivedy, M. Ali, et al. A taxonomy and survey on green data center networks. *Future Generation Computer Systems*, 36:189–208, 2014.

[2] A. Buluç, H. Meyerhenke, I. Safro, P. Sanders, and C. Schulz. Recent advances in graph partitioning. *Preprint*, 2013.

[3] I. I. Center. Cloud Computing Research for IT Strategic Planning. http://www.intel.com/content/www/us/en/cloud-computing/next-generation-cloud-networking-storage-peer-research-report.html?wapkw=peer+research, 2012. [Online; accessed 15-February-2015].

[4] V. Černỳ. Thermodynamical approach to the traveling salesman problem: An efficient simulation algorithm. *Journal of optimization theory and applications*, 45(1):41–51, 1985.

[5] C. H. Ding, X. He, H. Zha, M. Gu, and H. D. Simon. A min-max cut algorithm for graph partitioning and data clustering. In *Proceedings of the IEEE International Conference on Data Mining (ICDM)*, pages 107–114. IEEE, 2001.

[6] R. Eglese. Simulated annealing: a tool for operational research. *European journal of operational research*, 46(3):271–281, 1990.

[7] W. Fang, X. Liang, S. Li, L. Chiaraviglio, and N. Xiong. Vmplanner: Optimizing virtual machine placement and traffic flow routing to reduce network power costs in cloud data centers. *Computer Networks*, 57(1):179–196, 2013.

[8] D. S. Johnson, C. R. Aragon, L. A. McGeoch, and C. Schevon. Optimization by simulated annealing: An experimental evaluation; part i, graph partitioning. *Operations research*, 37(6):865–892, 1989.

[9] A. Khajeh-Hosseini, D. Greenwood, and I. Sommerville. Cloud migration: A case study of migrating an enterprise it system to iaas. In *IEEE International Conference on Cloud Computing (CLOUD)*, pages 450–457. IEEE, 2010.

[10] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi. Optimization by simulated annealing. In *Neurocomputing: foundations of research*, pages 551–567. MIT Press, 1988.

[11] T. C. Koopmans and M. Beckmann. Assignment problems and the location of economic activities. *Econometrica: journal of the Econometric Society*, pages 53–76, 1957.

[12] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi. Cloud computing: The business perspective. *Decision Support Systems*, 51(1):176–189, 2011.

[13] X. Meng, V. Pappas, and L. Zhang. Improving the scalability of data center networks with traffic-aware virtual machine placement. In *Proceedings of INFOCOM*, pages 1–9. IEEE, 2010.

[14] B. Oommen and D. Ma. Deterministic Learning Automata Solutions to the Equipartitioning Problem. *IEEE Transaction Computer*, 37(1):2–13, 1988.

[15] B. J. Oommen, D. S. Croix, et al. Graph partitioning using learning automata. *IEEE Transactions on Computers*, 45(2):195–208, 1996.

[16] C. Press. Cisco data center infrastructure 2.5 design guide. 2007.

[17] J. Sonnek, J. Greensky, R. Reutiman, and A. Chandra. Starling: Minimizing communication overhead in virtualized computing platforms using decentralized affinity-aware migration. In *International Conference on Parallel Processing (ICPP)*, pages 228–237. IEEE, 2010.