2015 IEEE 12th International Conference on Autonomic Computing

Toward Hierarchical Mixed Integer Programming for Pack-to-Swad Placement in Datacenters

Ye Xia*, Mauricio Tsugawa†, Jose A. B. Fortes† and Shigang Chen*
*Department of Computer and Information and Engineering
†Department of Electrical and Computer Engineering
University of Florida, Gainesville, Florida, USA

***Abstract*—In this paper, we introduce a pack-centric approach to datacenter resource management by abstracting a system as a pack of resources and considering the mapping of these packs onto physical datacenter resource groups, called swads. The assignment of packs/VMs to swads/PMs is formulated as an integer optimization problem that can capture constraints related to the available resources, datacenter efficiency and customers' complex requirements. Scalability is achieved through a hierarchical decomposition method. We illustrate aspects of the proposed approach by describing and experimenting with a concrete and challenging resource allocation problem.**

***Keywords*-cloud computing; datacenter; virtual machine placement; resource management; integer programming**

I. Introduction

Over the last ten years computational clouds have become widely used by enterprises as their most cost-effective means to deploy IT services. However, the value proposition offered by computational clouds has evolved to go beyond cost-effective on-demand hosting/management of IT resources and elasticity [1]. The added value now includes the ability to offer entire IT systems as a service (versus isolated resources that the customer needs to build into a system) that can quickly adapt to changing business environments (versus being statically configured) and are automatically optimized in response to changes in either the environment or the workload. These new capabilities - agility and continuous optimization - are enabled by building datacenters where resources and environments can be (re)configured programmatically on the basis of monitoring information and predictive models of behavior and workload. Industry has recently coined the term "software-defined" to refer to these new types of datacenters [1]–[3], built on the foundation of virtualization of computing, network, storage, software and all other datacenter resources.

The above-mentioned evolution of datacenters introduces new challenges in management as well as new opportunities. The challenges are scalability, system-orientation, and optimization supporting both datacenter efficiency and customer system agility and performance. The opportunities are in considering systems as units of management therefore creating hierarchical structure in resource management which contributes to scalability. In this paper, we pursue these opportunities by abstracting a system as a *pack* of (virtual) resources and considering the mapping of these packs onto physical datacenter resource groups (called *swads*) subject to constraints related to the physical datacenter topology and available resources, datacenter efficiency and customers' complex requirements. This is a major departure from most previous datacenter management approaches which rely on a VM (virtual machine)-centric view [4]–[6] rather than a system/pack-centric approach (hereon, pack-centric approach).

The main contributions of this paper are: (1) We propose a pack-centric approach to datacenter resource management, which is capable of supporting system-oriented services. (2) We adopt mixed integer programming formulations and algorithms for datacenter resource management problems since they are capable of capturing and solving complex, changing, sometimes vague requirements and constraints, thus providing the envisioned agility of the next-generation datacenters. Since optimization is involved, such formulations will lead to improved performance with respect to the datacenter's and customers' objectives. (3) For scalable solutions, we propose hierarchical decomposition of each resource management problem in accordance with the pack and swad hierarchies that are often natural in system-oriented, pack-centric datacenters.

Prior studies generally focus on smaller-scale, flat VM-to-PM assignment problems with simpler objectives and constraints. They generally avoid integer programming formulations all together. In the cases where integer programming is used, attention is usually on developing specialized combinatorial algorithms such as multi-dimensional bin-packing [7], [8], graph algorithms [5] or sophisticated heuristics [9], which are only applicable to special problems with the structures required by those algorithms. Practical cloud systems usually adopt less sophisticated heuristics, such as first-fit and round-robin, as evidenced by open-source middleware stacks [10], [11]. While they are highly scalable, simple heuristics can also be underachieving in the performance objective. The authors of [12] also propose an idea of hierarchical decomposition, but on a specific problem with the specific objective of reducing datacenter network traffic by intelligent placement of the VMs.

II. Pack-Centric Datacenter Management

A. Definitions and Examples of Pack and Swad

Pack Hierarchy: We propose a new abstraction called *pack*, which is a set of *VMs, smaller packs and other (virtual) resources* that should be placed as a group in a datacenter for the purpose of resource sharing or performance enhancement. This recursive definition allows a customer to organize its resource requirement in a hierarchical structure, as illustrated by Fig. 1, which shows a scenario of a multinational corporation outsourcing its IT infrastructure to the cloud. The corporation has a branch in London, a branch in Shanghai, and its headquarters in San Jose, corresponding to three packs. The headquarters pack further consists of a firewall VM and three lower-level packs, describing the resource requirement by the management, finance and engineering departments,

978-1-4673-6971-8/15 $31.00 © 2015 IEEE
DOI 10.1109/ICAC.2015.23

respectively. Suppose the London pack requires 100 VMs for various databases and other computing tasks. The workload on these VMs and the resource requirement cannot be fully predicted since the VMs may be assigned to different tasks as needed in the future. Instead of committing a fixed amount of each resource to each VM, it makes more sense to specify the combined requirement for each resource for the whole pack and let the VMs share the common pool of resources dynamically as needed. A natural way to facilitate resource sharing is to place these VMs as a group in a cluster of colocated PMs (physical machines). A controller can be implemented to monitor in real time the resource usage by the pack and dynamically adjust the resource distribution among the VMs.

Swad Hierarchy: We define a *swad* as a set of PMs, other physical resources (e.g., network storage) or lower-level swads in a cloud system. The resource capacity of a swad is equal to the sum of the capacities of its components, possibly excluding a certain percentage of resources that are set aside to support elasticity (which allows a VM or pack to dynamically scale up its resources in real time). In the example of Fig. 2 with a fat-tree topology [13], a swad corresponds to all the servers of a rack, shown by the nodes labeled with "swad" in the right figure. Some lower-level swads are grouped into a higher-level swad, labeled as "SWAD", giving rise to a hierarchical structure (tree), with the whole cloud system as the root swad.

The swad hierarchy is established by the cloud provider according its resource management policies, proximity of physical resources and various other constraints. The pack hierarchy is established by combining the customers' pack specifications and the cloud provider's considerations. A customer's request may be already in the form of a smaller pack hierarchy, such as the example in Fig. 1. The cloud provider collects such pack requests from the customers and establishes the final pack hierarchy (see Fig. 4). There is a great deal of flexibility in constructing both hierarchies. A comprehensive study is outside the scope of this paper. Subsequently, we assume both hierarchies are given, with each forming a tree. The depth of a node in a tree is called the *level* of the node, with the root node at level 0 by convention. The leaves of the pack hierarchy can be VMs or packs; the leaves of the swad hierarchy can be PMs or swads.

B. Hierarchical Decomposition - Pack-to-Swad Assignments

Besides the benefits of allowing system-oriented cloud services and increased customer agility, the other main use of the pack/swad hierarchies is to break a large resource allocation problem of enormous complexity into a series of much smaller subproblems that are far easier to solve and can be solved in parallel[1]. There is an assignment subproblem associated with each swad on the swad hierarchy, which assigns some packs/VMs to the child swads/PMs of the focal swad.

Starting from the root of the swad hierarchy, the assignment is performed recursively. First, the root of the pack hierarchy is assigned to the root of the swad hierarchy. The next step is to assign the children of the root pack to the children of the root swad. The assignment process continues downward along the swad hierarchy, in either depth-first or breath-first traversal order. In general, consider a swad at level i, where $i = 0, 1, \ldots$.

[1]All subproblems at the lowest levels can be solved independently from each other. As the experimental results show, the computation time for these subproblems dominates the overall computation time.

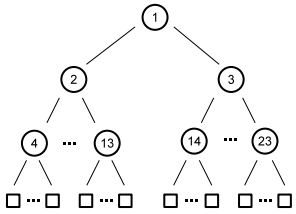

Fig. 3. An example of a swad hierarchy prepared by the cloud provider. Circles are swads; squares are PMs.

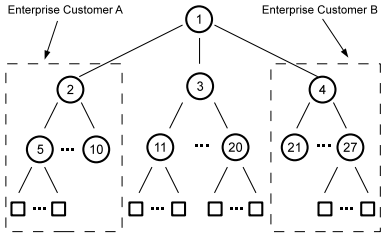

Fig. 4. An example of a final pack hierarchy. Circles are packs; squares are VMs. Enterprise customers A and B specified their own pack hierarchies, as shown in dashed boxes. Packs 11 to 20 are created by the cloud provider; each aggregates a set of VMs requested by individual customers.

Assume that some level-i packs have been assigned to the swad in question. For that swad, the assignment decision is to assign the child packs/VMs of those level-i packs to the child swads/PMs of the focal swad.

Example: Consider the assignment associated with the pack hierarchy in Fig. 4 and swad hierarchy in Fig. 3. The first step is trivial: Pack 1 is assigned to swad 1. The next step is to assign packs 2, 3 and 4 to swads 2 and 3. Suppose the result of the assignment (by solving an optimization problem chosen by the provider) is that packs 2 and 3 are assigned to swad 2, and pack 4 is assigned to swad 3. Next, for swad 2, the assignment problem is to assign packs $5 - 20$ to swads $4 - 13$; for swad 3, the problem is to assign packs $21 - 27$ to swads $14 - 23$. Suppose packs $11 - 14$ are assigned to swad 13 in the assignment step associated with swad 2 (based on solving another optimization problem). The assignment problem associated with swad 13 is to assign all the VMs in packs $11 - 14$ to the PMs in swad 13.

C. Periodic Re-Optimization by Integer Programming

Another key idea of our datacenter management framework is to perform periodic re-optimization and, based on that result, update the assignment of packs/VMs to swads/PMs. The optimization problems are formulated as integer programming problems to capture complex constraints and customer requirements, such as various colocation or anti-colocation constraints, topological relationship among components, or even workflow precedence relationship. Periodic re-optimization may be triggered by timers or by events (e.g., when the resource efficiency is below a threshold).

When re-optimization is due: The following actions are taken.

- A subset of packs/VMs is selected and a subset of PMs/swads is selected to participate in re-optimization. The selection depends on the provider's policy and the states of the packs/VMs and swads/PMs, e.g., whether a

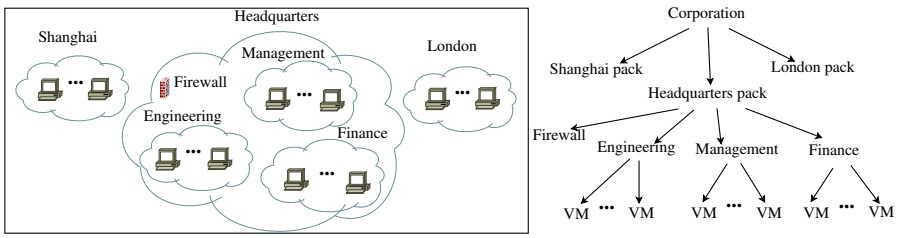

Fig. 1. VMs and other virtual resources can be organized through a recursive, hierarchical pack structure determined by administrative boundaries, locations and resource sharing requirement.

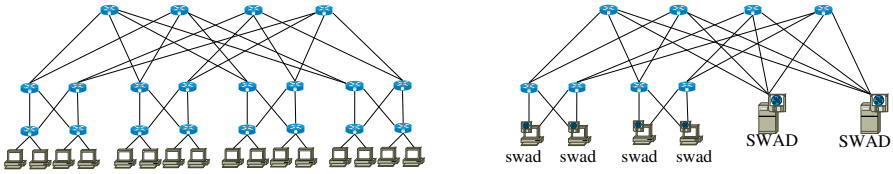

Fig. 2. Swad-based hierarchical abstraction of a cloud system

PM is running a workload that cannot be interrupted.

- A large integer optimization problem is formulated for optimal assignment of the packs/VMs to swads/PMs, taking into account both the customers' and the provider's objectives and constraints. The formulation covers all the packs/VMs and all the swads/PMs selected in the previous step.
- A pack hierarchy and a swad hierarchy are constructed, and the assignment subproblems in hierarchical decomposition are solved by integer programming algorithms. The packs/VMs are placed according to the solution, which may involve VM migration.

In-between re-optimization events: Customer requests for new or additional resources are handled immediately upon arrival. If a request is simple (e.g., for a set of non-interacting VMs), it is handled by some online heuristic algorithm such as first-fit or the randomized algorithm in [3]. If a request is complex (e.g., a pack with its own hierarchy and dependencies among its components), a small-scale integer optimization problem is formulated to allocate the requested resources, involving only the current request(s) and a small set of physical resources.

III. An Example and Experimental Results

The following sample problem serves to illustrate the capabilities of the integer programming formulation in describing complex constraints. In this case, it easily captures a subtle anti-colocation constraint of disks. The example also demonstrates the enormous scale and complexity of datacenter management problems in general. We will show how hierarchical decomposition provides a scalable solution framework.

Constraints: The problem is to assign N VMs to M PMs with disk exclusivity constraints, where N and M can both be fairly large, e.g., thousands or more. Each VM has the following resource requirement: memory, vCPU, number of local disk volumes (virtual ones) and their sizes. Each of the M PMs has certain memory, a number of vCPUs, and a number of local disks and their sizes. These local disks may be in the PM or directly attached. With respect to each resource (e.g., vCPUs or memory), the constraints are that the total amount of resource required by all the VMs assigned to each PM j cannot exceed the resource capacity of PM j. When a VM i requests multiple local disks, there is often an *exclusivity requirement*: no physical disk of the PM (to which VM i is assigned) can contain more than one of VM i's requested virtual disks[2]. A final set of constraints is that the capacity of each physical disk must be greater than or equal to the aggregate size of all virtual disks assigned to it.

Let the sets of VMs and PMs be denoted by $\mathcal{V}$ and $\mathcal{P}$, respectively. For each VM i, let α_i be the number of vCPUs required and let β_i be the memory requirement (in GiB). Suppose for each VM i, a set of virtual disks is requested and the set is denoted by R_i. For each of the requested virtual disks $k \in R_i$, let ν_{ik} be the requested disk volume size (in GB). For each PM j, let C_j be the number of available vCPUs, M_j be the amount of memory (in GiB), and D_j be the set of available physical disks. The sizes of the physical disks are denoted by S_{jl} (GB) for $l \in D_j$. For each $i \in \mathcal{V}$ and each $j \in \mathcal{P}$, let x_{ij} be the binary assignment variable from VM i to PM j, which takes the value 1 if i is assigned to j and 0 otherwise. The binary variables y_{ikjl} are used for disk assignment: y_{ikjl} is set to 1 if VM i is assigned to PM j and the requested virtual disk k, where $k \in R_i$, for VM i is assigned to the physical disk l of PM j, where $l \in D_j$; it is set to 0 otherwise. The problem's constraints can be written as follows:

$$y_{ikjl} \le x_{ij}, \quad i \in \mathcal{V}, j \in \mathcal{P}, k \in R_i, l \in D_j \tag{1}$$

$$\sum_{j \in \mathcal{P}} \sum_{l \in D_j} y_{ikjl} = 1, \quad i \in \mathcal{V}, k \in R_i \tag{2}$$

$$\sum_{j \in \mathcal{P}} x_{ij} = 1, \quad i \in \mathcal{V} \tag{3}$$

$$\sum_{k \in R_i} y_{ikjl} \le 1, \quad i \in \mathcal{V}, j \in \mathcal{P}, l \in D_j \tag{4}$$

$$\sum_{i \in \mathcal{V}} \sum_{k \in R_i} \nu_{ik} y_{ikjl} \le S_{jl}, \quad j \in \mathcal{P}, l \in D_j. \tag{5}$$

[2]Separate physical disks allow the end-user of the VM to enjoy higher total disk throughput and/or more fault tolerance.

$$\sum_{i \in \mathcal{V}} \alpha_i x_{ij} \leq C_j, \quad j \in \mathcal{P} \tag{6}$$

$$\sum_{i \in \mathcal{V}} \beta_i x_{ij} \leq M_j, \quad j \in \mathcal{P}. \tag{7}$$

Costs and Optimization Objective: We assume that, when a PM j is turned on to serve some VMs, there is a fixed cost $\hat{c}_j$ associated with running the PM; when the PM is off, there is zero cost involved. The operation cost may include the average energy cost when a machine is running and typical maintenance cost. Let z_j be a 0-1 variable indicating whether PM j is used by some VMs. To ensure that $z_j = 1$ if and only if $x_{i,j} = 1$ for some $i \in \mathcal{V}$, we add the following two constraints, where B is a large enough constant, e.g., $B = N$.

$$z_j \leq \sum_{i \in \mathcal{V}} x_{i,j}, \quad j \in \mathcal{P} \tag{8}$$

$$B z_j \geq \sum_{i \in \mathcal{V}} x_{i,j}, \quad j \in \mathcal{P}. \tag{9}$$

The optimization objective is to minimize the total operation cost: $\min_{x,y,z} \sum_{j \in \mathcal{P}} \hat{c}_j z_j$.

Experimental Results: For brevity, we only show one set of results[3]. The experiments are to assign 1000 VMs to 1000 PMs using our hierarchical decomposition method. We take subsets of the allowed VM and PM types in Amazon's EC2 [15]. To solve the optimization subproblems in the decomposition, we use the integer programming software Gurobi. The results of three experiments are summarized in Table I. Mix 1 and mix 2 have different mixes of VM and PM types.

For the first two experiments, we split the VMs into 25 packs and the PMs into 25 swads randomly. In this case, the pack and swad hierarchies have similar structures, each having two levels. Take the pack hierarchy as an example. The root pack has 25 child packs, each of which has 40 VMs as children. In the first level of assignment, we assign the 25 packs to the 25 swads by solving an optimization subproblem. A second-level assignment is performed for each of the swads that has some packs assigned to it. For each such swad, we collect all the VMs in all the packs that are assigned to the swad, and we collect all the PMs in the swad. We then perform optimal allocation of the VMs to the PMs using the formulation provided earlier. For each of the swads, the minimum cost is given by the optimization solution. The overall cost is the sum of those minimum costs.

Take mix 1 as an example. The two-level decomposition algorithm achieves a total cost $82,540$ (normalized). This number should be compared with a sophisticated randomized heuristics that we used for cost comparison, which has an average (over 50 runs) total cost $150,573$. The result suggests that the cost improvement of our scheme can be significant. The computation time for the first-level assignment is several seconds. A total 17 swads are used after the first-level pack-to-swad assignment. For the second-level VM-to-PM assignments, the total running time is 1281 seconds, which is the aggregate for 17 different computations for the 17 used swads. The average running time is 75 seconds per swad. Note that these 17 assignment subproblems are completely independent and can be solved in parallel on different computers. In general, the bottom-level VM-to-PM assignment subproblems dominate in computation time. To get a solution within a prescribed time budget, the size of each such subproblem needs to be limited, which can be achieved by sufficient decomposition. For instance, the third experiment is mix 1 with smaller pack/swad sizes (20 VMs or PMs); both the total and per-swad running times are drastic reduced. Finally, the heuristic algorithm has fairly long running times, hundreds of seconds per run. It is not sufficiently scalable for larger problems, whereas the decomposition algorithm is.

[3]See [14] for more results and details.

TABLE I
SUMMARY OF RESULTS: TWO-LEVEL DECOMPOSITION V.S. HEURISTICS

Experiments		Two-Level Decomp.	Heuristics
Mix 1	Cost	82540	150573
	Run Time (s)	1281;75 per swad	
Mix 2	Cost	487840	601914
	Run Time (s)	3366;280.5 per swad	
Mix 1; Smaller Pack/Swad Sizes	Cost	98040	150573
	Run Time (s)	202;7.8 per swad	

REFERENCES

[1] C.-S. Li, B. L. Brech, S. Crowder, D. M. Dias, H. Franke, M. Hogstrom, D. Lindquist, G. Pacifici, G. Kandiraju, H. Franke, M. D. Williams, M. Steinder, and S. M. Black, "Software defined environments: An introduction," *IBM Journal of Research and Development*, vol. 58, no. 2/3, March/May 2014.

[2] G. Kandiraju, H. Franke, M. D. Williams, M. Steinder, and S. M. Black, "Software defined infrastructures," *IBM Journal of Research and Development*, vol. 58, no. 2/3, March/May 2014.

[3] W. C. Arnold, D. J. Arroyo, W. Segmuller, M. Spreitzer, M. Steinder, and A. N. Tantawi, "Workload orchestration and optimization for software defined environments," *IBM Journal of Research and Development*, vol. 58, no. 2/3, March/May 2014.

[4] J. Xu and J. Fortes, "Multi-objective virtual machine placement in virtualized data center environments," *Proceedings of IEEE Online Green Communications Conference (GreenCom)*, 2010.

[5] M. Wang, X. Meng, and L. Zhang, "Consolidating virtual machines with dynamic bandwidth demand in data centers," *Proceedings of IEEE INFOCOM*, pp. 71–75, 2011.

[6] J. Xu and J. Fortes, "Optimization in autonomic data center resource and performance management," *Technical Report, Department of Electrical and Computer Engineering, University of Florida*, 2012.

[7] M. Chen, H. Zhang, Y. Y. Su, X. Wang, G. Jiang, and K. Yoshihira, "Effective VM sizing in virtualized data centers," *Proc. of IFIP/IEEE Integrated Network Management (IM)*, 2011.

[8] Y. Ajiro and A. Tanaka, "Improving packing algorithms for server consolidation," *Proc. of Computer Measurement Group Conference (CMG)*, 2007.

[9] 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*, vol. 57, no. 1, pp. 179–196, 2013.

[10] "Apache CloudStack Project," http://cloudstack.org/.

[11] "OpenStack Project," http://www.openstack.org/.

[12] X. Meng, V. Pappas, and L. Zhang, "Improving the scalability of data center networks with traffic-aware virtual machine placement," in *Proceedings of IEEE INFOCOM*, 2010.

[13] M. Al-Fares, A. Loukissas, and A. Vahdat, "A scalable, commodity data center network architecture," *Proc. of ACM SIGCOMM*, 2008.

[14] Y. Xia, M. Tsugawa, J. A. B. Fortes, and S. Chen, "Hierarchical mixed integer programming for pack-to-swad placement in datacenters: Concept and analysis," *In Preparation*, http://www.cise.ufl.edu/%7Eyx1/publications/cloudlong.pdf.

[15] Amazon, "Amazon EC2 Instances," http://aws.amazon.com/ec2/instance-types/.