On the Feasibility of Automation for Bandwidth Allocation Problems in Data Centers

Yifei Yuan, Anduo Wang, Rajeev Alur, and Boon Thau Loo
University of Pennsylvania

Abstract—**Mapping virtual networks to physical networks under bandwidth constraints is a key computational problem for the management of data centers. Recently proposed heuristic strategies for this problem work efficiently, but are not guaranteed to always find an allocation even when one exists. Given that the bandwidth allocation problem is NP-complete, and the state-of-the-art SAT solvers have recently been successfully applied to NP-hard problems in planning and formal verification, the goal of this paper is to study whether these SAT solvers can be used to solve the bandwidth allocation problem exactly with acceptable overhead. We investigate alternative ways of encoding the allocation problem, and develop techniques for abstraction and refinement of network graphs for scalability. We report experimental comparisons of the proposed encodings with the existing heuristics for typical data-center topologies.**

I. Introduction

Allocating computing resources to customers' requests is the central task for a data center provider. The requests submitted usually define virtual networks involving a number of virtual machines (VMs) and also the bandwidth requirement for virtual links between VMs. To handle such a request, the data center provider should allocate server resources, as well as bandwidth on the links of the data center's physical network.

However, bandwidth allocation is a computationally hard problem. It is NP-complete to determine whether there is a valid allocation even for simple topologies of physical networks and virtual networks. Current proposed techniques focus on designing heuristic algorithms processing requests efficiently ([1], [2], [3]). These heuristics only consider local constraints, e.g. if there is enough link bandwidth of a server to host a VM [1], to determine allocation. While these techniques work efficiently, they provide no guarantee to always succeed in finding an allocation even if one exists.

On the other hand, recent success of applying SAT/SMT solvers [4], that is, solvers for constraint satisfaction problems, to NP-hard problems in planning and formal verification suggests a promising way to solve NP-hard problems in practice ([5], [6], [7], [8]). In this paper, we encode the bandwidth allocation problem into SAT formulas and utilize state-of-the-art SAT/SMT solvers to find the feasible allocation. We develop abstraction and refinement for the physical networks for scalability. Our experimental results show that the SAT approaches work effectively with acceptable overhead for small data center networks.

This research was partially supported by the NSF Expeditions in Computing project ExCAPE (CCSF 1138996).

The remaining paper is organized as follows. Section II provides a formulation of the bandwidth allocation problem, and the hardness result of it when restricting to the topology of trees. In Section III, we show the SAT encoding of the bandwidth allocation problem and we develop abstraction and refinement techniques for scalability. In Section IV, we show the simulation results. Section V concludes this paper.

II. Problem Formulation

In this section, we provide a formal definition of the bandwidth allocation problem. Formally, we model the physical network of a data center as a graph $PN = (A \cup B, L)$, where A is the set of host servers, B is the set of switches. L is the set of physical links that connect servers(switches) with switches. Moreover, each host server $s \in A$ has a capacity $c(s)$ that models the maximum number of virtual machines which a server can host. Each physical link $l \in L$ has a bandwidth $b(l)$. The virtual network is also modeled as a graph $VN = (V, E)$, and V is the set of virtual machines, E is the set of virtual links connecting virtual machines. Each virtual link $e \in E$ has a bandwidth requirement $r(e)$. The mapping $f : V \to A$ maps a VM to a host server. Due to the capacity of servers, f should not map VMs more than a server's capacity. Let P denote the set of paths in PN. The mapping $\rho : E \to P$ defines the routing path for each virtual link. That is $\rho(v_1, v_2) = (f(v_1), \cdots, f(v_2))$. To meet the bandwidth requirement, every physical link on the routing path must have enough available bandwidth for the virtual link.

We define the *bandwidth allocation* problem as follows.

Given a physical network $PN = (A \cup B, L)$ with server capacity c and link bandwidth b, and a virtual network $VN = (V, E)$, the bandwidth allocation problem seeks to find the mapping functions f and ρ, satisfying the following two conditions: (1) $\forall s \in A$, $c(s) \geq |\{v \in V | f(v) = s\}|$, and (2) $\forall l \in L$, $b(l) \geq \sum_{e \in E : l \in \rho(e)} r(e)$.

A feasible allocation can be checked in polynomial time and thus the bandwidth allocation problem is in the class NP. It is also proved in [1] that the bandwidth allocation problem is NP-hard in a general physical network.

In data centers, the tree structure is widely deployed for the physical network topology. Even though the routing path in a tree between any two servers is unique, the bandwidth constraint is still the bottleneck for solving the problem efficiently. We show that even when restricting the physical network topology to be a tree, the bandwidth allocation problem is still strongly NP-hard, i.e., NP-hard even when the numerical parameters are encoded in unary.

Theorem 1 (NP-hardness for Trees)**.** *Given a physical network PN in the tree structure, and a virtual network VN, finding the mappings f and ρ is strongly NP-complete.*

Proof. Membership in NP is evident. For the NP-hardness, we show the reduction from a strongly NP-complete problem, namely, the 3-partition problem: given a multiset S of $3m$ positive numbers $\{x_1, ..., x_{3m}\}$, partition S into m subsets $S_1, ...S_m$, such that each subset has 3 numbers and the sum of the numbers in each subset is equal. We construct the physical network as a 1-level tree with $m+1$ servers $s_1, ..., s_{m+1}$. Each server s_i connects to a switch s. For $i = 1, ..., m$, set the bandwidth of the link connecting s_i and s to be t, which is the desired sum of each subset in the 3-partition problem, and the bandwidth of link (s_{m+1}, s) to be $3t$, which is the sum of all the numbers. Let the capacity of each server be 3. We construct a virtual network as a 1-level tree with $3m+1$ virtual machines. Suppose the leaves are $v_1, .., v_{3m}$ and the root is v_{3m+1}. The bandwidth requirement for the virtual link (v_i, v_{3m+1}) is x_i. Since the virtual machine v_{3m+1} can only be mapped to the server s_{m+1}, it is evident that there is a one-to-one mapping between these two instances. Therefore, the bandwidth allocation problem for trees is also NP-hard. $\square$

III. SAT Approach

In this section, we provide an alternative solution to the bandwidth allocation problem using SAT/SMT solvers. First, we show how to encode the bandwidth allocation problem into SAT formulas with integers. Second, we develop abstraction and refinement of physical network topologies for scalability.

A. SAT Encoding

In this section, we show how to encode a bandwidth allocation problem into SAT formulas that work for any physical network topology and virtual network topology. For each virtual machine $v \in V$, and server $s \in A$, let $X(v, s)$ be an integer variable indicating $f(v) = s$, i.e. $X(v, s) = 1$ if $f(v) = s$, and $X(v, s) = 0$, otherwise. To ensure that each virtual machine gets mapped, and only mapped to one server, we have the constraint that

$$\alpha_s : \bigwedge_{v \in V} \left(\sum_s X(v, s) = 1 \right).$$

To encode the routing path for each pair of virtual machines, we define the variable $R(l, e, k)$ to indicate that the physical link l is the k'th link of the routing path that is allocated to the virtual link e. The following formula encodes the constraint that there is no more than 1 physical link as the k'th one:

$$\alpha_r : \bigwedge_{e,k} \left(\sum_{l \in L} R(l, e, k) \leq 1 \right).$$

The following constraint ensures that $R(l, e, k)$ indeed encodes a path in the physical network.

$$\alpha_c : \bigwedge_{e,k} \left(\bigvee_{l_1, l_2 : l_1, l_2 \text{ are adjacents}} R(l_1, e, k) \wedge R(l_2, e, k+1) \right).$$

The constraint $\bigvee_{l_1, l_2 : l_1, l_2 \text{ are adjacents}} R(l_1, e, k) \wedge R(l_2, e, k+1)$ means that the k'th and the $k+1$'th physical link on the routing path assigned for the virtual link e should be adjacent. For each physical link $l \in L$ and virtual link $e \in E$, let $Y(l, e)$ be an integer variable indicating the bandwidth of l reserved for virtual link e. We have the following constraint:

$$\alpha_y : Y(l, e) = r(e) \Leftrightarrow \bigvee_k R(l, e, k) = 1.$$

To encode the constraint that for each virtual link, there is a routing path between the host servers to which the two VMs of the virtual link are mapped, we have:

$$\alpha_v : \bigwedge_{\substack{(v_1, v_2) \in E, \\ s_1, s_2 \in A, s_1 \neq s_2}} \Bigg((X(v_1, s_1) = 1 \wedge X(v_2, s_2) = 1) \rightarrow \bigvee_{\substack{l_1 : s_1 \in l_1 \\ l_2 : s_2 \in l_2}} (Y(l_1, e) = r(e) \wedge Y(l_2, e) = r(e)) \Bigg).$$

Finally, the server capacity condition can be encoded as

$$\beta_{server} : \bigwedge_{s \in A} \left(\sum_v X(v, s) \leq c(s) \right)$$

, and the link capacity condition is encoded as

$$\beta_{link} : \bigwedge_{l \in E} \left(\sum_e Y(l, e) \leq b(l) \right).$$

Putting all the pieces together, we have the encoding $\Phi_{PN,VN}$ for the bandwidth allocation problem:

$$\Phi_{PN,VN} = \alpha_s \wedge \alpha_r \wedge \alpha_c \wedge \alpha_y \wedge \alpha_v \wedge \beta_{server} \wedge \beta_{link}.$$

The following theorem easily follows.

Theorem 2. *Given any physical network $PN = (A \cup B, L)$ and virtual network $VN = (V, E)$, there exists mappings f and ρ satisfying the requirements of the bandwidth allocation problem, if and only if the formula $\Phi_{PN,VN}$ is satisfied.*

B. Abstraction and Refinement

We use 2-level tree as an example topology for the data centers to demonstrate how to abstract and refine the network topology. In the tree topology, all the leaves are host servers, and there are two levels of switches. On the first level, the switches connect with the host servers, and the root switch sitting on the second level connects with the switches on the first level. For notation brevity, let T denote the physical tree topology. We denote the root of the 2-level tree as r, and its children as $w_1, ..., w_k$, and $s_{i,j}$ as the jth children of w_i.

The routing path for any two servers in a tree is unique, therefore, the encoding involving $R(l, e, k)$ can be omitted.

For each pair of host servers u, v, let $p(u,v)$ be the unique path between them. The constraint α_v can be simplified as:

$$\alpha_v : \bigwedge_{\substack{(v_1,v_2)\in E,\\ s_1,s_2\in A, s_1\neq s_2}} \Bigg((X(v_1,s_1)=1 \wedge X(v_2,s_2)=1) \rightarrow \bigwedge_{l\in p(s_1,s_2)} (Y(l,e)=r(e)) \Bigg).$$

Therefore, the whole formula is:

$$\Phi_{PN,VN} = \alpha_s \wedge \alpha_v \wedge \beta_{server} \wedge \beta_{link}.$$

a) Abstraction: In the abstraction phase, we "compress" T into a 1-level tree T_{abs}, which has a single root r with its children $w_1, ..., w_k$. By abstracting the subtree rooted at w_i, we set the capacity of w_i to be the sum of capacities of all its children. That is, $c(w_i) = \sum_j c(s_{i,j})$. Using the encoding technique above, we build the constraint $\Phi_{T_{abs},VN}$ and solving the constraint gives us a solution for the allocation problem for the abstracted tree and the original virtual network.

b) Refinement: In the refinement phase, we need to solve for the subtree T_i with root w_i and its children $s_{i,1}, ...s_{i,m}$. By solving the abstracted tree T_{abs} in the first phase, we know the set of virtual machines that are mapped to this subtree. The virtual network VN_i that are mapped to the subtree T_i is a subgraph of the original virtual network. Suppose the set of VMs that are mapped to the subtree is V_i, then $VN_i = (V_i, E_i)$, and here $(v_1,v_2)\in E_i$ if and only if $v_1, v_2 \in V_i$ and $(v_1,v_2)\in E$. The formula Φ_{T_i,VN_i} encodes the constraint for mapping the virtual network VN_i to the subtree T_i. However, this is not sufficient to ensure that the mapping is feasible for the original virtual network. In fact, there may be a virtual link of which only one end is mapped to the subtree. To handle this situation, we need to establish a route for the server which the virtual machine is mapped to and the root switch in the subtree. That is,

$$\alpha_{v'} : \bigwedge_{\substack{(v_1,v_2)\in E, v_1\in V_i,\\ v_2\notin V_i, s_1\in T_i}} \Bigg(X(v_1,s_1)=1 \rightarrow Y((s_1,w_i),e)=r(e) \Bigg).$$

Therefore, the formula for the refinement is

$$\Phi_{T_i,VN_i,VN} = \Phi_{T_i,VN_i} \wedge \alpha_{v'}.$$

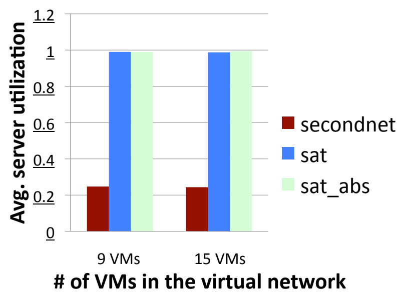
The algorithm for the abstraction and refinement approach is shown as algorithm 1. The algorithm first solves the formula of the abstracted tree (line 4), and then refines each subtree (line 5-7). To facilitate the search, if the refinement phase of some subtree fails, we stop refining the next subtree and return back to solve the first phase. To force the SMT solver to provide a different solution, we add the counter-example that makes the refinement fail. That is $\alpha_{counter} : \neg\big(\bigwedge_{v\in V_i} X(v,w_i)=1\big)$ (line 8-11). If all the subtrees can be refined, the algorithm finds a feasible solution (line 13-17). Otherwise, the formula Φ for the abstracted tree is unsatisfied after a number of iterations, in which case, there is no feasible solution (line 19).

Algorithm 1 Abstraction&Refinement SAT solving.

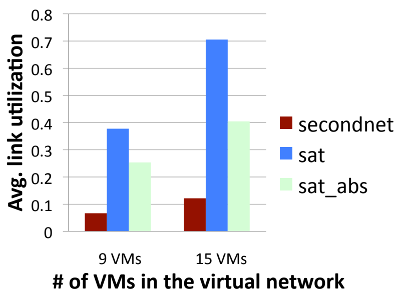
```
1:  build the formula $\Phi_{T_{abs},VN}$ for the abstracted tree $T_{abs}$;
2:  $\Phi = \Phi_{T_{abs},VN}$;
3:  while $\Phi$ is satisfied do
4:      solve $\Phi$;
5:      for all $i$ do
6:          let $VN_i = (V_i, E_i)$ be the virtual network that needs to be mapped to the subtree $T_i$;
7:          build the formula $\Phi_{T_i,VN_i,VN}$;
8:          if $\Phi_{T_i,VN_i,VN}$ is unsatisfiable then
9:              $\Phi = \Phi \wedge \neg\big(\bigwedge_{v\in V_i} X(v,w_i)=1\big)$;
10:             break;
11:         end if
12:     end for
13:     if all subtrees can be refined then
14:         set $f(v) = s$ if $X(v,s)=1$;
15:         compute $\rho$ using the unique routing path in $T$;
16:         return $f$ and $\rho$;
17:     end if
18: end while
19: return no solution;
```

IV. Simulation Results

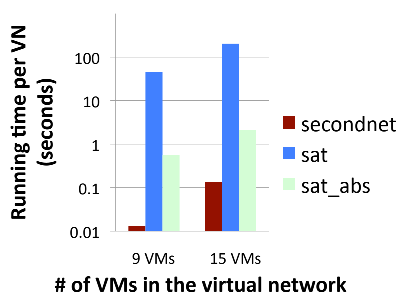
In this section, we show some empirical evaluation of our SAT solution. The topology of the physical network in our evaluation is a 2-level tree and the leaves denote the host servers and there are 2-levels of switches. There are 200 servers and each server can host 4 VMs and we set the bandwidth of the lower level link to be 20, while the higher level links' bandwidth is set to be 200. We use the topology generated by connecting 3 complete graph with 3 links as the topology of the virtual network. This topology is used to model the distributed storage systems that have 3 identical replicas, and each replica communicates with one another. We use two different size of replicas, namely 3 VMs and 5 VMs in one replica, and thus 9 VMs and 15 VMs in the virtual network respectively. In each replica, the bandwidth requirement for each virtual link ranges from 0 to 2 at random, and the bandwidth of the links connecting each replica is always 1. That is, each virtual link requires 5% to 10% bandwidth of that of the lower level links in the physical network. For comparison, we run the SAT encoding algorithm without abstraction and refinement (referred as *sat*), as well as the SAT encoding with the abstraction and refinement (referred as *sat_abs*). In addition, we compare the algorithm proposed in [1] (referred as *secondnet*). For each virtual network, we try to map as many copies of the virtual network as possible to the physical network using the allocation algorithms. We run 3 times for each virtual network, and compare the server utilization, link utilization and the running time of the 3 algorithms. All the evaluations are run on a server with quad-core 2.67GHz Intel Xeon CPU, 4GB of RAM, and we use



(a) Average Server Utilization.



(b) Average Link Utilization.



(c) Running Time.

Fig. 1: Simulation Results.

Z3 [5] as the SAT/SMT solver in the allocation algorithms. Figure 1a shows the *server utilization* for the 3 allocation algorithms. Server utilization measures the ratio of the number of VMs that are mapped to the physical network and the total capacity of the network. The larger the server utilization is, the more effective the allocation algorithm is. As shown in Figure 1a, both algorithms based on SAT encodings achieve 99% server utilization, while the secondnet heuristic only achieves 30% server utilization in both virtual networks with 9 VMs and 15 VMs, respectively. The reason why secondnet can only map a few virtual networks is that it only takes the requirement for servers into account, and does not consider whether there is a feasible routing path between the servers that two VMs are mapped to. Moreover, secondnet is highly sensitive to the order in which the VM connections are requested. Different orderings result in large differences in server utilization. On the other hand, the techniques based on SAT solving encode the bandwidth allocation problem completely, and they achieve high server utilization.

The *link utilization* measures the ratio between the link bandwidth utilized and the total bandwidth. When achieving the same server utilization, the allocation algorithm with lower link utilization is usually better than those with higher link utilization, because it leaves more bandwidth for future allocation. As shown in Figure 1b, secondnet only maps a few virtual networks and thus results in low link utilization. The sat algorithm achieves very high link utilization. For the virtual network with 15 VMs, the link utilization is more than 70%. Mapping the same number of virtual networks, sat_abs utilizes the link bandwidth no more than $2/3$ of that by sat. This is due to the abstraction technique. With abstraction, VMs are mapped more locally, and thus more communication happens within the subtree. Without abstraction, the two VMs on a virtual link are more likely to be mapped into two different subtrees, and thus increases the link utilization on the links between the root switch and other switches.

Figure 1c shows the running time of allocating 1 virtual network using each allocation technique. The heuristic of secondnet runs in polynomial time, and it is orders of magnitude more efficient than the other two algorithms. In particular, the sat runs for about 3 hours to map 53 virtual networks with 15 VMs, while secondnet takes no more than 2 seconds to map 13 virtual networks. It is also shown that abstraction reduces the running time significantly. To map a virtual network with 15 VMs, sat takes 200 seconds, while sat_abs only uses about 2 seconds. Let's remark that the SAT encoding approaches and existing approaches are not mutually exclusive. In practice, those approaches can be run in parallel and the first outputting feasible allocation is adopted. Moreover, SAT encoding approaches can be applied when optimizing the data center by re-locating allocated virtual networks. In this case, finding the optimal solution is more critical than running time.

V. Conclusion

Bandwidth allocation problem is the key computational problem in data center management. In this paper, we show an alternative approach that encodes the problem into SAT formulas and apply SAT/SMT solvers to solve the problem. We report simulation results showing that by using abstraction and refinement techniques, we are able to provide high quality solutions within acceptable overhead.

References

[1] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "Secondnet: a data center network virtualization architecture with bandwidth guarantees," in *Proceedings of the 6th International COnference*. ACM, 2010, p. 15.

[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron, "Towards predictable datacenter networks," in *ACM SIGCOMM*, 2011.

[3] Y. Zhu and M. Ammar, "Algorithms for assigning substrate network resources to virtual network components," in *Proc. IEEE INFOCOM*, vol. 2, 2006, pp. 1–12.

[4] L. De Moura and N. Bjørner, "Satisfiability modulo theories: introduction and applications," *Commun. ACM*, vol. 54, no. 9, pp. 69–77, Sep. 2011. [Online]. Available: http://doi.acm.org/10.1145/1995376.1995394

[5] ——, "Z3: An efficient smt solver," in *Tools and Algorithms for the Construction and Analysis of Systems*. Springer, 2008, pp. 337–340.

[6] B. Dutertre and L. De Moura, "The Yices SMT solver," *Tool paper at http://yices. csl. sri. com/tool-paper. pdf*, vol. 2, p. 2, 2006.

[7] L. De Moura, S. Owre, H. Rueß, J. Rushby, N. Shankar, M. Sorea, and A. Tiwari, "Sal 2," in *Computer Aided Verification*. Springer, 2004, pp. 496–500.

[8] T. Ball, B. Cook, V. Levin, and S. K. Rajamani, "Slam and static driver verifier: Technology transfer of formal methods inside microsoft," in *Integrated formal methods*. Springer, 2004, pp. 1–20.