2017-07-01

Multi-capacity bin packing with dependent items and its application to the packing of brokered workloads in virtualized environments

Christine Bassem, Azer Bestavros. 2017. "Multi-Capacity Bin Packing with Dependent Items and its Application to the Packing of Brokered Workloads in Virtualized Environments." FUTURE GENERATION COMPUTER SYSTEMS-THE INTERNATIONAL JOURNAL OF ESCIENCE, Volume 72, pp. 129 - 144 (16).

https://hdl.handle.net/2144/25979

Network-Constrained Packing of Brokered Workloads in Virtualized Environments

Christine Bassem
Computer Science Department
Boston University
Boston, MA 02215
Email: cbassem@cs.bu.edu

Azer Bestavros
Computer Science Department
Boston University
Boston, MA 02215
Email: best@cs.bu.edu

***Abstract*—Providing resource allocation with performance predictability guarantees is increasingly important in cloud platforms, especially for data-intensive applications, in which performance depends greatly on the available rates of data transfer between the various computing/storage hosts underlying the virtualized resources assigned to the application. Existing resource allocation solutions either assume that applications manage their data transfer between their virtualized resources, or that cloud providers manage their internal networking resources. With the increased prevalence of brokerage services in cloud platforms, there is a need for resource allocation solutions that provides predictability guarantees in settings, in which neither application scheduling nor cloud provider resources can be managed/controlled by the broker. This paper addresses this problem, as we define the Network-Constrained Packing (NCP) problem of finding the optimal mapping of brokered resources to applications with guaranteed performance predictability. We prove that NCP is NP-hard, and we define two special instances of the problem, for which exact solutions can be found efficiently. We develop a greedy heuristic to solve the general instance of the NCP problem , and we evaluate its efficiency using simulations on various application workloads, and network models.**

I. Introduction

Service brokerage in a cloud computing setting allows *brokers* to act as intermediaries between the resources offered by the data center providers, such as Amazon EC2 [1], and their consumers. Service brokers allow customers to access services that are not usually offered by the individual providers;*e.g.*, in the form of unique services over an existing cloud platform, or in the form of an aggregation of existing services provided by multiple cloud platforms [6], [18]. This brokerage model creates a marketplace, in which the resources of existing cloud platforms are used to provide services tailored according to the customer's need, at a competitive price.

Data-intensive applications, such as MapReduce [10], and Message Passing Interface (MPI) [32] applications, devote most of their processing time to data transfer and manipulation, and their execution time depends mainly on the size of the data to be processed, and the characteristics of the network over which this data is transferred [30]. Resource virtualization has enabled data center providers to offer such applications virtually isolated computing resources for a fraction of the cost of the actual physical hardware. These virtual resources share the data center's network infrastructure, i.e., the fabric, and thus allow all applications to access the fabric in an uncontrolled and opportunistic manner, resulting in unpredictable network performance, which in turn affects the execution times of data-intensive applications, and consequently their costs [19], [22], [2].

Improving the performance predictability of applications has been a great concern, to the extent that public cloud providers started to offer computing resources with access to a dedicated full-bisection, high bandwidth network, as in Amazon's HPC instances [1], and various resource allocation models have been proposed, which guarantee or improve an application's performance predictability on non-dedicated networks. Existing models, that we explore later in the paper, fall into one of two categories; *network-aware application management*, and *application-aware network management*, both of which are not suitable for a brokerage model.

To provide a brokerage service for predictable performance on cloud platforms, we need a model for predictable resource allocation, in which neither the application, nor the data center resources can be managed. We propose a brokerage model, in which a broker acts as an intermediary between the resources provided by the data center providers, and the applications demanding these resources. This model is based on the abstraction of the main properties of both the data center provider, and the application's workload, which allows us to efficiently allocate physical resources obtained from any set of data center providers to applications, without the need to alter their internal management structures.

Fig.1 shows an architectural view of a framework based on our proposed brokerage model. Given a set of virtual machine containers, i.e., slots, obtained from a data center provider, an inference service is used to infer the physical resources available for them. Recent studies have shown that the network resources accessible by a virtual machine (VM) is highly dependent on the properties of the hypervisor on the physical machine that it's placed on [5], [35]. Therefore, we assume that the inference service identifies collocated VM slots, infers the network resources available from their hosting physical machines [25], and the relationship between them [5].

On the application's side, the profiling service is used to understand the behavior of the application's internal components. We assume that it abstracts the application's requirements into the number of homogeneous VMs it needs, and the bandwidth required on the path between each pair of VMs to achieve a predefined execution time. This abstraction is similar to that used in existing application-aware network management

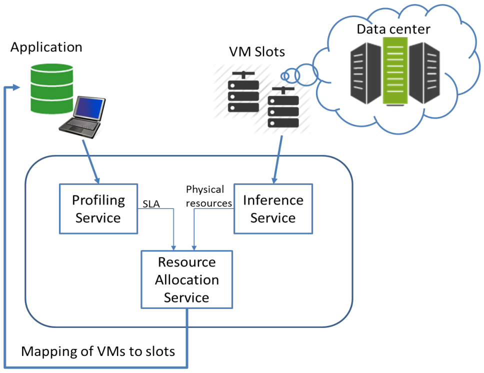


Fig. 1: An architectural view of a framework

models, as it considers only the necessary details that affect the predictability of an application, and it can be obtained using approaches similar to these presented in [36], [37].

Finally, the resource allocation service takes as input the output of the inference and profiling services, and provides predictability guarantees by efficiently allocating physical resources to the applications according to their demand. In this paper, we present the Network-Constrained Packing (NCP) problem, which is that of finding the optimal packing of the components of a brokered workload onto the brokered physical containers. An optimal packing minimizes the cost of using the underlying data center fabric, while satisfying the capacity and demand constraints of the system. By observing the range of data-intensive applications and data center topologies, we exploit the abstractions defined in our model to classify the problem model into several special instances, and to develop polynomial-time exact algorithms for two of these special instances that occur frequently in reality. Finally, we develop a greedy heuristic to solve the NCP problem in the general model, and exemplify its ability to provide a good allocation for data-intensive applications through extensive simulations.

The remainder of this paper is organized as follows; in Section 2, we provide a brief description of the related work. In Section 3, we present our model, define the NCP problem problem, prove it to be NP-hard, and identify special instances of the model. In Section 4, we offer exact algorithms for two of these special instances, with a proof of optimality for each described in [4], and we define the Greedy NCP algorithm to solve the general instance of the problem. In Sections 5, we evaluate the efficiency of the G-NCP algorithm using extensive simulations, and in Section 6, we conclude the paper with a summary of results, and our on-going and future research.

II. Related Work

The resource allocation problem of parallel applications has long been studied in the domains of grid, and cloud computing. In this section, we provide a summary of the approaches used to manage resources allocated to applications with inter-dependent parallel components, in which dependence is in the form of required bandwidth between the components, and we classify them according to the management model adopted.

Application-aware Network Management

It's the model in which the physical network provider is responsible of managing its own physical resources according to the application demands it receives, with an objective to optimize some performance metric. Virtual network embedding (VNE) is the most commonly adopted approach in grid systems, such as PlanetLab, in which a slice of the physical network resources is reserved for the application, according to its properties. In VNE, the physical provider's network follows an internet topology, and is represented as a substrate graph. The application's requested virtual network is embedded within that substrate graph, such that virtual nodes are mapped in a one-to-one fashion to substrate nodes, and virtual links are mapped to paths in the substrate network. Algorithms for VNE vary according to how they perform the mapping; link and node mapping can be performed separately, as in [40], [39], [27], or in a joint manner as in [8], [7]. Approaches also vary according to their embedding objectives, and optimization strategies used; recent surveys of VNE can be found in [12], [11].

With the emergence of data center networks, with well-structured topologies, performance predictability is achieved either using network sharing models, or explicit allocation of network resources. In network sharing [3], [17], bin packing [23] approaches are used allocate computing resources to the workloads, and then providers attempt to share the network resources among the different workloads according to their demand, or payment. For explicit allocation, several approaches have been proposed to perform joint reservation of both the computing and networking resources during allocation to guarantee predictability. Such algorithms differ according to the model assumed for the application's network demands; applications with hose-model abstractions have been assumed in [2], [37], while pipeline-model abstractions have been assumed in [16], [14], [13], and product traffic patterns have been assumed in [38]. Moreover, the degree of network management varies from managing the routing between the switches in the fabric [16], to rate limiting applications by managing the servers' hypervisors [2].

Network-aware Application Management

In this model, the resources allocated to the user cannot be chosen, controlled, or substituted, and the user is responsible for scheduling its inner application tasks to optimize for its own objective. This approach is fairly recent in the cloud computing domain, and algorithms vary according to their placement strategies. In [25], performance predictability is improved by greedily placing communicating tasks in VMs with high bandwidth among them, while in [26], the application components are divided into smaller tasks, and packed inside the virtual machines to maximize virtual machine utilization, while minimizing the cost of tasks communicating across these virtual machines.

III. Network-Constrained Workload Packing

In this section, we present our proposed brokerage service model, and the NCP problem based on it.

A. *The Broker Model*

In our model, we differentiate between the data center provider, and the service broker. A physical data center network is composed of a set of physical severs each capable of hosting multiple virtual machines. Servers are connected to rack switches, which are in turn connected to each other through the datacenter's backbone network, i.e., the fabric, which usually follows some tree topology, such as fat tree [31], or VL2 [15]. VMs hosted on a server are controlled by a hypervisor that multiplexes their access to the various physical resources on the server, and the network it has access to. The data center provider manages all the resources in its network, and allocates them to its tenants, in the form of VM slots.

We define the service broker, as an entity that has access to only a subset of the VM slots provided by the data center provider. The broker is oblivious to the status of other VM slots in the data center, and the data center fabric. An inference service groups collocated VM slots into m servers, and represents the properties of these servers by the tuple (D, C, B), which represents the relationship between the servers, and the computing and bandwidth resources available on them. The distance matrix $D :< m \times m >$ represents the relationship between the servers, i.e., the value of $d_{i,j}$ could represent the network distance between the servers, the delay of communication between them, or some other cost of communication metric as defined by the service broker. The vector $C :< 1 \times m >$ represents the computing resources available at each server, which is basically the number of VM slots available on the server, since we assume a single type of VM in our model. Finally, the vector $B :< 1 \times m >$ represents the bandwidth shared between the VM slots collocated on each server. We note that the communication between VMs on separate servers i and j is limited to a bandwidth of $min\{b_i, b_j\}$, while communication between VMs on the same server experience a virtually unlimited bandwidth with negligible cost.

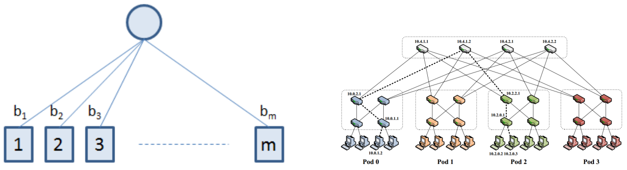
An application in our model is represented by the number of homogeneous virtual machines it requires, n, and the bandwidth required between each pair of VMs represented by the traffic matrix T. The traffic matrix $T :< n \times n >$ is a symmetric matrix with a diagonal of 0's, representing the network bandwidth required on each path between all pairs of VMs.

B. The Network-Constrained Packing Problem

A feasible packing is a mapping of a job's VM to a server hosting a VM slot, with a the mapping function, $M : 1, ..., n \rightarrow 1, ..., m$, such that $M(i) = j, \forall 1 \leq i \leq n, 1 \leq j \leq m$, which satisfies all the demands of the job, while remaining within the capacity constraints of the service broker. Each packing is associated with a cost, and an optimal packing is the one with the minimum associated cost.

Definition 1: **Network-Constrained Packing**. We define the NCP problem as that of finding the optimal packing of the job's components onto the servers with minimal cost of using the data center fabric. The cost of using the data center fabric, as associated with a packing, is defined as the sum of product of the bandwidth allocated between two servers, and the distance between them; $\sum_{1\leq i\leq n}(\sum_{\forall j:M(i)\neq M(j)} t_{i,j} \times d_{M(i),M(j)})$.

1) Problem Formulation: The NCP problem can be formulated as an integer linear optimization problem, which is known to be NP-hard to solve [24].



(a) Virtual 1:1 over-subscription (b) VL2 topology

Fig. 2: Classes of server distances

2) Complexity of the NCP problem : We prove that the NCP problem is NP-hard by providing a polynomial time reduction of the bin packing problem to it, such that the optimal solution found for the constructed NCP problem instance corresponds to an optimal solution for the original bin packing problem instance [4].

C. Instances of the NCP Problem

Based on an examination of the range of variables in our model, specifically the job's traffic matrix and the server distances, we develop a taxonomy of special instances of the problem that we develop based on an examination of the range of variables in our model.

The job's traffic matrix. Work on profiling traffic demands of data-intensive applications have shown that these types of applications usually have a structured workload pattern [9], which can be used to identify the most typically occurring communication patterns among its VMs. Accordingly, we can classify the communication patterns of data-intensive applications into four main classes; constant, tree-based, clustered, and arbitrary traffic. Applications with constant traffic, such as MapReduce [10] and Dryad [20], initially distribute the data to be processed among all of its VMs, and then uniform communication occurs among the VMs through out the job execution, in which data is transferred from one VM to almost all of the other VMs for processing [2], [37]. Jobs with tree-based traffic, such as those found in back-end web services, usually follow a partition-aggregate model of computation, in which a master VM that holds the master task assigns tasks to its workers that in turn assign them to their workers, and so on. In tree-based traffic patterns the bandwidth required between a VM, and its parent VM is greater than or equal to the bandwidth required between that same VM, and its children VMs. Jobs with clustered communication patterns, such as those presented in [2] as virtual over-subscribed clusters, can be used to describe applications with clusters of VMs with high intra-cluster communication and lower inter-cluster communication, as in Hive [34]. Finally, arbitrary communication patterns can be seen in domain-specific applications such as in MPI [32], and MTC [33] applications.

The service broker's distance matrix. The service broker model varies as well, according to the properties that the data center fabric is based upon. We consider three main classes of server distances; equal distance, tree-based distance, and arbitrary distance. The interpretation of these classes varies according to the semantics of the distance metric. For example, if the distance metric represents the delay encountered on the paths between the servers, then equal distances indicate that the data center fabric provides virtual 1:1 over-subscription (Fig.2a), tree-based distances indicate a balanced tree-structured fabric (Fig.2b), and arbitrary distances indicate otherwise.

IV. Algorithms for Network-Constrained Packing

According to our analysis of the NCP problem , and the taxonomy of its special instances, we define polynomial-time exact algorithms for two of these special instances, and we develop the Greedy Network-Constrained Packing (G-NCP) algorithm for a general problem instance, of servers with arbitrary distances, and jobs with arbitrary communication patterns. Before we define the algorithms, we present some essential definitions.

A. Algorithm Definitions

1) Server Effective Capacity: Since the number of VMs from a specific job that can be packed in a server not only depends on the server's available computing resources, but also its bandwidth resources, we need to estimate the server's effective VM slot capacity according to the job's network demands. Given a job that requires n VMs, with required bandwidth described in T following some arbitrary communication model, we compute the average VM-pair bandwidth required by all pairs of VMs in that job. Then, we define the effective capacity of a server as the number of VMs that can be packed in the server, assuming all VM pairs in the job require bandwidth equal to the average VM-pair bandwidth.

$$EC(i,k) = max\{k \leq c_i\}, \text{ such that } k(n-k)t_k \leq b_i$$

in which t_k is the job k's average VM-pair bandwidth, and c_i and b_i are the servers available computing and bandwidth resources respectively.

2) Server Cost: For each server, we need to have an estimate of how it will affect the packing cost of the jobs, and then choose the server with minimum estimated cost. According to the definition of packing cost in the NCP problem , a low cost server would be one which maximizes the collocation of the job's VMs, by having a high effective capacity, and by being closer to other servers with high effective capacities.

Given a job k requesting n VMs, the server i's cost is defined as,

$$Cost(i,k) = (n - EC(i,k)) * \frac{\sum_{1\leq j\leq m} \frac{d_{i,j}}{EC(j,k)}}{m-1}$$

, in which the effective capacity of a server i according to job k's requirements is represented by $EC(i,k)$, and $d_{i,j}$ represents the distance between servers i and j.

The value computed above is suitable for choosing the first server for packing, but it is not sufficient when a set of servers have already been chosen to accommodate a subset of the job's VMs. Given a set of chosen servers, to which n' of the job's VMs have already been mapped, a server's cost for allocating resources to the remaining VMs has to include its distance to these already chosen servers. Thus, we update our definition of server cost to be,

$$Cost(i,k) = \frac{\sum_{j\in chosen} d_{i,j}}{|chosen|} \tag{1a}$$

$$+ (n' - EC(i,k)) * \frac{\sum_{1\leq j\leq m} \frac{d_{i,j}}{EC(j,k)}}{m-1} \tag{1b}$$

3) VM Connectivity: Since the packing cost is directly proportional to the amount of traffic reserved from a server's bandwidth, partitioning the job's VMs into groups of VMs with the minimum traffic demand among them is preferred. Due to the varying capacity constraints of the servers in our model, basic partitioning and clustering techniques cannot be directly applied. We propose a greedy approach to choose a cluster of VMs to be collocated on a single server, in which an initial VM is chosen as the cluster seed, and then VMs that provide maximum connectivity to the cluster are added to it until the compute capacity constraints of the server are violated. We define a *VM's connectivity*, as the difference between the traffic it saves the cluster, and the traffic it adds to the cluster. In this work, we adopt a simple greedy heuristic to decide on the seed VM in a cluster; we choose the unmapped VM with the highest connectivity to other unmapped VMs.

B. Optimal Algorithms for Constant-Traffic Applications

An application with constant traffic patterns (CTA) is represented by the number n of VMs it requires, and a single bandwidth value b required between all its VM pairs. This constant communication model equates that any k VMs collocated together will always need a total of $k(n-k)b$ bandwidth from the server hosting them. The advantage of this property is that it allows us to ignore the bandwidth constraints enforced by a server during packing, by computing the effective capacity of each server before performing the packing. Since the VM-pair bandwidth required is constant for CTA, the effective capacity of a server represents the actual number of the job's VMs that can be packed in that server. This removal of the bandwidth capacity constraints from the packing problem reduces its complexity, and enables us to define exact algorithms for service broker models with equal, and tree-based distances among their servers.

1) Allocation on Equally-Distant Servers (EDS): In the case of servers with equal distances, servers can be considered as unrelated bins, in which the cost of mapping VMs to the bins is only affected by the number of bins used, which resembles the Bin Packing problem with equal-sized items. We develop the E-NCP algorithm, which is similar in nature to the first fit bin packing algorithm [23] with bins sorted in decreasing order according to their effective capacity. Given an instance of a service broker and a job, we initially compute the effective computing capacity of each server in our instance, and sort them according to their computed capacities. Then, we simply map the job's VMs to the servers, in order, until there are no more VMs to be mapped. This complexity of the algorithm is $O(m.logm)$, and we prove that the packing cost associated with this algorithm is always optimal [4].

2) Allocation on Tree-Organized Servers (TOS): In this special instance of the model, the distance between the servers is in the form of a tree, in which the leaves represent the servers, and the inner tree nodes represent virtual switches connecting them to each other. The advantage of this distance structure is that it can be exploited to find the optimal packing in polynomial time. The algorithm's main idea is to annotate every node in the distance tree, in a bottom-up approach, with the optimal cost of hosting n VMs on the servers in the sub-tree rooted at that node. This algorithm's worst case complexity is $O(m.logm)$, and it is guaranteed to provide an optimal packing due to the bottom-up approach used to compute the optimal cost of packing [4].

C. The Greedy NCP Algorithm

With our analysis of the special instances of the NCP problem , we note the similarity of the general problem instance to that of the two-dimensional bin packing, in which the computing and networking resources represent the dimensions of the bins. The increased complexity of the NCP problem is two-fold; the distanceS between the servers, i.e., the bins, and the non-additive property of the network dimension, since collocation might result in a reduced packing cost. Consequently, we define the G-NCP algorithm , inspired by bin packing, in which the smallest set of servers with minimum distances among them are chosen for the packing of the job's VMs.

In the G-NCP algorithm , as presented in Alg.1, the packing is performed in an iterative manner. In the first iteration of the algorithm, the server costs is computed according to Equation 1, and the server with the minimum cost is chosen to be packed with a cluster of VMs that doesn't violate its capacity. The mapped VMs are recorded, and the server is added to the chosen set of servers, with an updated effective capacity. In the next iteration, the server costs are recomputed, and the process is repeated until either all the job's VMs are mapped successfully, or none of the servers could accommodate any of the job's VMs in a given iteration.

Algorithm 1 The Greedy NCP Algorithm

```
Precondition: The job $J = (n, T)$ and physical servers $S = (D, C, B)$

function G-NCP($J, S$)
    Define list $R$ to record the mapping result of each VM
    $n' \leftarrow n$; $b \leftarrow$ average bandwidth in $T$; $matchFound \leftarrow false$
    while $n' > 0$ and not($matchFound$) do
        Compute cost of each server
        Sort according to cost
        for $i \leftarrow 1$ to $m$ do
            Get V; the most connected VMs to be packed on server i
            if $|V| > 0$ then
                $matchFound \leftarrow true$; $n' \leftarrow n' - |V|$
                Add allocation information to $R$
                Update $c_i$ and $b_i$
                break
    Return $R$ if matchFound, and $\phi$ otherwise
```

V. Performance Evaluation

In this section, we evaluate four aspects of the G-NCP algorithm via extensive simulations; the algorithm's efficacy, the algorithm's performance in an offline setting with varying properties, the benefit of reserving network bandwidth for jobs in an online setting, and the disadvantage of using abstractions.

A. Baseline Model

To evaluate the performance of our proposed algorithm in multiple settings, we designed a Java simulator that generates various problem instances, namely scenarios. In each scenario, we generate a problem instance with a service broker, a set of jobs requesting resources from that broker, and a resource allocation algorithm.

Service broker model. A service broker is defined by the number of available VM slots it holds, and the number of servers they are distributed onto. Once these values are defined, the available slots are uniformly distributed over the servers, and the available network resources, i.e., bandwidth, at each server is computed. In most of our experiments, a server's available bandwidth is a random value generated from the uniform distribution, $U(max/4, max)$. The value max is defined as the product of the average job size, the average bandwidth required on a path between any pair of VMs, and the average capacity of a server. Finally, the distance between the servers could be defined as equal, with a matrix of all 1's and a diagonal of 0's, or according to some tree-based topology similar as defined in [29].

Job communication model. The size of a job, i.e., the number of VMs it requires, is a uniform random variable which is an appropriate distribution for evaluating the effect of the job size as adopted in [14],. The communication pattern of a job can either be constant, clustered, or arbitrary. For arbitrary communication, the bandwidth demands in a job's traffic matrix are obtained from a Gaussian distribution as adopted in [29]. Clustered traffic is generated using a Gaussian distribution with high mean for intra-cluster traffic, and another Gaussian distribution with low mean for inter-cluster traffic.

VM allocation algorithms. In a scenario, VM allocation can be performed using the G-NCP algorithm as defined above, or using one of the three algorithms that we compare our algorithm against. The first two are baseline algorithms that perform basic VM placement without considering the job's traffic patterns; first-fit, and random allocation. The first-fit allocation algorithms is one of the most commonly used allocation algorithms, in which the VMs of a chosen job are collocated on servers according to their compute and bandwidth capacities, until all VMs are mapped to physical servers. On the other hand, the random allocation algorithm represents the worst case scenario of placing VMs in a datacenter, in which VMs are mapped to severs randomly, with no attempt of collocation. Finally, since no allocation algorithm exists, to the best of our knowledge, which considers the adequacy of a physical server to a specific job during allocation, we developed a greedy virtual network embedding algorithm inspired from [40] to evaluate the G-NCP algorithm against. In this embedding algorithm, we model the broker as substrate graph, with the nodes representing the available VM slots, and we compute the ranks of the nodes according to the stress on their links. Servers are ordered according to their rank, and VMs are ordered according to their bandwidth demand, and then we greedily map the VMs to the VM slots, with consideration to the capacity constraints of the servers hosting them.

Performance metrics. We evaluate the efficiency of an allocation using several system-centric metrics, which represent the effect of the allocation on the service broker. The *allocation cost*, as defined above, represents the cost of traffic flowing through the data center fabric, as defined by the service broker properties, and it's better minimized. The *server utilization* represents the ratio between the number of used VM slots, and the total number of VM slots originally offered by the service broker, and it's better maximized. Moreover, for online experiments, we measure the *response time*, which represents the average time a job spends in the system from arrival until completion at steady state, and the *throughput*, which represents the average number of jobs served per unit time at steady state.

B. Single-Job Experiments

To evaluate the efficacy of our greedy heuristic, we need to compare its objective value, the allocation cost, to that of the optimal solution. Since we have proven the hardness of the problem and its corresponding ILP formulation, it's infeasible to obtain optimal solutions for larger instances to compare against. Therefore, we resort to generating problem instances with a known optimal solution by construction, as done in [21]. To guarantee a correct optimal construction, we assume equal distances between all servers, and we eliminate all bandwidth constraints over the servers, i.e, allow unlimited bandwidth to be used between any two servers. Communication between any two VMs is never constrained, with 0 cost within the same server, and a constant cost of 1 if performed across multiple servers. We set up the service broker model in that way, to ensure that the allocation cost only depends on the traffic demands of the job.

In this set of experiments, we generate a service broker with 400 available computing slots, to be uniformly distributed over some number of physical servers, which differ according to the scenario to be simulated. As mentioned above, there is infinite bandwidth available at each server, and the distance matrix contains all 1's with a diagonal of 0's. We generate a single job to be allocated resources, with varying size starting at 2 VMs, up to 400 VMs. The job's traffic matrix is generated after the job's VMs are placed on the servers, to guarantee optimal allocation cost. We start by sorting the servers in decreasing order according to their computing capacity, and placing the job's VMs using the first-fit algorithm. Then, we generate the traffic matrices using one of two methods; Gaussian optimal, and sparse optimal.

In the method of Gaussian optimal construction, we create two Gaussian distributions with different means, and the same variance, with a guarantee that the minimum value generated by the high-mean distribution is always higher than the maximum value generated by the low-mean distributio. For every pair of VMs, we generate a random traffic value of from the high-mean distribution if they are collocated, and from the low-mean distribution otherwise. In the method of sparse optimal construction, which represents a worst case scenario of the job's communication model, we define two values for the traffic demand; a minimum and a maximum value. Then, for every pair of VMs, their requested traffic is maximum if they are collocated, and is minimum otherwise.

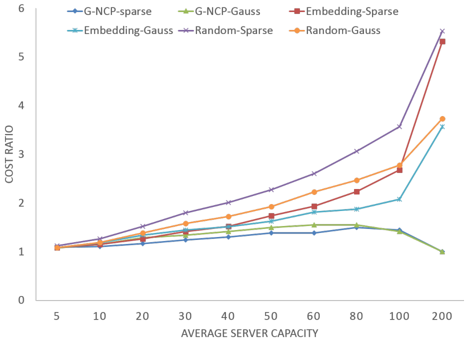
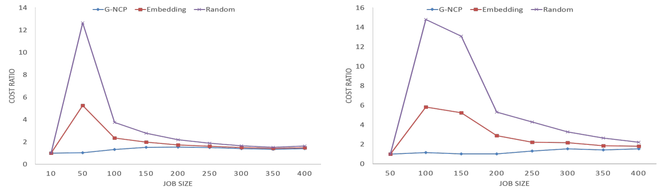


Fig. 3: Cost ratio improves with larger cluster sizes.

Traffic generated by both approaches corresponds to the clustered communication model, in which cluster sizes depend on the average capacity of the servers. Moreover, jobs are allocated their optimal resources using a decreasing first fit approach, thus it's excluded from our experiments below, in which evaluate the effect of the server capacity, and the job size on the efficacy of the algorithm. For each scenario within an experiment, we compute the ratio between the allocation cost of our proposed G-NCP algorithm , and the optimally constructed allocation cost, and similarly for the greedy, and the random algorithms. The results shown below are the average of the values obtained from running each scenario 40 times, each with high mean / maximum value of 0.7, low mean / minimum value of 0.1, and a standard deviation of 0.2.

In the first experiment, we evaluate the effect of the server capacity, i.e., the average number of collocated VM slots within a single server, on the efficacy of the algorithms. In each scenario, we generate a single job with a size of 400 VMs, with traffic demands that are either Gaussian or sparse, as defined above. The results in Fig.3 represent the allocation cost ratio with increased server capacity, i.e., with decreased number of servers over which the 400 VM slots are distributed. For scenarios with low server capacity, all algorithms provide almost-optimal allocation costs, which is a direct result of the small sizes of the clusters of communicating VMs within the job that result in less communication cost. As the average server capacity increases, the cluster sizes increase, creating more structured communication patterns among the VMs, and higher allocation costs. The cost ratio of the G-NCP algorithm increases at a very low rate because it considers the internal structure of the job during allocation, which is evident in the almost optimal cost ratio at the highest values of server capacity.

In the second experiment, we evaluate the effect of the job size on the efficacy of the algorithms. In each scenario, we generate a single job with varying size, and Gaussian-based optimally constructed traffic matrix. We show the experiment results for scenarios with servers with an average capacity of 20 VM slots each in Fig.4a, and an average capacity



(a) Capacity = 20 (b) Capacity = 40

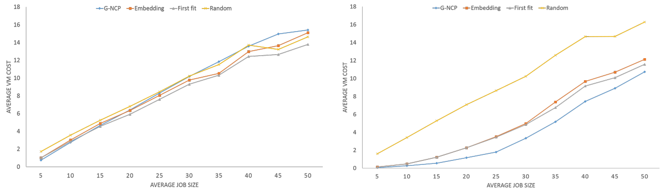
Fig. 4: Cost ratio is unaffected by the number of clusters in a job.

of 40 VM slots in Fig.4b. The cost ratio of the random and greedy algorithms reaches the worst cost ratio when job sizes are double that of the average server capacity, and then decreases with increased job size. This occurs because the number of clusters within the job increases with its size, given a fixed server capacity, which reduces the chance of deviating from the true allocation. Meanwhile, the cost ratio of the G-NCP algorithm is consistently lower than that of the other algorithms, and is almost unaffected by the job size.

C. Batch of Jobs Experiments

In the next set of experiments, we evaluate the performance of the allocation algorithms when applied to a batch of jobs. To apply these algorithms on a batch of jobs, with varying sizes and traffic demands, we sort the jobs according to their average traffic demand, and consider one job at a time for allocation. For each job, one of the algorithms is used to find an allocation. If one is found, the job's demanded resources are reserved, and the service broker properties are updated. If no solution is found, the job is considered rejected, and it is not allocated any resources.

1) Packing in G-NCP algorithm : In the first experiment, the importance of choosing servers that increase the chance of collocating the job's VMs is highlighted. Jobs are created with random sizes generated by a uniform distribution with an average varying between 5 and 50 VMs, and an arbitrary communication model with a random mean between $[0, 1]$, and a standard deviation of 0.2. The average allocation costs per VM achieved by the algorithms for a service broker with 400 VM slots, and an average server capacity of 6, and 10 VM slots are shown in Fig.5. In Fig.5a, the allocation cost of all algorithms approach that of the random due to the small server capacity, which doesn't enable enough collocation to minimize the allocation cost. This is confirmed in Fig.5b, in which the server capacity is increased by a small factor, allowing for better collocation for all algorithms. Moreover, this increased capacity exemplifies the G-NCP algorithm 's capability of choosing better servers for allocation, as evident by its lower costs.



(a) Capacity = 6 (b) Capacity = 10

Fig. 5: Even for arbitrary traffic, server choice improves the allocation cost.

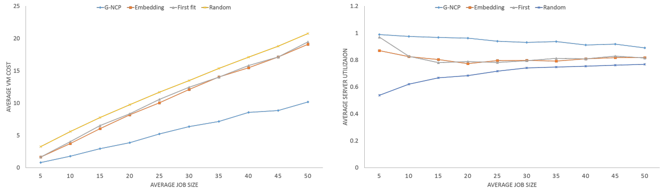


Fig. 6: The G-NCP algorithm is superior for jobs with structured communication models.

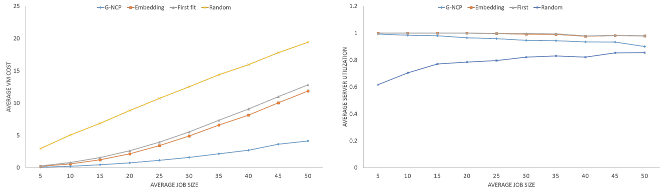


Fig. 7: The efficiency of the G-NCP algorithm improves with higher server capacities.

In the second experiment, the superiority of the G-NCP algorithm with jobs that have some structure in their communication model is highlighted. Jobs are created with random sizes generated by a uniform distribution with an average varying between 5 and 50 VMs, and a clustered communication model is generated using two Gaussian distributions, one with a mean of 2.0 for intra-cluster communication, and another with a mean of 0.5 for inter-cluster communication, and both with a standard deviation of 0.5. The results shown in Fig.6, 7 are for a service broker with 400 VM slots, and an average server capacity of 6 and 10 VM slots respectively. The efficiency of the G-NCP algorithm over the other algorithms is a result of our server packing approach, in which we start by choosing the most cost-efficient server according to the job demands, and then perform the VM packing according to the server's capacity. This approach results in choice of a smaller number of servers for allocation, while clustering the job's VMs in a way that minimizes the cross communication between these servers, resulting in a much improved allocation cost. Moreover, this superiority is highlighted by the average utilization of the service broker's resources, and an even more improved cost with higher server capacities.

2) Effect of traffic demand: In the next offline experiment, we evaluate the effect of the variation of traffic demand among VMs on the allocation performance. Jobs are created with random sizes generated by a uniform distribution with an average of 30, and their traffic matrices are generated using a Gaussian distribution with a random mean, and a standard deviation that varies between of 0.1, and 2.0. The allocation results for a service broker with 400 VM slots, and an average server capacity of 6 VM slots, are shown in Fig.8. As mentioned above, allocation cost of jobs with

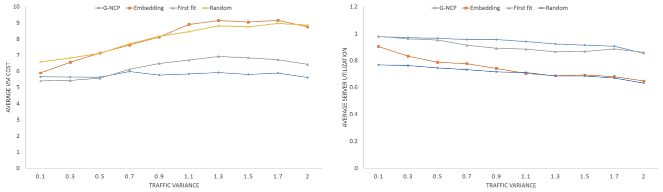


Fig. 8: Increased traffic variance enables better allocation.

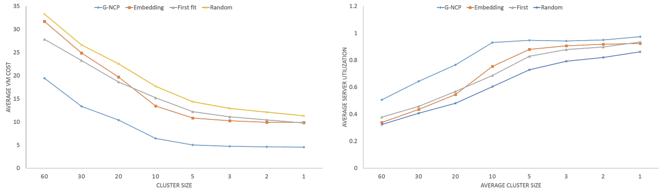


Fig. 9: The G-NCP algorithm is always superior for allocation jobs with structured communication patterns.

arbitrary communication models with small variance on servers with small capacities usually isn't affected by the nature of the allocation algorithm. When the variance is increased in this experiment, the communication model exhibits some structure, which allows the G-NCP algorithm to decide on better allocations.

Since the superiority of the G-NCP algorithm is better observed for jobs with communication patterns, we perform this third offline experiment, in which jobs have clustered communication patterns. The allocation results for a service broker with 400 VM slots, an average server capacity of 6 VM slots, and jobs with cluster sizes varying between 60, i.e., high communication among all VMs, and 1, i.e, low communication among all VMs, are shown in Fig.9. We note that the decreased allocation cost trend for all algorithms is attributed to the decreased cluster size, which minimizes the amount of bandwidth required by the job in general.

D. Online Experiments

In this next set of experiments, we aim to demonstrate the advantages of explicit bandwidth reservation for jobs in an online setting, as that observed in commercial cloud data center providers. In an online setting, jobs arrive to the system at some rate requesting resources from the service broker, and they are allocated physical resources according to the admission control policy adopted by the provider. In the next set of experiments, we compare between two admission control policies; network-oblivious and network-aware. Network-oblivious admission control is the policy currently adopted by commercial data center providers, in which only a job's computing requirements are considered upon the job's arrival, and physical resources are allocated to the job regardless of its network demand. In such a model of admission control, a decreasing first-fit allocation algorithm is usually applied, to increase the locality of the allocation. On the other hand, in network-aware admission control, both the job's computing and traffic demands are considered upon the job's arrival, and the admission of a job is only allowed if both the computing and network bandwidth can be reserved for the job. In such a model of admission control, a network-aware allocation algorithm is required, and we use the G-NCP algorithm in our experiments.

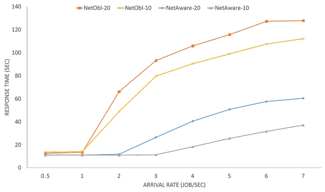


Fig. 10: Network aware admission provides better response time.

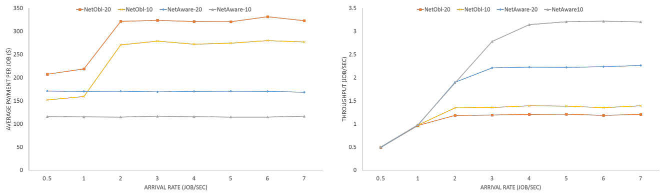


Fig. 11: Jobs are charged a constant amount of money, with improved system capacity.

In all of our online experiments, we generate a service broker with 400 VM slots with average server capacity of 10 VMs, and jobs have clustered communication models with a high mean of 2.0, a low mean of 0.5, and a standard deviation of 0.2. In the first online experiment, we evaluate the effect of the jobs' arrival rate on the system's performance, in terms of response time and throughput, as well as its effect on the monetary payment to be paid by the job's owner. For the purposes of this experiment, we define a job's payment as the product of the number of VMs required by the job, and the amount of time that the job uses these VMs. Since network-aware admission guarantees predictable execution of jobs, we observe how their payments are constant, depending only on the size of the job, regardless of the load on the system. Unlike in the network-oblivious admission model, in which the job's payment increases until the network gets saturated as the system reaches its maximum capacity. Moreover, the results indicate that network-aware admission leads to higher throughput, consequently an increased system capacity, which depends on the average job size and not the job's network bandwidth demand.

E. Effect of Inaccurate Inference on Performance

So far, in all of our previous experiments, we assumed that the input of the allocation service, in terms of the service broker's bandwidth capacities, and the job's bandwidth demands, are accurate, which might not be the case in a real system. In the next couple of experiments, we study the effect of using bandwidth capacity values as estimated by an inference service, as opposed to using their corresponding accurate values. We generate a data center with 16 servers, each with 30 VM slots, connected in a fat tree topology, as show in Fig.2b. In such a topology, all network links have a bandwidth of 1024 Mbps, and there is a single path between servers connected to the same edge switch, two paths between

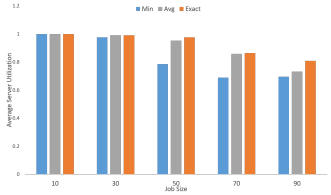


Fig. 12: The effect of abstraction on system utilization.

servers connected to the same pod, and four paths between servers connected to different pods. To emulate the state of the data center at about 50% utilization, we generate jobs with arbitrary communication patterns to fill all 480 VM slots in the data center, and then we remove half of them. For each job, the VMs are mapped to random servers, and the data between any pair of VMs is routed through an available path, chosen at random, between the servers accommodating them. When a job is removed, the VM slots it was using are made available, and its traffic demand is removed from the fabric.

To emulate the behavior of the inference service, we consider the available bandwidth capacity of a server to be defined as the maximum bandwidth measured between that server and every other server accessed by the service broker. We assume that the inference service performs multiple experiments to measure the available bandwidth between pairs of servers, and that it estimates the bandwidth between a pair of servers according to one of two policies; the minimum, or the average observed bandwidth. To compare the effectiveness of both estimation approaches, we attempt to allocate resources for a batch of jobs with arbitrary traffic, and we compute the utilization of the compute resources of the service broker. The results shown in Fig.12, indicate that using the minimum observed bandwidth leads to the under-utilization of the service broker resources, especially for large job sizes, as opposed to using the average observed bandwidth.

Although average observed bandwidth estimates provide better resource utilization, they lead to situations in which the reserved bandwidth, during allocation, is more than what is actually provided by the data center fabric. To examine the effect of this over-reservation, we evaluate the performance of network-aware admission control with bandwidth capacities estimated using the average observed bandwidth. In this experiment, the server bandwidth capacity is computed at the beginning of each scenario using average values, and never changes during the simulation. Meanwhile, the true available bandwidth varies through out the simulation, by changing the paths used to route data between servers. The results in Fig.13 represent the average response time, and cost per job with an increased load on the system. Jobs stay longer than expected in the system, but their payments and response time are still not affected by the load on the system, and is only affected by the error in estimated bandwidth. Leading to a slightly degraded performance as compared to accurate values, but a significantly better performance than network-oblivious approaches.

VI. Conclusion

In this paper, we presented two growing trends in cloud platforms; service brokerage, and providing predictability guarantees for data-intensive jobs, and argued the need to define a resource allocation model that can satisfy both trends. To that extent, we propose the first brokerage model for providing predictability guarantees on cloud platforms, which does not assume any control over either the cloud provider, or the customer, and present our view of a framework based on that model. We focused on the resource allocation component of that framework, and proposed the Network-Constrained Packing problem, prove its hardness, and exploit its resemblance to bin packing to develop efficient exact algorithms for two of its special instances, and a greedy heuristic for the general problem instance. Evaluation results for the proposed G-NCP algorithm confirm that it performs better with jobs with structured communication models, and that its performance is not affected by the job size, or the average server capacity. Moreover, our evaluations exemplify the advantage of performing network-aware admission control, even with uncertain inference of the available resources. Our current work involves the analysis of the other special cases of the NCP problem to identify more efficient algorithms for them, and the development of better approaches for clustering the job's components according to the properties of the server to be packed in. In our future work, we plan to study the problem of scheduling and allocating resources to a batch of brokered workloads, as well as the design and implement the other elements of our proposed brokerage framework

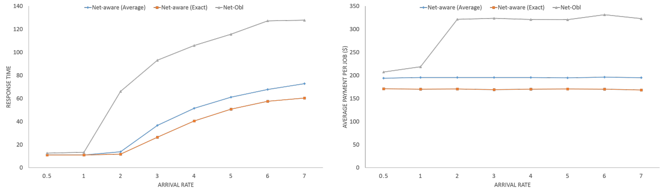


Fig. 13: Admission with inaccurate estimations is still better that network-oblivious admission.

Acknowledgments

This work is supported by NSF CISE CNS Award # 1347522, # 1239021, # 1012798.

References

[1] I. Amazon Web Services. Amazon EC2.

[2] H. Ballani, P. Costa, T. Karagiannis, and A. Rowstron. Towards predictable datacenter networks. *ACM SIGCOMM Computer Communication Review*, 41(4):242, Oct. 2011.

[3] O. G. Ballani H, Jang K, Karagiannis T, Kim C, Gunawardena D. Chatty Tenants and the Cloud Network Sharing Problem. In *NSDI*, 2013.

[4] C. Bassem and A. Bestavros. Network-Constrained Packing of Brokered Workloads in Virtualized Environments. Technical report, Boston University, 2014.

[5] D. Battré, N. Frejnik, S. Goel, O. Kao, and D. Warneke. Evaluation of network topology inference in opaque compute clouds through end-to-end measurements. In *Cloud Computing (CLOUD), 2011 IEEE International Conference on*, pages 17–24. IEEE, 2011.

[6] A. Bestavros and O. Krieger. Toward an open cloud marketplace: Vision and first steps. *IEEE Internet Computing*, 18(1):72–77, 2014.

[7] X. Cheng, S. Su, Z. Zhang, K. Shuang, F. Yang, Y. Luo, and J. Wang. Virtual network embedding through topology awareness and optimization. *Computer Networks*, 56(6):1797–1813, Apr. 2012.

[8] M. Chowdhury, M. R. Rahman, and R. Boutaba. ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping. *IEEE/ACM Transactions on Networking*, 20(1):206–219, Feb. 2012.

[9] M. Chowdhury and I. Stoica. Coflow: A networking abstraction for cluster applications. In *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pages 31–36. ACM, 2012.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[11] F. Esposito, I. Matta, and V. Ishakian. Slice embedding solutions for distributed service architectures. *ACM Comput. Surv.*, 46(1):6:1–6:29, July 2013.

[12] A. Fischer, J. F. Botero, M. T. Beck, H. de Meer, and X. Hesselbach. Virtual Network Embedding: A Survey. *IEEE Communications Surveys & Tutorials*, 15(4):1888–1906, Jan. 2013.

[13] C. Fuerst, S. Schmid, and A. Feldmann. Virtual network embedding with collocation: Benefits and limitations of pre-clustering. In *2013 IEEE 2nd International Conference on Cloud Networking (CloudNet)*, pages 91–98. IEEE, Nov. 2013.

[14] I. Giurgiu, C. Castillo, A. Tantawi, and M. Steinder. Enabling efficient placement of virtual infrastructures in the cloud. In *Proceedings of the 13th International Middleware Conference*, pages 332–353. Springer-Verlag New York, Inc., 2012.

[15] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta. Vl2: A scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*, SIGCOMM '09, pages 51–62, New York, NY, USA, 2009. ACM.

[16] C. Guo, G. Lu, H. J. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang. SecondNet. In *Proceedings of the 6th International COnference on - Co-NEXT '10*, page 1, New York, New York, USA, 2010. ACM Press.

[17] Y. Guo, A. L. Stolyar, and A. Walid. Shadow-routing based dynamic algorithms for virtual machine placement in a network cloud. In *2013 Proceedings IEEE INFOCOM*, pages 620–628. IEEE, Apr. 2013.

[18] I. Houidi, M. Mechtri, W. Louati, and D. Zeghlache. Cloud service delivery across multiple cloud platforms. In *Services Computing (SCC), 2011 IEEE International Conference on*, pages 741–742. IEEE, 2011.

[19] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, and D. H. Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *Parallel and Distributed Systems, IEEE Transactions on*, 22(6):931–945, 2011.

[20] M. Isard, M. Budiu, Y. Yu, A. Birrell, and D. Fetterly. Dryad: Distributed data-parallel programs from sequential building blocks. In *Proceedings of the 2Nd ACM SIGOPS/EuroSys European Conference on Computer Systems 2007*, EuroSys '07, pages 59–72, New York, NY, USA, 2007. ACM.

[21] J. Ishakian, Vatche and Sweha, Raymond and Bestavros, Azer and Appavoo. CloudPack* Exploiting Workload Flexibility Through Rational Pricing. In *Proceedings of the 13th International Middleware Conference*, pages 374—–393, 2012.

[22] K. R. Jackson, L. Ramakrishnan, K. Muriki, S. Canon, S. Cholia, J. Shalf, H. J. Wasserman, and N. J. Wright. Performance analysis of high performance computing applications on the amazon web services cloud. In *Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on*, pages 159–168. IEEE, 2010.

[23] D. S. Johnson. *Near-optimal bin packing algorithms*. PhD thesis, Massachusetts Institute of Technology, 1973.

[24] R. M. Karp. *Reducibility among combinatorial problems*. Springer, 1972.

[25] K. LaCurts, S. Deng, A. Goyal, and H. Balakrishnan. Choreo. In *Proceedings of the 2013 conference on Internet measurement conference - IMC '13*, pages 191–204, New York, New York, USA, 2013. ACM Press.

[26] Li, Xin, Jie Wu, Shaojie Tang and S. Lu. Lets Stay Together: Towards Traffic Aware Virtual Machine Placement in Data Centers. In *INFOCOM*, 2014.

[27] J. Lischka and H. Karl. A virtual network mapping algorithm based on subgraph isomorphism detection. In *Proceedings of the 1st ACM workshop on Virtualized infrastructure systems and architectures - VISA '09*, page 81, New York, New York, USA, 2009. ACM Press.

[28] J. Londoño, A. Bestavros, and S.-H. Teng. Colocation games: And their application to distributed resource management. In *Proceedings of the 2009 Conference on Hot Topics in Cloud Computing*, HotCloud'09, Berkeley, CA, USA, 2009. USENIX Association.

[29] X. Meng, V. Pappas, and L. Zhang. Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement. In *2010 Proceedings IEEE INFOCOM*, pages 1–9. IEEE, Mar. 2010.

[30] A. Middleton. Data-intensive technologies for cloud computing. In B. Furht and A. Escalante, editors, *Handbook of Cloud Computing*, pages 83–136. Springer US, 2010.

[31] R. Niranjan Mysore, A. Pamboris, N. Farrington, N. Huang, P. Miri, S. Radhakrishnan, V. Subramanya, and A. Vahdat. Portland: A scalable fault-tolerant layer 2 data center network fabric. *SIGCOMM Comput. Commun. Rev.*, 39(4):39–50, Aug. 2009.

[32] P. S. Pacheco. *Parallel programming with MPI*. Morgan Kaufmann, 1997.

[33] I. Raicu, I. T. Foster, and Y. Zhao. Many-task computing for grids and supercomputers. In *Many-Task Computing on Grids and Supercomputers, 2008. MTAGS 2008. Workshop on*, pages 1–11. IEEE, 2008.

[34] A. Thusoo, J. S. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff, and R. Murthy. Hive: a warehousing solution over a map-reduce framework. *Proceedings of the VLDB Endowment*, 2(2):1626–1629, 2009.

[35] G. Wang and T. Ng. The impact of virtualization on network performance of amazon ec2 data center. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–9, March 2010.

[36] T. Wood, L. Cherkasova, K. Ozonat, and P. Shenoy. Profiling and modeling resource usage of virtualized applications. In *Proceedings of the 9th ACM/IFIP/USENIX International Conference on Middleware*, Middleware '08, pages 366–387, New York, NY, USA, 2008. Springer-Verlag New York, Inc.

[37] D. Xie, N. Ding, Y. C. Hu, and R. Kompella. The only constant is change. *ACM SIGCOMM Computer Communication Review*, 42(4):199, Sept. 2012.

[38] K. You, B. Tang, and F. Ding. Near-optimal virtual machine placement with product traffic pattern in data centers. In *2013 IEEE International Conference on Communications (ICC)*, pages 3705–3709. IEEE, June 2013.

[39] M. Yu, Y. Yi, J. Rexford, and M. Chiang. Rethinking virtual network embedding. *ACM SIGCOMM Computer Communication Review*, 38(2):17, Mar. 2008.

[40] Y. Zhu and M. Ammar. Algorithms for Assigning Substrate Network Resources to Virtual Network Components. In *Proceedings IEEE INFOCOM 2006. 25TH IEEE International Conference on Computer Communications*, pages 1–12. IEEE, 2006.