Cost-Efficient Virtual Server Provisioning and Selection in Distributed Data Centers

Jielong Xu, Jian Tang, Brendan Mumey, Weiyi Zhang, Kevin Kwiat and Charles Kamhoua

***Abstract*—In this paper, we study a Virtual Server Provisioning and Selection (VSPS) problem in distributed Data Centers (DCs) with the objective of minimizing the total operational cost while meeting the service response time requirement. We aim to develop general algorithms for the VSPS problem without assuming a particular queueing model for service processing in each DC. First, we present a Mixed Integer Linear Programming (MILP) formulation. Then we present a 3-step optimization framework, under which we develop a polynomial-time $\ln(N)$-approximation algorithm (where N is the number of clients) along with a post-optimization procedure for performance improvement. We also show this problem is NP-hard to approximate and is not possible to obtain a better approximation ratio unless NP has TIME$(n^{O(\log \log n)})$ deterministic time algorithms. In addition, we present an effective heuristic algorithm that jointly obtains the VS provisioning and selection solutions. Extensive simulation results are presented to justify effectiveness of the proposed algorithms.**

***Index Terms*—Cloud Computing, Distributed Data Centers, Server Provisioning, Server Selection.**

I. Introduction

In this paper, we focus on a large-scale cloud computing system with multiple Data Centers (DCs) and clients (which could be mapping nodes [19] or traditional DNS servers) distributed in different locations. Note that a *client* is not an end-user but a node that aggregates service requests from end users and dispatches them to a subset of DCs. In a DC, a service request is handled by a *Virtual Server* (VS), which consists of one or multiple application programs (corresponding to a service) that are hosted by one or multiple related VMs. We are interested in a Virtual Server Provisioning and Selection (VSPS) problem, i.e., determine how many VSs to be provisioned (VS provisioning) in each DC and how to dispatch service requests to VSs in DCs (VS selection) for a service. The objective is to minimize the total cost of operating VSs subject to the service response time requirement imposed by the Service Level Agreement (SLA). The service response time of a request consists of two parts: 1) the communication delay (i.e., the Round Trip Time (RTT)) between the client and the DC the request is dispatched to; and 2) the queueing delay in that DC. Operating a VS in different DCs (at different locations) might be different due to various location-dependent factors such as electricity cost [16], which makes the problem particularly interesting. In two closely related works [16], [17], the simple M/M/c model [12] was assumed for queueing in each DC, which is only suitable for simple single-tier services but not applicable for multi-tier services such as web services. We, however, aim to design general (*model-independent*) algorithms to solve this problem, which are not restricted to a particular queueing model for service processing in a DC. The only assumption we make here is that the queueing delay is given by a function of the total request arrival rate, the number of VSs in a DC and the service rate of each VS, whose values can be obtained by either using a proper queueing model (designed particularly for that kind of service) or a profile-based approach. But this makes the problem and algorithm design very challenging due to lack of a simple closed-form equation for calculating queueing delay. We summarize our contributions in the following:

- We formally define the VSPS problem and present an Mixed Integer Linear Programming (MILP) formulation to provide optimal solutions.
- To solve the VSPS problem, we present a general optimization framework to guide algorithm design, under which we develop a polynomial-time $\ln(N)$-approximation algorithm (where N is the number of clients) along with a post-optimization procedure for performance improvement. We also show this problem is NP-hard to approximate and is not possible to obtain a better approximation ratio unless NP has TIME$(n^{O(\log \log n)})$ deterministic time algorithms. Although this statement is not as strong as $P = NP$, it is still considered unlikely at the time of writing.
- It has been shown by extensive simulation results based on data generated by both M/M/c model and a profile-based approach that the proposed algorithms work well for both cases and can produce close-to-optimal solutions.

To the best of our knowledge, we are the first to develop theoretically well-founded and practically useful algorithms for VSPS in geo-distributed DCs without assuming a particular queueing model for service processing in a DC.

II. Related Work

Server provisioning and selection problems have been studied in two closely related works [16], [17]. In [16], Rao *et al.* studied a server provisioning and selection problem with the objective of minimizing the total electricity cost in a multi-electricity-market environment with the assumption of M/M/c (multi-server) queueing model for each DC. They modeled the problem as a constrained mixed-integer programming problem and presented a polynomial-time heuristic algorithm. In a later work [17], they considered a similar problem and modeled the problem as a constrained mixed integer programming

Jielong Xu and Jian Tang are with the Department of Electrical Engineering and Computer Science at Syracuse University. Email: {jxu21, jtang02}@syr.edu. Brendan Mumey is with the Department of Computer Science at Montana State University. Weiyi Zhang is with AT&T Labs Research. Kevin Kwiat and Charles Kamhoua are with Air Force Research Lab (AFRL), Rome, NY. The paper has been approved for Public Release; Distribution Unlimited: 88ABW-20144044&20140826. This research is supported by NSF grant CNS-1443966. The information reported here does not reflect the position or the policy of the federal government. We would like to thank Dr. K. K. Ramakrishnan for his helpful suggestions and comments.

978-1-4673-6432-4/15/$31.00 ©2015 IEEE

problem based on the Generalized Benders Decomposition (GBD) technique.

In [19], the authors presented DONAR, a distributed system that can offload the burden of replica selection, while providing services with a sufficiently expressive interface for specifying mapping policies. DONAR uses a simple distributed algorithm to solve a server selection problem that jointly considers both client performance and server load. The commercial system, Akamai [2], has the same functionalities as DONAR but uses a centralized hierarchical stable-marriage algorithm for assigning clients to servers. In a recent paper [3], assuming that users specify their resource needs, such as the number of VMs needed for a large computational task, Alicherry and Lakshman developed an efficient 2-approximation algorithm for the optimal selection of DCs in a distributed cloud. In another recent work [1], Adnan *et al.* investigated how much workload to be executed in each DC and how much workload to be delayed and migrated to other DCs for energy saving while meeting deadlines. They presented an offline formulation for the geographical load balancing problem with dynamic deferral and gave online algorithms to determine assignment of workload to DCs and migration of workload between DCs in order to adapt with dynamic electricity price changes.

The differences between this work and these related works are summarized as follows: 1) We develop novel model-independent algorithms that can be used for different kinds of applications. However, two closely related works [16], [17] assumed the M/M/c queueing model at each data center, which cannot be applied to some popular services such as the multi-tier web service [7]. 2) The VSPS problem studied here addresses end-to-end delay constraint, which is *mathematically* different from those problems considered in [1], [2], [3], [19] 3) We conduct a comprehensive theoretical study for the computational complexity of the problem and present an approximation algorithm. However, most related works [2], [16], [17], [19] only presented heuristic algorithms without analyzing computational complexities of the problems or the worst-case performance of the proposed algorithms.

III. Problem Formulation

We consider a cloud computing system including N clients and M geo-distributed DCs connected via a WAN. In this system, each client aggregates service requests from a group of users in a common region (such as a city or a state). The service demand of a client can be fulfilled by one or multiple DCs. In a DC, a service is usually hosted by VSs. The more VSs a DC has, the higher the service rate it can support, but the higher the operational cost.

Usually, the Service Level Agreement (SLA) specifies a service response time threshold T for a service, which should not be violated by the cloud service provider. The service response time consists of two parts: the communication delay from a client to a DC and the queueing delay in the DC. For each client-DC pair (i, j), we assume that the communication delay t_{ij} can be measured and estimated/predicted by an existing traffic engineering method [14]. Note that in this work, communication delay is basically average end-to-end packet delay, which includes transmission delay, propagation delay and queueing delay on intermediate routers. Obviously, if $t_{ij} > T$, then it is not possible to use DC j to serve client i. Therefore, we can identify a set of (potentially) *usable* DCs for each client i. In a DC j, the queueing delay depends on the (service request) arrival rate λ_j, the number of VSs and the service rate μ_j of each VS at DC j. The arrival rate λ_j is the summation of service request rates distributed to DC j from those clients which choose DC j to serve. In closely related works [16], [17], the M/M/c queueing model was assumed for service processing and the corresponding formula was used to calculate queueing delay. However, it is known that the M/M/c queueing model cannot be applied to complicated multi-tier services such as web services [7]. In this work, we aim to develop a general method that is not restricted to any particular queueing model and can be applied to any service. Specifically, the only assumption we make here is that if the (service request) arrival rate λ_j, the number of VSs and the service rate μ_j of each VS at DC j are given, we can obtain the queueing delay. Equivalently, if the number of VSs, the service rate of each VS and a queuing delay bound are given, we can obtain the capacity of DC j, i.e., the maximum allowable service request arrival rate. The queuing delay bound can be given by $T - t_j^{\max}$, where $t_j^{\max}$ is the maximum communication delay of clients served by DC j. For different services, different methods can be applied to obtain such queueing delay and DC capacity. For example, for a single-tier service, the M/M/c queueing model can be applied; for a multi-tier web service, the queueing model proposed in [7] can be used; or a profile-based approach can be used to obtain them based on historical data collected from real systems. We do not want to restrict our work to a particular model or approach.

We are interested in knowing a VS provisioning solution that specifies how many VSs to provision in each DC as well as a VS selection solution which specify how to distribute service requests from each client to its usable DCs. A VS provisioning and selection solution is said to be *feasible* if the following conditions are satisfied: 1) For each client, all its service demand is fulfilled. 2) For each client, the (mean) service response time should not exceed the given threshold T. Now we are ready to define the optimization problem.

Definition 1 (VSPS): Given a cloud computing system including N clients and M geo-distributed DCs, the service demand of each client i d_i, a service response time threshold T, and communication delay for each client-DC pair t_{ij}, the **Virtual Server Provisioning and Selection (VSPS)** problem seeks a feasible VSPS solution such that the total cost of operating VSs $\sum_{j=1}^{M} c_j m_j$ is minimized, where c_j is the cost of operating a VS in DC j and m_j is the number of VSs provisioned in DC j.

Note that even for a common service, the cost of operating a VS in different DCs (at different locations) might be quite different due to various location-dependent factors such as electricity cost [16].

It is not trivial to provide a mathematical programming formation for the VSPS problem due to the lack of a closed-form equation for calculating queueing delay or capacity

of a DC. However, we find out that the VSPS problem can be formulated as an MILP problem. For the sake of MILP formulation, we need to find a bound M_j for the number of VSs in each DC j. This can be easily obtained by considering the worst case in which demands from all clients are distributed to DC j and calculating M_j accordingly. For each DC j, we need to sort all the clients in the ascending order of the corresponding communication delays t_{ij}. In this list, we call the client with the kth smallest communication delay, the kth *closest* client of DC j. The MILP formulation is presented as follows:

Unknown decision variables:

1) $\lambda_{ij} \geq 0$: The service request rate from client i at DC j .
2) $x_{kj}^{m_j} = \{0, 1\}$: $x_{kj}^{m_j} = 1$ if m_j VSs are provisioned in DC j to serve k closest clients; 0, otherwise.
3) $y_{ij} = \{0, 1\}$: $y_{ij} = 1$ if client i is served by DC j; 0, otherwise.

VSPS-MILP:

$$\min \sum_{j=1}^{M} c_j \Big(\sum_{k=1}^{N} \sum_{m_j=1}^{M_j} x_{kj}^{m_j} m_j\Big) \tag{1}$$

Subject to:

$$\sum_{j=1}^{M} \lambda_{ij} = d_i, \quad i \in \{1, \ldots, N\}, \tag{2}$$

$$\sum_{i=1}^{N} \lambda_{ij} \leq \sum_{k=1}^{N} \sum_{m_j=1}^{M_j} x_{kj}^{m_j} C_{kj}^{m_j}, \quad j \in \{1, \ldots, M\}; \tag{3}$$

$$\sum_{k=1}^{N} \sum_{m_j=1}^{M_j} x_{kj}^{m_j} \leq 1, \quad j \in \{1, \ldots, M\}; \tag{4}$$

$$y_{ij} = \sum_{k=(\pi^j)^{-1}(i)}^{N} \sum_{m_j=1}^{M_j} x_{kj}^{m_j}, \quad i \in \{1, \ldots, N\}, \; j \in \{1, \ldots, M\}; \tag{5}$$

$$0 \leq \lambda_{ij} \leq d_i y_{ij}, \quad i \in \{1, \ldots, N\}, \; j \in \{1, \ldots, M\}. \tag{6}$$

The objective (1) is to minimize the total operational cost. Constraints (2) ensure that the demand of every client is satisfied. In each DC, the total service request arrival rate should not exceed its capacity (which can be obtained if m_j and the clients it serves are given), which is guaranteed by constraints (3). Here, $C_{kj}^{m_j}$ is the capacity of DC j if m_j VSs are provisioned to serve k closest clients. According to the definition of variables $x_{kj}^{m_j}$, Equations (4) must hold. Constraints (5) establish a connection between y_{ij} and $x_{kj}^{m_j}$. Here $(\pi^j)^{-1}(i)$ gives the index of client i in DC j's client list which is sorted in the ascending order of communication delays t_{ij}. Finally, constraints (6) set bounds for λ_{ij}. Specifically, if client i is served by DC j, the demand distributed to DC j from client i should not exceed its total demand; otherwise, λ_{ij} should be 0. Note that we must consider a special case where no VS is provisioned in a DC j, then the values of all corresponding $x_{kj}^{m_j}$ ($m_j \in \{1, \cdots, M_j\}$) variables will be 0. In this case, we need to make sure that the values of all corresponding λ_{ij} ($i \in \{1, \ldots, N\}$) are forced to 0, which can be guaranteed by constraints (5) and (6) too.

IV. Optimization Framework

In this section, we first present a 3-step framework to guide algorithm design for the VSPS problem. Under this framework, we develop an approximation algorithm. Essentially, a VSPS problem consists of two subproblems: *VS provisioning and VS selection*. The VS provisioning problem is to determine how many VSs to provision in each DC. The VS selection subproblem can be further divided into two subproblems: *DC selection* (determine which subset of DCs are used to serve each client, or equivalently, determine which subset of clients are served by each DC) and *request distribution* (how to distribute demand from each client to the subset of DCs selected in the DC selection solution). We formally present the optimization framework in the following.

Algorithm 1 VSPS-Framework

Step 1	Compute a DC selection solution;
Step 2	Based on the DC selection solution, obtain a feasible VSPS solution;
Step 3	Improve the feasible solution by keep removing VSs and re-distributing demands of clients until it is no longer possible.

A. Approximation Algorithm

Following this framework, we first present a greedy algorithm based on the well-known Weighted Set-Cover (WSC) algorithm [20]. The main idea of the algorithm is to consider sets of clients that can reach each DC and weight them according to the cost of serving all the clients by that DC. We then employ the standard WSC algorithm to determine the set of clients that should use each DC. The algorithm is formally presented as Algorithm 2. In the algorithm presentation, we use the same notations as those in VSPS-MILP. Note that the algorithm does not explicitly compute the $\langle y_{ij} \rangle$ values, since they can be calculated directly from the $\langle x_{kj}^{m_j}, \lambda_{ij} \rangle$ values returned using Equations (5).

The algorithm works as follows: In Step 1, it first sorts the clients at each DC j in the ascending order of its communication delays. For each possible prefix of the sorted list (i.e., the first k clients), it creates a set S_{kj}. The algorithm then assigns this set a weight value $w(S_{kj})$ equal to the cost of the number of VSs necessary to support 100% of the demands from all of the clients in S_{kj}. This creates an instance of the WSC problem, since we want to serve each client with one or multiple DCs. In Step 2, the standard greedy approach is used to find a solution to this WSC instance: the set that provides the lowest cost to number of uncovered clients ratio is repeatedly added to the cover $\mathcal{C}$. In Step 3, we convert $\mathcal{C}$ back to a VSPS solution: for each DC j, we determine the set $S_{kj} \in \mathcal{C}$ with the

Algorithm 2 VSPS-WSC

Step 1 $\mathcal{A} := \emptyset$, $\mathcal{C} := \emptyset$, $U := \{1, \ldots, N\}$;
for $j := 1$ **to** M
Let π^j sort the clients by ascending t_{ij};
for $k := 1$ **to** N
$S_{kj} := \{\pi^j(1), \ldots, \pi^j(k)\}$;
$m_{kj} := \text{argmin}_{m \geq 1} \sum_{i \in S_{kj}} d_i \leq C^m_{kj}$;
$\mathcal{A} := \mathcal{A} \cup (k, j)$;
Assign weight $w(S_{kj}) := c_j m_{kj}$;
endfor
endfor
Step 2 **while** $(U \neq \emptyset)$
Compute $(k^*, j^*) := \text{argmin}_{(k,j) \in \mathcal{A}} \frac{w(S_{kj})}{|S_{kj} \cap U|}$;
Add $S_{j^*k^*}$ to $\mathcal{C}$;
$U := U \setminus S_{j^*k^*}$;
endwhile
Step 3 Set all $x^{m_j}_{kj}, \lambda_{ij} := 0$;
Reset $U := \{1, \ldots, N\}$;
for $j := 1$ **to** M
$K := \{k : S_{kj} \in \mathcal{C}\}$;
if $(K \neq \emptyset)$
$k^* := \max(K)$;
$k' := \text{argmin}_{k \in K} S_{kj} \cap U := S_{kj^*} \cap U$;
$U_j := S_{k'j} \cap U$;
$m_j := \text{argmin}_{m \geq 1} \sum_{i \in U_j} d_i \leq C^m_{k'j}$;
$x^{m_j}_{k'j} := 1$;
for $i \in U_j$
$\lambda_{ij} := d_i$;
endfor
$U := U \setminus U_j$;
endif
endfor
Step 4 **return** $\langle x^{m_j}_{kj}, \lambda_{ij} \rangle$.

largest k value. Rather than immediately chose this k value for DC j, we see if we can choose a smaller value k' and still cover the same set of uncovered clients. We then set m_j to the number of VSs needed to handle the demands from the uncovered clients in $S_{k'j}$ and let these clients send all of their demands to DC j. The $x^{m_j}_{kj}$ variables are determined accordingly.

Next, we show the worst-case performance guarantee that can be provided by the VSPS-WSC algorithm. The approximation ratio depends on two additional parameters of the input:

- $\Delta = \dfrac{\max_i d_i}{\min_i d_i}$: the ratio of the largest to the smallest demand.
- $K = \min_{\text{optimal } \langle x^{*m_j}_{kj} \rangle} \max\{k : x^{*m_j}_{kj} = 1\}$: the minimum *furthest* client distance (in terms of k) among all optimal solutions $\langle x^{*m_j}_{kj} \rangle$.

Theorem 1: The VSPS-WSC algorithm is a $2(1 + (K - 1)\Delta) \ln N$-approximation algorithm for the VSPS problem, provided that each DC's capacity is linear in the number of its VSs.

Proof: Let $\langle x^{*m_j}_{kj}, \lambda^*_{ij} \rangle$ be an optimal solution to the MILP with minimum possible K value (defined above). Our goal is to relate this optimal VSPS solution to a solution of the associated instance of the WSC problem created by Algorithm 2. We do this by first relaxing the m_j values from $\langle x^{*m_j}_{kj}, \lambda^*_{ij} \rangle$ to be real valued. By doing so, the m_j values may potentially be lowered to a non-integer value. Here, we assume that each DC's capacity (maximum allowable arrival rate) is linear in the number of its VSs m_j, i.e., $\alpha_{kj} m_j - \beta_{kj}$. So $m_j = (\sum_i \lambda^*_{ij} + \beta_{k(j)j})/\alpha_{k(j)j}$, where $k(j)$ is such that $x^{*m_j}_{k(j)j} = 1$. The cost of the solution is reduced to $\sum_j c_j m_j$. Next, let

$$l^*(i) = \underset{j:\pi^j(i) \leq k(j)}{\text{argmin}} \alpha_{k(j)j}/c_j$$

be the DC available to client i given the $k(j)$ values, which provides the most additional capacity per unit cost to i. We will reroute all of client i's demand to DC $l^*(i)$. We observe that this cannot increase the cost of the solution. After this is done, each client i sends all of its demand d_i only to DC $l^*(i)$ (that may employ a non-integer number of VSs) at a total cost at most cost$(\langle x^{*m_j}_{kj}, \lambda^*_{ij} \rangle)$. Let $J^* = \{j : \exists i \; l^*(i) = j\}$. Each $j \in J^*$ now needs to allocate $m_j = (\sum_{i:l^*(i)=j} d_i + \beta_{k(j)j})/\alpha_{k(j)j}$ VSs. We modify this solution further so to convert into a solution to the WSC instance considered by the algorithm: For each $j \in J^*$, consider the clients i such that $\pi^j(i) \leq k(j)$: they may or may not send any demands to DC j at this point, but we shall now provision VSs for DC j so as to accommodate all of their collective demands by setting $m'_j = (\sum_{i:\pi^j(i) \leq k(j)} d_i + \beta_{k(j)j})/\alpha_{k(j)j}$. This will increase the cost at each DC by the ratio:

$$\begin{aligned}
\frac{m'_j}{m_j} &= \frac{\sum_{i:\pi^j(i) \leq k(j)} d_i + \beta_{k(j)j}}{\sum_{i:l^*(i)=j} d_i + \beta_{k(j)j}} \\
&\leq \frac{\sum_{i:\pi^j(i) \leq k(j)} d_i}{\sum_{i:l^*(i)=j} d_i} \\
&\leq \frac{\sum_{i:l^*(i)=j} d_i + \sum_{i:\pi^j(i) \leq k(j), l^*(i) \neq j} d_i}{\sum_{i:l^*(i)=j} d_i} \\
&\leq 1 + (K-1)\Delta.
\end{aligned}$$

So, the total cost has now increased by a factor of $1+(K-1)\Delta$. Since, the m_j values are still non-integer, we will round them up to the nearest integer value, $\lceil m_j \rceil$. This at most doubles the cost and makes it agree with the cost assigned to the set $S_{k(j)j}$ in the WSC instance considered by the algorithm. We observe this process has now created a solution $\mathcal{C} = \{S_{k(j)j} : j \in J^*\}$ to the WSC instance, since each client i will be covered by a set in this collection. We have

$$\text{weight}(\mathcal{C}) \leq 2(1 + (K-1)\Delta)\text{cost}(\langle x^{*m_j}_{kj}, \lambda^*_{ij} \rangle).$$

Let $\mathcal{C}^*$ be the optimal solution to the WSC instance and let $\mathcal{C}^{\text{alg}}$ be the solution found by the algorithm. Note that the corresponding VSPS solution $\langle x^{m_j}_{kj}, \lambda_{ij} \rangle$ produced has a cost at most weight$(\mathcal{C}^{\text{alg}})$,

$$\text{cost}(\langle x^{m_j}_{kj}, \lambda_{ij} \rangle) \leq \text{weight}(\mathcal{C}^{\text{alg}}).$$

This is because Step 4 may not need to use all of the sets in $\mathcal{C}^{\text{alg}}$. Since we employ the standard greedy algorithm for

WSC, we have

$$\text{weight}(\mathcal{C}^{\text{alg}}) \leq \ln(N)\text{weight}(\mathcal{C}^*).$$

So,

$$\begin{aligned}\text{cost}(\langle x_{kj}^{m_j}, \lambda_{ij}\rangle) \leq\ & \ln(N)\text{weight}(\mathcal{C}^*) \\ \leq\ & \ln(N)\text{weight}(\mathcal{C}) \\ \leq\ & 2(1+(K-1)\Delta)\ln(N) \\ & \cdot \text{cost}(\langle x_{kj}^{*m_j}, \lambda_{ij}^*\rangle).\end{aligned}$$

■

The proposed VSPS-WSC algorithm is a general algorithm, which can be used to solve the VSPS problem without assuming any particular queueing model. However, only the condition that each DC's capacity is linear in the number of its VSs is satisfied, this algorithm has the approximation ratio as shown above. Note that the M/M/c queueing model has this property because the queueing delay at DC j can be given by $\frac{1}{m_j\mu_j-\lambda_j}$, (where μ_j and λ_j are the service rate of each VS and the total arrival rate respectively) and if a queueing delay bound is given, the maximum allowable arrival rate (capacity) of DC j is linear in m_j.

Next, we analyze the time complexity of the VSPS-WSC algorithm: In Step 1, for each of M DCs, we must sort the clients in the ascending order of communication delays t_{ij}: this can be done in $O(N\log N)$ time per DC, for a total of $O(MN^2\log N)$ time. Additionally N weighted sets are created per DC. A simple way to represent these sets is to use a $MN \times N$ matrix, with the rows representing the sets and the columns representing the clients. For each set S we will also maintain $S \cap U$. The argmin operation in Step 3 can be implemented by scanning the list of MN sets to find the one with minimum weight to remaining size ratio. There will be at most N iterations of the while loop in Step 3 since at least one element is removed from U each iteration. With simple data structures, Step 3 can be completed in $O(MN^2)$ time. Step 4 selects a single set for each DC and builds the corresponding VSPS solution. It can also be done in $O(MN^2)$ time, so the overall time complexity of the VSPS-WSC algorithm is $O(MN^2\log N)$.

Even though the approximation ratio obtained above is not a constant, we can show this is the best approximation ratio a polynomial-time algorithm can achieve due to the computational complexity of the VSPS problem.

Theorem 2: The VSPS problem does not have an approximation algorithm better that $(1-o(N))\ln(N)$, unless NP has TIME($n^{O(\log\log n)}$) deterministic time algorithms.

Proof: We can show a simple reduction from the set-cover problem to the VSPS problem: Given an instance of the set-cover problem consisting of n elements and m sets, create a client i corresponding to each element i and a DC j corresponding to each set S_j. For $i \in S_j$, set the communication delay $t_{ij} = 0$ and for $i \notin S_j$, set $t_{ij} > T$. Set the DC capacities such that one VS is sufficient to handle all possible demands to each DC, and set $m_j = 1$. Thus the VSPS problem becomes deciding which DCs should turn on one VS in order to serve all of the clients. This directly encodes the set-cover problem and so any γ-approximation algorithm for the VSPS problem will provide a γ-approximation algorithm for the set-cover problem. It is well-known [11] that the set-cover problem does not have an approximation algorithm better that $(1-o(n))\ln(n)$, unless NP has TIME($n^{O(\log\log n)}$) deterministic time algorithms. ■

Even though the VSPS-WSC algorithm can provide certain worst-case performance guarantee, it may not return a "good" solution on average cases since its Step 3 may not give the best request distribution that can lead to further cost reduction. Once the VS provisioning and DC selection solutions are determined, an optimal request distribution solution can be obtained by solving an LP since the only decision variables $\langle\lambda_{ij}\rangle$ take real (instead of integer) values. Therefore, we propose a post optimization algorithm to improve the cost of a given VSPS solution further. We define, the expected reduced cost-to-load ratio of a DC j, $r_j = \frac{c_j}{\Delta C_j^-}$, where c_j is the per-VS cost and ΔC_j^- is the expected reduced load, which is the load that has to be removed from DC j if DC j shuts down one VS. Of course, this ratio of a DC with no VS is 0. The algorithm is presented in the following.

Algorithm 3 Post-Opt ($\langle y_{ij}, m_j\rangle$)

```
Step 1  optimized := FALSE;
Step 2  while (optimized = FALSE)
          Let θ sort the DCs by descending r_j
          for i := 1 to M
            j := θ(i);
            if (r_j = 0)
              optimized := TRUE;
              break;
            endif
            m_j := m_j − 1;
            Compute capacity of DC j, C_j;
            Solve Request-Distribution-LP(⟨y_ij, C_j⟩);
            if (Found feasible solution)
              break;
            else
              m_j := m_j + 1;
            endif
            if (j = θ(M))
              optimized := TRUE;
            endif
          endfor
        endwhile
Step 3  return ⟨λ_ij, m_j⟩.
```

Request-Distribution-LP ($\langle y_{ij}, C_j\rangle$):

$$\min \sum_{j=1}^{M} p_j \sum_{i=1}^{N} \lambda_{ij} \tag{7}$$

$$\sum_{j:y_{ij}=1} \lambda_{ij} = d_i, \quad i \in \{1,\ldots,N\}, \tag{8}$$

$$\sum_{i:y_{ij}=1} \lambda_{ij} \leq C_j, \quad j \in \{1,\ldots,M\}; \tag{9}$$

As described above, the capacity of a DC C_j can be obtained by using a queueing model or a profile-based approach,

once DC selection and VS provisioning solutions $\langle y_{ij}, m_j \rangle$ are given, where $y_{ij} = 1$ if client i is served by DC j and m_j is the number of VSs on DC j. By solving Request-Distribution-LP ($\langle y_{ij}, C_j \rangle$), we can obtain a feasible solution for request distribution corresponding to the given DC selection and VS provisioning solutions specified by $\langle y_{ij}, m_j \rangle$. Even though we only need to find a feasible solution here, we try to minimize the cost by using the objective function (7), where p_j is the per-load-cost on DC j corresponding to given $\langle y_{ij}, m_j \rangle$. We found this objective function is more effective than others via simulation. The algorithm keeps reducing VSs and solving the LP to check if a feasible request distribution can be obtained until it is no longer possible. Note that this post optimization will only return a solution whose cost is no larger than that of the given solution. This post optimization can be used after running the VSPS-WSC algorithm described above. We refer to the corresponding algorithm as the *VSPS-WSC-Opt* algorithm. Since The VSPS-WSC-Opt will only return a solution whose cost is no larger than that of the solution given by the VSPS-WSC algorithm, it has the same approximation ratio as the VSPS-WSC algorithm. In addition, it is known an LP with polynomial number of variables and constraints can be efficiently solved in polynomial time [6], hence, the VSPS-WSC-Opt algorithm is still a polynomial time algorithm.

V. Joint Algorithm

Instead of solving the VSPS problem in three separate steps, we develop an effective heuristic algorithm that jointly solves the VS provisioning and selection subproblems. The basic idea of the proposed algorithm is to pack service demands of clients to DCs one by one until every client's demand is satisfied. The algorithm deals with the clients one by one in the descending order of their demands. Every time, the algorithm first tries to accommodate the demand of the client in question (let's call it client i) without adding a new VS. If its demand is not fully fulfilled, it then tries to find a DC j that gives the lowest effective cost-to-demand ratio e_{ij} to cover the remaining demand: $e_{ij} = \min_{k \in \{1, \cdots, M_j - m_j\}} (\frac{k \cdot c_j}{\Delta C^+_{kij}})$, where ΔC^+_{kij} is the effective demand that can be covered by adding k new VSs on DC j, which is determined by the remaining demand d'_i of client i and the additional capacity that can be provisioned by adding k new VSs (whichever is smaller). Note that its remaining demand may not be completely fulfilled in this DC. In addition, if DC j becomes unusable for client i due to the service response time constraint, the corresponding $\Delta C^+_{kij} = 0$. Moreover if $\Delta C^+_{kij} = 0, \forall k \in \{1, \ldots, M_j - m_j\}$, then the corresponding $e_{ij} = \infty$. The Joint algorithm is presented as Algorithm 4. In the algorithm, k_{j^*} and $\Delta C^+_{j^*}$ are the number of VSs and the effective additional demand corresponding to the selected DC j^*.

Next, we analyze the time complexity of this algorithm. Step 1 can be done in $O(N \log N)$ time. In Step 2, it takes $O(MM_{\max})$ time (where $M_{\max} = \max_{j \in \{1, \cdots, M_j\}} M_j$) to obtain the best DC selection and request distribution solution for each client. Hence, the running time of Step 2 is $O(NMM_{\max})$. The overall time complexity of the joint algorithm is then $O(N \log N + NMM_{\max})$.

Algorithm 4 VSPS-Joint

Step 1 Let γ sort the clients by descending demands d_i;
Step 2 **for** $h = 1$ **to** N
  $i := \gamma(h)$;
  Find free capacities from DCs to serve client i without adding a new VS;
  Let d'_i be the remaining demand;
  while $(d'_i > 0)$
    $j^* := \text{argmin}_{j \in \{1, \ldots, M\}} e_{ij}$;
    if $(e_{ij^*} = \infty)$
      return Infeasible;
    endif
    $m_j := m_j + k_{j^*}$;
    $\lambda_{ij} := \lambda_{ij} + \Delta C^+_{j^*}$;
    $d'_i := d'_i - \Delta C^+_{j^*}$;
  endwhile
endfor
Step 3 **return** $\langle \lambda_{ij}, m_j \rangle$.

VI. Simulation Results

In the simulation, we used the real locations of 12 Amazon's DCs in USA as input. We obtained the electricity prices from the U.S. energy information administration's website [10] and used the method in [16] to calculate the per-VS costs in different locations. The clients were uniformly placed within a rectangular area that covers all the DCs. To estimate the communication delay between a client and a DC, we employed the equation presented in [14]:

$$RTT_{i,j} = 62 + 0.02 \times d_{ij},$$

where d_{ij} is the geographical distance between two nodes i and j in kilometers. Note that the authors of [14] obtained such an equation by analyzing a large set of measurement data.

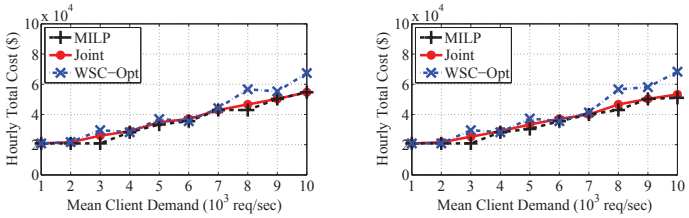
To estimate queueing delay and the corresponding DC capacity (maximum allowable arrival rate), we employ two approaches: M/M/c model and profile-based. It is well known if the M/M/c model [12] is assumed for queueing, and arrival/service rates and the number of servers are given, a closed-form equation can be used to calculate queueing delay and correspondingly, if a queueing delay bound, the service rate and the number of servers are given, we can easily obtain the capacity. In the profiled-based approach, we set up a real testing environment using a popular HTTP server NGINX [15] (for serving HTTP requests) and the Apache benchmarking tool [5] (for generating HTTP requests). In addition, we created a VM running Linux using VMware VSphere 5.0 [18] to host the HTTP server. We obtained the server's capacity after performing a large number experiments by setting the number of requests and concurrency parameter to different values. The delay threshold was set to $T = 110$ms in the simulation.

We randomly generated client's demands and assumed that they followed Gaussian distributions. We fixed the standard deviation to 500 req/sec and increased the mean demand of a client from 1000 req/sec to 10000 req/sec, with a step size of 1000 req/sec. All the results presented in the following figures

are average over 10 runs and in each run, a different seed was used for random generation of demands.

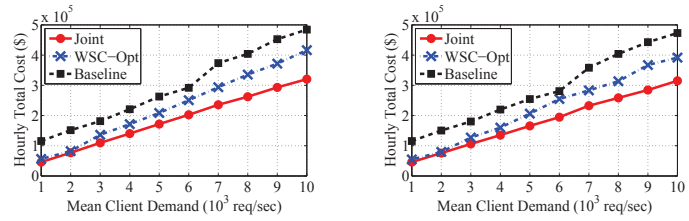
In the simulation, we evaluated the performance of the proposed VSPS-WSC-Opt (labeled as "WSC-Opt") and VSPS-Joint (labeled as "Joint") algorithms in terms of the total hourly (electricity) cost. We solved the VSPS-MILP (labeled as "MILP") using the Gurobi Optimizer 5.0 [13] and presented the results for comparison. Moreover, we implemented a baseline algorithm in which each client uses the closest DC (in term of communication delay) to serve all its demand.

We conducted simulation runs on both small cases and large cases. In the small cases, we had 3 DCs and 8 clients, while in the large cases, we had 12 DCs and 60 clients. The simulation results are presented in Figs. 1(a)-2(b).



(a) Small cases, M/M/c model (b) Small cases, profile-based

Fig. 1. Performance of the proposed algorithms on small cases



(a) Large cases, M/M/c model (b) Large cases, profile-based

Fig. 2. Performance of the proposed algorithms on large cases

We make the following observations from these results:

1) The main purpose of running simulation over small cases is to find out the gap between MILP solutions and those provided by the proposed algorithms. It can be seen from Fig. 1, the proposed algorithms yield close-to-optimal performance, especially the VSPS-Joint algorithm. On average, the difference between the cost given by the VSPS-Joint algorithm and the optimal is only 5.05%.

2) In the large cases, we compared our algorithms against the baseline algorithm since it took too long to solve the VSPS-MILP. From Fig. 2, we can see that the proposed algorithms consistently outperform the baseline algorithm. Among all the proposed algorithms, the Joint algorithm still perform the best. On average, the VSPS-WSC-Opt and VSPS-Joint algorithms achieve 25.58%, and 39.37% cost reduction compared to the baseline algorithm. Note that the VSPS-Joint algorithm turns out to perform best on these average (randomly generated) cases even though the VSPS-WSC-Opt algorithm can provide certain worst-case performance guarantee.

3) No matter which approach (M/M/c or profile-based) was used for estimating queueing delay and DC capacity, the proposed algorithms perform quite similarly. Specifically, they all consistently outperform the baseline algorithm, and the costs given by all three algorithms monotonically increase with the client demands. This shows that the proposed algorithms can be used for different services/applications.

VII. Conclusions

In this paper, we have studied a Virtual Server Provisioning and Selection (VSPS) problem in distributed DCs with the objective of minimizing the total operational cost while meeting the service response time requirement. First, we formulated the VSPS problem as an MILP problem that can be used to provide optimal solutions. Then we proposed a 3-step optimization framework to guide the algorithm design. Under such a framework, we developed a polynomial-time $\ln(N)$-approximation algorithm. We also showed that the VSPS problem is NP-hard to approximate and is not possible to obtain a better approximation ratio unless NP has TIME$(n^{O(\log \log n)})$ deterministic time algorithms. In addition, we presented an effective heuristic algorithm that jointly obtain the VS provisioning and selection solutions. It has been shown by simulation results that the proposed algorithms provide close-to-optimal performance and achieve 25% or more cost reduction compared to a baseline algorithm.

References

[1] M. A. Adnan, R. Sugihara and R. K. Gupta. Energy efficient geographical load balancing via dynamic deferral of workload, *Proceedings of IEEE Cloud'2012*, pp. 188-195.
[2] Akamai Technologies Inc., *http://www.akamai.com/*
[3] M. Alicherry and T.V. Lakshman, Network aware resource allocation in distributed clouds, *Proceedings of IEEE Infocom'2012*, pp.963–971.
[4] Amazon Web Service Global Infrastructure, *http:// aws.amazon.com/about-aws/globalinfrastructure/*
[5] Apache HTTP server benchmarking tool, *http://httpd.apache.org/docs/2.2/programs/ab.html*
[6] M. S. Bazaraa, J. J. Jarvis and H. D. Sherali, *Linear Programming and Network Flows (3rd Edition)*, John Wiley & Sons, 2005.
[7] J. Cao, M. Andersson, C. Nyberg and M. Kihl, Web server performance modeling using an M/G/1/K*PS queue, *Proceedings of ICT'2003*, pp. 1501-1506.
[8] V. Cardellini, M. Colajanni and P. S. Yu, Geographic load balancing for scalable distributed web systems, *Proceedings of IEEE MASCOTS'2000*.
[9] M. Conti, E. Gregori, and F. Panzieri, Load distribution among replicated Web servers: a QoS-based approach, *ACM SIGMETRICS Performance Evaluation Review*, Vol. 27, No. 4, 2000, pp. 12–19.
[10] U.S. Energy Information Administration Electricity Data, *http://www.eia.gov/electricity/data.cfm*
[11] U. Fiege, A threshold of ln n for approximating set cover, *J. ACM*, Vol. 45, No. 4, 1998, pp. 634–652.
[12] D. Gross and C. Harris, *Fundamentals of Queueing Theory (3rd Edition)*, John Wiley & Sons, 1998.
[13] Gurobi Optimizer 5.0, *http://www.gurobi.com/*
[14] S. Kaune, K. Pussep, C. Leng, A. Kovacevic, G. Tyson and R. Steinmetz, Modelling the Internet delay space based on geographical locations, *Proceedings of ACM PDP'2009*.
[15] NGINX,Inc., *http://nginx.com/*
[16] L. Rao, X. Liu, L. Xie and W. Liu, Minimizing electricity cost: optimization of distributed Internet data centers in a multi-electricity-market environment, *Proceedings of IEEE Infocom'2010*.
[17] L. Rao, X. Liu, M. Ilic and J. Liu, MEC-IDC: joint load balancing and power control for distributed Internet data centers, *Proceedings of ACM ICCPS'2010*, pp. 188–197.
[18] VMware vSphere, *http://www.vmware.com/products/vsphere/*
[19] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford, DONAR: decentralized server selection for cloud services, *Proceedings of the ACM SIGCOMM'2010*, pp. 231–242.
[20] Weighted Set Cover (online lecture notes), *http://www.cs.huji.ac.il/course/2005/algo2/scribes/lecture2.pdf*