Low-cost Application Image Distribution on Worldwide Cloud Front Server

Yang Liu†, Shi Bai†, Weiyi Zhang†, Jun Zhang‡

†Dept. of Computer Science
‡Dept. of Industrial and Manufacturing Engineering
North Dakota State University, Fargo, ND 58105,
Email: {Yang.Liu.3, Weiyi.Zhang, Shi.Bai, Jun.Zhang}@ndsu.edu

***Abstract*—Cloud computing opens a new area of supplement, consumption, and delivery framework for IT services. Customers could be able to order Virtual Applications through the cloud. To reduce the latency time, the cloud service providers implement some strategies (e.g., cloudfront service [1]) to speed up the applications delivery. However, these strategies do not consider the profit of application providers. In this paper we address the problem which is to maximize the profit of application providers based on the Original-Front server network model. We studied two different scenarios and proposed two efficient heuristic algorithms. Our simulation results show that our heuristic algorithms can increase the profit of application providers significantly.**

***Keywords*: Cloud Computing, Front Server, Application Distribution, Maximize profit.**

I. Introduction

Cloud computing opens a new area of supplement, consumption, and delivery framework for IT services, and it involves over-the-Internet provision of dynamically scalable and virtualized resources which is significant trends with the potential to increase agility and lower costs of IT [2]. Virtual Infrastructure cloud services (e.g., [1], [5]) are virtual hardware provider, where customers can deploy virtual servers and run applications. The virtual server vendor which is an emerging cloud service is the motivation of this paper. Cloud customers could be able to order Virtual Applications which can be delivered virtually by the cloud providers on the cloud. Three characters are involved in this framework: virtual application provider, cloud service provider and virtual application customer. Virtual application providers create the applications and put them on the original cloud storage servers which are provided by the cloud providers. The customers purchase the virtual applications from the application providers. It is worth noting that customers want their orders to be delivered as fast as possible, so they do not waste their time slot. In particular, customers do not want to wait very long latencies associated with transferring large objects over the Internet. However, the original cloud storage servers do not always lie near the target clients. And the provisioning time could be much longer than the expected time. Therefore, a strategy is needed to speed up delivery of the reserved applications.

The research developed in this paper is supported by NSF CNS-1022552, FAR-0016614, NSF ND EPSCoR under the Infrastructure Improvement Program FAR-0015846 and FAR-0017488.

To reduce the latency time, the cloud service providers implement a *front server* strategy (e.g., [1]), shown in Fig. 1. This model is name Original-Front Server (OFS) Model in this paper. Since the front servers are located near to the terminal customers geographically and globally, when the reservations are made by the customers, application images can be delivered to the front server which is the nearest one to the customer geographically. Here we assume that the latency time for delivering the applications is shorter if the front server is closer to the customer's locations. Hence, the provisioning time can be reduced significantly. Theoretically, requests for the applications are automatically routed to the nearest front server, so content is delivered with the best possible performance. However, we notice that this model is good for cloud providers but not for application providers. Since application providers need to pay for renting the front servers from the cloud providers. For example, the expense is e_k^j if the application providers need to put the application k on the front server j. If the customers want to use the *front service*, they have to pay the application providers and activate this service. For example, for customer i, by making the reservation of the application k in the front server, the cost is v_k^i. Hence, delivering the content to the nearest front server for each customer will benefit the cloud providers but not application providers. An efficient application images distribution strategy is needed to maximize the profit of application providers.

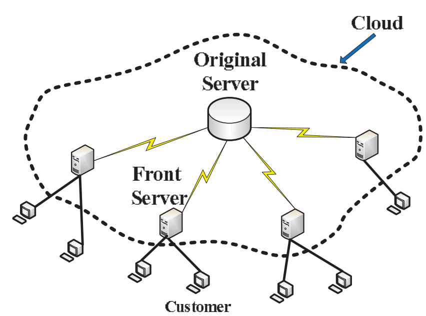


Fig. 1. Illustration for Original-Front Server Model

In this paper, we have this assumption that application providers have the option to decide which front servers need to be deployed with application images. Obviously, if application providers try to maximize the total profit Φ_{total}, they need to satisfy the customers with the minimum number of front servers. We assume that if the expected latency time for delivering the application *k* to the customer *j* is t^{k}_{exp}, there will always exist one or more front servers which can meet the deadline. The problem we target to solve in this paper is to maximize the profit Φ_{total} for application providers.

Our contribution in this paper is two-fold. First, we present an application image distribution problem to maximize the profit of application providers based on the Original-Front Server model. Second, we study this problem in two scenarios and propose two efficient heuristic algorithms for each scenario to solve this problem.

In the rest of the paper, we describe the related work in Section II. Section III demonstrates the problem statement of our work. And the solution for this work is studied in Section IV. We present numerical results in Section V and which is followed by the conclusions in VI.

II. Related Works

How to provision the Virtual applications through cloud rapidly has been study recently [6]. In paper [4], the authors studied a fundamental storage staging problem and presented it as a scheduling problem with capacity constraints under two models, a continuous model and an integral model.

Similar to application distribution, content distribution has been studied in the context of web content through Content Distribution Networks [8], [9]. Some Content Distribution Networks implementations introduce related job scheduling problems. In [3], the authors studied the scheduling problem for cache pre-filling. Many content distribution systems adopted web caching techniques [7], where frequently accessed objects are stored near the customers. These techniques can reduce both access latency and network traffic.

In could computing, virtual infrastructure cloud services (e.g., [1], [5]) are both a virtual hardware provider and a virtual hosting premise, where customers can deploy virtual servers and run applications. Cloud providers provide some special web services for content delivery, such as Amazon CloudFront [1] which cooperates with other Amazon Web Services to serve developers and businesses an easy way to distribute content to end customer with high data transfer speeds, low latency, and no commitments. With a global network of edge locations, Amazon CloudFront can deliver your static and streaming content rapidly.

In this work, the problem we target to solve is to maximize the profit Φ_{total} for application providers, and to the best of our knowledge, this problem is rarely studied in the previous works.

III. Problem Statement

In this work, the Original-Front Server (OFS) model is adopted. By cooperating with cloud providers, application providers serve the virtual applications to terminal customers. In the OFS model, applications are stored on the original storage servers initially. Customers send the application reservation information to the application providers. According to the information, the distribution strategies are proposed by application providers and processed by the cloud providers.

We are given a set of customers $C = \{c_1, c_2, ...c_t\}$, a set of front servers $F = \{f_1, f_2, ...fm\}$ and a set of virtual applications $K = \{k_1, k_2, ...k_n\}$. The number of reservation of applications for each customer is different. For customer *i*, h_i denotes the number of applications which are reserved by customer *i*, and v^{i}_{k} denotes the cost when customer *i* makes the reservation of application *k* on one front server. The cost of transferring and retaining application *k* on server *j* is e^{j}_{k} which is the payout of application providers. The price strategies for both cloud providers and application providers are out of the scope of this paper. Since the capacity and bandwidth for each front server is limited, we have the following constraint in this work. For each front server, the maximum number of customers which connect to one front server at the same time should not exceed ε. When customer *i* reserves application *k*, the expected delivery time denotes by $t^{i,k}_{exp}$. If provisioning time $t^{i,k}_{pro}$ is greater than $t^{i,k}_{exp}$, the penalty we assumed is that customer *i* will request the cost v^{i}_{k} back. However, even the money is called back for the single application, the customers are still unsatisfied if many applications cannot be delivered in time. Hence, we state the following definition, Def. 1.

Definition 1 (Minimum Satisfaction Ratio (MSR)): A factor $\alpha_i = \frac{\sum K^{i}_{sat}}{\sum K^{i}_{all}}$ *is used to represent the minimum satisfaction ratio of customer i. Customer* i *reserves* x *applications totally and the minimum number of applications should be satisfied is* y*, where* $y \leq x$. *In this case,* α_i *is calculate as* $\frac{y}{x}$. *If the satisfaction ratio is less than* α_i*, customer* i *has right to refuse the payment of all the applications which have been reserved.* □

TABLE I

Symbols	Definitions
C	Set of application customers
F	Set of front servers
K	Set of virtual applications
v^{i}_{k}	Cost of customer *i* by using app *k*
e^{j}_{k}	Cost for deploying app *k* on server *j*
$t^{i,k}_{pro}$	Provisioning time of app *k* of customer *i*
$t^{i,k}_{exp}$	Expected delivery time of app *k* for customer *i*
γ_{in}	Income of the application provider
γ_{out}	Payout of the application provider
W_i	Set of eligible front servers of customer *i*
α_i	Expected satisfaction ratio of customer *i*
D_i	Set of leaves on node *i*
Φ	Profit of application providers
V	Set of intermediate nodes
W	Set of leaf nodes

Definition 2 (Eligible Front Server): For customer i, if the front server j can deliver the reserved application to customer i in time, then front server j is an eligible front server for customer i. In other words, if the latency time $t_{pro}^{i,k} \leq t_{exp}^{i,k}$, then server j is an eligible server to customer i. □

Definition 3 (Maximum prOfit of Application Distribution): Given a set of customers $C = \{c_1, c_2, ...c_t\}$, a set of application $K = \{k_1, k_2, ...k_n\}$, and a set of front servers $F = \{f_1, f_2, ...fm\}$, for the application provider, the income γ_{in} is calculated as

$$\gamma_{in} = \sum_{i=1}^{t} \sum_{k=1}^{h_i} v_k^i s_k^i \alpha_i$$

where s_k^i denotes that if there exists an eligible front server to serve customer *i* to use application *k*. s_k^i can be determined by the value of x_i, which is

$$x_i = \max[x_k^{i,1}, ..., x_k^{i,j}, ..., x_k^{i,m}],$$

where, $x_k^{i,j} = [\frac{t_{exp}^{i,k}}{t_{pro}^{i,j,k}}]f(j,k)$. *f(j,k)* is the decision variable which denotes if the application k is deployed on front server j. If $x_i \geq 1$, then s_k^i=1; otherwise, s_k^i=0.
For the application provider, the payout is calculated as

$$\gamma_{out} = \sum_{j=1}^{m} \sum_{k=1}^{n} e_k^j f(j,k),$$

The objective in this work is to maximize the total profit of the application provider:

$$\textbf{Maximize} \sum_{i=1}^{t} \alpha_i [\sum_{k=1}^{h_i} v_k^i s_k^i] - \sum_{j=1}^{m} \sum_{k=1}^{n} e_k^j f(j,k)$$

□

IV. Proposed Solutions

In this section, we study two different scenarios of application distribution and present two distinguish heuristic algorithms to solve the *Maximum prOfit of Application Distribution* **MOAD** problem in these two scenarios.

A. *OFS model with single application*

We start with a special case where there is only one application be provided in the OFS model. Since there is only one application, MSR α should equal to *1* for each customer in this case.

We construct a tree network *G* where the original storage server is the root of the tree. The set of intermediate nodes $V = \{n_1, n_2, ...n_i\}$ denote the front servers, and a set of leaf nodes $l = \{l_1, l_2 ..., l_j\}$ denote the customers. (in the following, we use front server and intermediate node interchangeably, as well as customer and leaf node.) Let the set $D_i = \{d_1^i, d_2^i, ...d_a^i\}$ represents the the leaves on intermediate node *i*, which also means the customers who are routed to the front server f_i. According to the geographical location information, the set of eligible front servers of customer *i* who reserved application *k* is calculated and represented by $W_i = \{w_1^i, w_2^i, ...w_a^i\}$ where *a* is no more than the number of front servers *m*.

Following the conditions assumed for the OFS model, the income of front server j which is deployed with application k is $\gamma_{in}^j = \sum_{i=1}^{x} v_k^{i,j}$ where *x* is the number of the customers connected to the front server *j*. We use Ω_j to denote the ratio between the income and the payment of front server *j*,

$$\Omega_j = \frac{\gamma_{in}^j}{\gamma_{out}^j}$$

where $\gamma_{out}^j = e_k^j$.

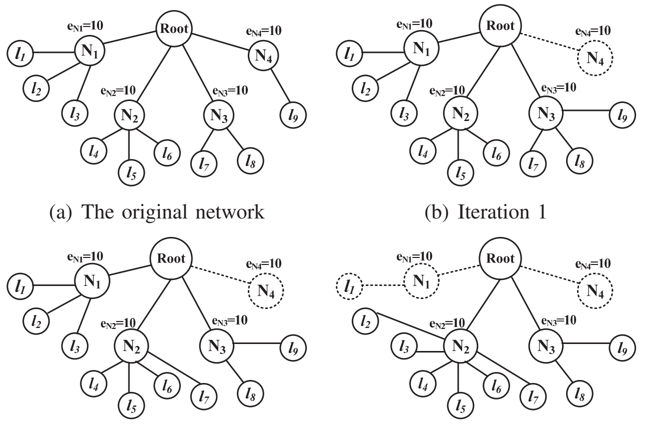
In this case, we assume the application images have been deployed on all of the front servers initially. Our *H-MOAD* algorithm, shown in Algorithm 1, first chooses the node n_i with the minimum value Ω_{min} in network *G*. Then, if the leaf on node n_i has other eligible front nodes, it will be transferred to the one which has the maximum value Ω_{max}. For example, if leaf l_a has three eligible nodes which are $\{n_1, n_2, n_3\}$, and l_a currently connected to node n_1. According to Algorithm 1, if $\Omega_{n_2} \geq \Omega_{n_3}$, l_a should be transferred to n_2.

On the other hand, if leaf l_a do not have any other eligible node, l_a should be kept on the current node. After processing the transfer, we recalculate the value Ω'_{n_i} for node n_i. If $\Omega'_{n_i} \leq 1$, we remove node n_i and its current leaves from network *G*, otherwise, node n_i is marked as *"pruned"*. Then our algorithm continues to repeat this process among the *"unpruned"* nodes until none *"unpruned"* nodes left. Based on our algorithm, the whole process should be repeated until all of the leaves and nodes in network *G* achieve stable. In other words, no leaf transfer occurs in *G*.

Algorithm 1 Single App H-MOAD(G)

1: Construct sets of intermediate nodes V, V', and V'';
2: $V \leftarrow \{n_1, n_2, \ldots, n_i\}$; $V' \leftarrow V$; $V'' \leftarrow \emptyset$;
3: **for** each *leaf* *i* in *G* **do**
4:    Connect *i* to the node n_{nea} which is nearest node to *i*;
5: **end for**
6: **while** $V \neq V''$ **do**
7:    Find the node n_i with the minimum value $\Omega_{min}^{n_i}$;
8:    **for** each *leaf* *l* in the set D_{n_i} **do**
9:       **if** $|W_l| > 1$ (E_l is the set of eligible nodes of *l*) **then**
10:          Transfer *l* to the node n_{max} which is the node with the maximum value in set W_l .
11:       **end if**
12:    **end for**
13:    **if** $\Omega'_n \geq 1$ **then**
14:       Mark *n* as "*pruned*";
15:    **else**
16:       Delete n_i and leaves in D'_{n_i};
17:    **end if**
18:    Repeat step 6 to step 15; Move node n_i to set V''.
19: **end while**

Let's use an example in Fig. 2(a) to illustrate Algorithm 1. In this example, we simply assume that the payout of the application k on each front server is the same which is $e_k = 10$, and the income from each customer who uses front service is $v_k = 5$. Initially, leaves are automatically routed to the nearest nodes, shown in Fig. 2(a). Following to Algorithm 1, in the first iteration, node N_4 is selected due to $\Omega_{N_4}^k = \frac{5}{10}$ which is the minimum value in network G.



(a) The original network (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

Fig. 2. Illustration of Algorithm 1

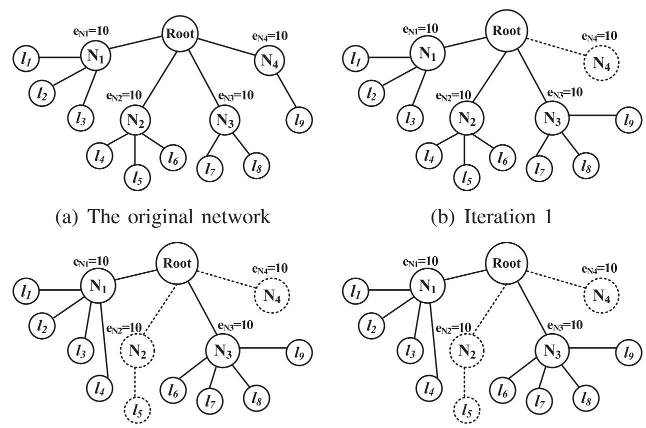
From the set of eligible nodes of leaf l_9 $W_{l_9} = \{N_3, N_4\}$, we choose node N_3 as a target node to which leaf l_9 can be transferred, shown in Fig. 3(b). There is no leaf left on node N_4 after transferring leaf l_9 to N_3. Hence, node N_4 is deleted from the network, in other words, the image of application k will not be deployed on front server N_4. In the second iteration, since $\Omega^k_{N_2} = \Omega^k_{N_3} = \frac{15}{10}$, node N_3 is selected randomly. The sets of eligible nodes of leaves l_7, l_8 and l_9 are $E^k_{l_7} = \{N_2, N_3\}$, $E^k_{l_8} = \{N_3\}$, $E^k_{l_9} = \{N_3\}$, respectively. Only leaf l_7 is transferred to node N_2 in this iteration, shown in Fig. 2(c). Since the value $\Omega'^k_{N_3} = \frac{10}{10} = 1$ which means N_3 can be kept in the network and marked as "*pruned*". Fig. 2(d) illustrates the third iteration. The same situation happens on node N_1. Leaves l_2 and l_3 are transferred to N_2, and the value of node N_1 is $\Omega'^k_{N_3} = \frac{5}{10}$ which means N_1 should be deleted from the network. At this time, the network get stable. Consequently, N_1 and N_3 are chosen to be distributed with application images.

B. *OFS model with multi-applications*

Now we need to generalize our approach to the networks with multiple applications. For the general network model, the number of applications are *k*. The value of MSR α is assigned to each customer. In this multi-application model, we present a multi-layer strategy which could separate the **MOAD** problem into numbers of subproblem. Our Multi-apps H-MOAD algorithm, shown in Algorithm 2, is implemented on each subproblem. It is worth noting that in this multi-layers OFS model, we aim to maximize the profit of the whole network, not only in the single layer. And we adopt a new factor $\varphi = \frac{1}{\beta_i - \alpha_i}$ and calculate the value by the formula Ω_{min}=$\frac{\sum_{i=1}^{x,q} \frac{v^q_{i_k}}{\beta_i - \alpha_i}}{\sum_{i=1}^{x,q} e^{j,q}_k}$. A set of tree networks $G_{mul} = \{g_1, g_2, ...g_k\}$ is constructed to demonstrate the multi-layers model. Each tree network represents a layer, and only one application are considered in each layer. Algorithm 2 illustrates our Multi-application heuristic algorithm.

Algorithm 2 Multi-Apps H-MOAD(G)

1: Construct sets of intermediate nodes V, V', and V'';
2: $V \leftarrow \{n^q_1, n^q_2, \ldots, n^q_i\}$; $V' \leftarrow V$; $V'' \leftarrow \emptyset$;
3: **for** each *leaf* i in G^q **do**
4:   Connect i to the node n_{nea} which is nearest node to i;
5: **end for**
6: **while** $V \neq V''$ **do**
7:   Find the node n with the minimum value Ω_{min}=$(\sum_{i=1}^{x,q} \frac{v^q_{i_k}}{\beta_i - \alpha_i})/\sum_{i=1}^{x,q} e^{j,q}_k$ where β_i is the number of remained applications to leaf i in all of the layers;
8:   **for** each *leaf* l in the set D^q_n, where D^q_n is node n's leaf set **do**
9:     **if** $|W_l| > 1$ **then**
10:       Transfer l to the node n_{max} which is the node with the maximum value in the set W_l;
11:     **end if**
12:   **end for**
13:   **if** $\Omega'_n \geq 1$ **then**
14:     Move *n* to set *V'*;
15:   **else**
16:     Delete *n* and the leaves in D'_n;
17:   **end if**
18:   Repeat step 6 to step 15; Move node *n* to set V''.
19: **end while**



(a) The original network (b) Iteration 1

(c) Iteration 2 (d) Iteration 3

Fig. 3. Illustration of Algorithm 2

In the layer *q*, customer *i* has a set of eligible front server which represented by $W^{i,q} = \{w^{i,q}_1, w^{i,q}_2, ..., w^{i,q}_a\}$. For the front server *j*, the income is

$$\gamma^{j,q}_{in} = \sum_{i=1}^{x,q} v^q_{i_k} \alpha_i$$

where *x* is the number of the customers connected to the front server *j*. The profit for the application provider in layer *q* is

$$\Phi^q_j = \gamma^{j,q}_{in} - \gamma^{j,q}_{out} = \sum_{i=1}^{x,q} v^q_{i_k} \alpha_i - \sum_{i=1}^{x,q} e^{j,q}_k$$

.

We use the following example to illustrate algorithm 2,

TABLE II
EXAMPLE

Symbols	Value	Symbols	Value	Symbols	Value
α_{l1}	0.5	β_{l1}	0.9	φ_{l1}	2.5
α_{l2}	0.5	β_{l2}	0.8	φ_{l2}	3.3
α_{l3}	0.6	β_{l3}	0.75	φ_{l3}	6.7
α_{l4}	0.6	β_{l4}	0.9	φ_{l4}	3.3
α_{l5}	0.4	β_{l5}	0.6	φ_{l5}	5
α_{l6}	0.4	β_{l6}	0.9	φ_{l6}	2
α_{l7}	0.7	β_{l7}	0.8	φ_{l7}	10
α_{l8}	0.7	β_{l8}	0.95	φ_{l8}	4
α_{l9}	0.5	β_{l9}	0.7	φ_{l9}	5

shown in Fig. 3. The value of α, β and φ for each leaf can be found in Table. II. In layer q, we simply assume that the payout for each front server is $e_k^q = 10$, and the income from each customer can be calculated by $\gamma_{in}^{j,q}=\sum_{i=1}^{x,q} \frac{v_{i_k}^q}{\beta_i-\alpha_i}$ where $v_{i_k}^q$=1 in this example. Algorithm 2 is implemented on each layer with considering the MSR α. First, we calculate the value Ω of each node, where $\Omega_{N_1}=\frac{12.5}{10}$, $\Omega_{N_2}=\frac{10.3}{10}$, $\Omega_{N_3}=\frac{14}{10}$, $\Omega_{N_4}=\frac{5}{10}$. Hence, in the first iteration, node N_4 which has the minimum value in network G^q is selected. We choose node N_3 as a target node to which leaf l_9 can be transferred. Since N_3 is another eligible node for l_9. We delete N_4 after the transfer, since no leaf left on N_4, shown in Fig. 3(b). In iteration 2, we recalculate the value for each node where $\Omega_{N_1}=\frac{12.5}{10}$, $\Omega_{N_2}=\frac{10.3}{10}$, $\Omega_{N_3}=\frac{19}{10}$. Obviously, node N_2 should be selected. The sets of eligible nodes of leaves l_4, l_5 and l_6 are $E_{l_4}^k = \{N_1, N_2\}$, $E_{l_5}^k = \{N_2\}$, $E_{l_6}^k = \{N_2, N_3\}$, respectively. According to 2, leaf l_4 is transferred to node N_1 and l_6 is transferred to node N_3 in this iteration, shown in Fig. 3(c). At this time value of node N_2 $\Omega_{N_1}=\frac{5}{10} \leq 1$ which means N_2 should be deleted from G^q. Fig. 2(c) illustrates the second iteration. In the third iteration, neither the leaves on node N_1 nor the leaves on node N_3 have other eligible nodes, which means network G^p achieve stable. As a result, N_1 and N_3 are chosen to be distributed with application images in this case. The similar processes will be performed in all of the rest layers.

V. NUMERICAL RESULTS

In this section, we presented numerical results to evaluate the performances of our solutions. We implemented our Heuristic Algorithm, which was denoted as H-MOAD in the figures. For comparison, we also implemented the scenario without optimization which aims to satisfy all the customers. This scenario was denoted as Original Distribution in the figures. All our simulation runs were performed on a 2.8 GHz Linux PC with 2G bytes of memory. We used different network topologies in a 100 × 100 sq. units playing field to evaluate our proposed solutions. All the front servers and customers were randomly distributed in the playing field.

In our simulation, the number of front servers was set to 20. The cost e_k^j of deploying an application on a front server was set to 20. The cost of customer by using a particular application was set to 3. We also set the constraint ε that the number of customers connected to one server can not be more than 50 in our simulations. In our simulation, we implemented the scenario of OFS model with single application. The scenario of OFS model with multiple applications will be further studied and implemented in our future work.

We test the performances in terms of the profit of application providers, satisfaction ratio of customers, and number of deployed front server of our solution, which were shown in Fig. 4 and Fig. 5. Fig. 4 illustrated that H-MOAD always has a better performance of profit. Another observation is that as the number of customers increased, the profit also increased. For the satisfaction ratio, both H-MOAD and Original Distribution have the similar performance. The satisfaction ratio of Original Distribution is a little better than the one of H-MOAD, since the Original aims to satisfy all the requirements of the customers. Fig. 6 shows us that, compare to the original distribution, our H-MOAD protocol can satisfy the near maximum number of customers with much less front servers.

To sum up, our simulations demonstrated that the H-MOAD protocol achieves similar satisfaction ratio as the optimal solution, while increasing the profit of application providers. Hence, the H-MOAD protocol is suitable for Original-Front Server framework.

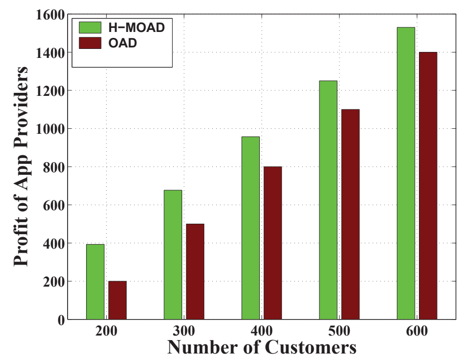


Fig. 4. profit of Application Providers

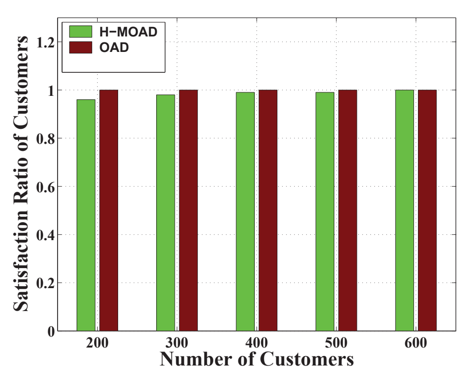


Fig. 5. Satisfaction Ratio of Customers

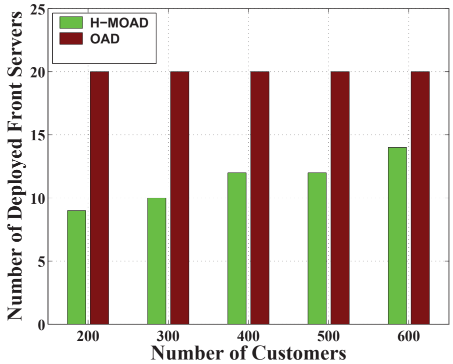


Fig. 6. Number of Deployed Front Servers

VI. Conclusions

In this work, we studied the *Maximum prOfit Application Distribution (**MOAD**)* problem, which seeks to provide an efficient strategy for the application providers to maximize profit. We studied two different scenarios in terms of single application and multi-applications. Furthermore, we proposed two fast heuristic algorithms which is called H-MOAD and Multi-app H-MOAD to solve the MOAD problem. Our simulation results show that the H-MOAD protocol can increase the profit of application providers.

References

[1] Amazon Elastic Compute Cloud (EC2), *http://aws.amazon.com/ec2*.

[2] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, Above the clouds: A berkeley view of cloud computing, University of California, Tech. Rep., 2009.

[3] R. Cohen, L. Katzir, and D. Raz, Scheduling algorithms for a cache pre-filling content distribution network, *in INFOCOM*, vol. 2, 2002, pp. 940-949.

[4] A. Epstein, D. H. Lorenz, E. Silvera, I. Shapira, Virtual Appliance Content Distribution for a Global Infrastructure Cloud Service, *INFOCOM'10 Proceedings IEEE*, Mar. 2010.

[5] Google App Engine, *http://code.google.com/appengine.*

[6] E. Kotsovinos, T. Moreton, I. Pratt, R. Ross, K. Fraser, S. Hand, and T. Harris, Global-scale service deployment in the XenoServer platform, *in Proceedings of the First Workshop on Real, Large Distributed Systems (WORLDS'04)*, Dec. 2004.

[7] M. Rabinovich and O. Spatschek, Web caching and replication, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 2002.

[8] A. Vakali and G. Pallis, Content delivery networks: Status and trends, *IEEE Internet Computing*, vol. 7, no. 6, pp. 68-74, 2003.

[9] D. C. Verma, Content Distribution Networks: An Engineering Approach. New York, NY, USA: John Wiley & Sons, Inc., 2002.