Cost Reduction in Hybrid Clouds for Enterprise Computing

Biyu Zhou,*† Fa Zhang,‡ Jie Wu,§ and Zhiyong Liu*¶
*Beijing Key Laboratory of Mobile Computing and Pervasive Device, ICT, CAS, Beijing, China
†University of Chinese Academy of Sciences, Beijing, China
‡Key Lab of Intelligent Information Processing, ICT, CAS, Beijing, China
§Department of Computer and Information Sciences, Temple University, USA
¶State Key Laboratory for Computer Architecture, ICT, CAS, Beijing, China
{zhoubiyu, zhangfa, zyliu}@ict.ac.cn, jiewu@temple.edu

***Abstract*—Hybrid cloud-based deployment is a trend in cloud computing which enables enterprise to benefit from cloud infrastructures while honoring privacy restrictions on some services. Enterprise application migration is an effective way to improve the efficiency of using the cloud infrastructures. However, it is a challenging problem to decide which parts of the applications to migrate and where to migrate. In this paper, we focus on the problem of planning the migration of enterprise applications in hybrid cloud infrastructures. Unlike previous studies, we consider a general hybrid cloud architecture that involves multiple public clouds rather than only one. Our aim is to maximize the enterprise cost reduction under the constraint of user experience in terms of response time. We first formulate the application migration problem as an optimization problem. Aware of its NP-hardness, we design an efficient migration framework to approximate the optimum for a large problem size. First, we leverage the application characteristic to reduce the scale of the problem by dividing it into multiple smaller subproblems. Then, an efficient algorithm based on dynamic programming is proposed to solve the small scale subproblems. Finally, we construct a feasible solution to the original problem. Simulation results demonstrate that our framework can bring significant benefits to enterprises.**

I. Introduction

In recent years, cloud computing has shown great potential for reducing capital and operational expense for enterprises. This benefit mainly stems from the cloud's economies of scale and the buy-on-demand model of cloud computing. However, migrating the entire enterprise application to public cloud may introduce issues in security, performance and reliability [1]. In response to these concerns, several solutions based on hybrid cloud infrastructures, which involve both on-premise cloud and public cloud, have been proposed. A hybrid cloud based solution enables enterprise to find the right balance between costs, user experience and privacy considerations. It is reported that about 60% of IT decision-makers in the US and UK choose to adopt hybrid cloud architecture to deploy their applications [2].

Typically, enterprise applications involve multi-tiers, in which each tier provides a different functionality and contains a certain amount of homogeneous servers [2]. Servers belonging to different tiers may communicate with each other. To deploy an enterprise application in a hybrid cloud environment, a key challenge is determining the location of each server. In order to solve this issue, both academia and industry have proposed solutions in recent years [1–8]. The authors in [1] are the first to study the potential benefits of hybrid cloud deployments of enterprise applications. They formulate this problem as an integer programming problem and use centralized optimization solvers to solve it. Several more sophisticated approaches in planning the placement of virtual machines that minimizes the financial cost while obeying the deadlines constraint are proposed in [4, 5, 7, 8]. In [3], the authors focus their attention on dynamic migration of content distribution services into hybrid cloud infrastructures. Their aim is to minimize operational costs with a service response time guaranteed at all times. They propose a solution that employs Lyapunov optimization techniques.

Unfortunately, most previous studies on enterprise application migration to hybrid cloud infrastructure use a centralized optimization solver to obtain the optimal placement of each server. These methods are effective when the instance is small. However, they are ineffective against typical medium-scale enterprise applications in which thousands of servers are involved. In addition, while existing works have done a good job exploring the benefits of deploying applications in the simple two clouds environment(which contains both a private and a public cloud), they seldom evaluate the benefits of employing a hybrid cloud architecture with a private cloud and multiple geographically distributed public clouds. In fact, many large cloud providers (e.g., Amazon Web Services [9] and Microsoft Azure [10]) enabled the placement of instances in multiple locations.

In this paper, our objective is to explore the benefits of migrating medium- and large-scale enterprise applications to hybrid cloud infrastructures, in which a local cloud and multiple geographically distributed public clouds are involved. To solve this problem, we propose a simple but efficient three-stage framework: 1) Partition. The framework divides the large migration problem into multiple smaller subproblems leveraging the characteristics of typical multi-tier enterprise applications. 2) Solving subproblems. An efficient algorithm based on a dynamic programming approach is then proposed to solve each subproblem. Since the scale of each subproblem is small and all the subproblems can be computed in a parallel

manner, the algorithm can output efficient solutions for all the subproblems with tolerable running time overheads. 3) Constructing the solution. The framework finally constructs a feasible and efficient solution for the original migration problem with all the solutions to subproblems. Instead of solving the optimization problem with inefficient optimization solvers, the proposed framework can approximate the optimum for a very large problem size. Note that our method may also be of independent interest for other problem settings, such as computation offloading in mobile cloud computing.

The main contributions of this paper are summarized as follows:

1) We tackle the issue of migrating enterprise applications in hybrid cloud infrastructures. Leveraging the characteristics of enterprise applications, we propose a three-stage framework to solve the problem.

2) We evaluate our framework with comprehensive simulations, and experimental results verify that our framework can bring significant benefits to enterprises.

The remainder of this paper is structured as follows. Section II describes the problem. Section III presents our framework. Section IV demonstrates the evaluation results. Section V concludes the paper.

II. Problem Modeling

We describe the Enterprise Application Migration (EAM) problem that maximizes the total cost reduction while ensuring the completion time constraint in this subsection. Mathematically, we model a typical hybrid cloud architecture as a node set $H = H' \cup h_0$, where h_0 represents the on-premise data center and H' represents M public cloud sites located in M geographic regions. The on-premise data center holds the entire application before migration.

The enterprise application is abstracted as a graph $G = (V, E)$, where V denotes the servers involved in the application ($|V| = N$). Each server v_i is associated with a value t_i, representing the average aggregated traffic requests from the Internet. Note that these requests are from users in M regions with different proportions. The edge $(i, j) \in E$ means that two servers v_i and v_j communicate. Each edge is associated with a value $t_{i,j}$, representing the average communication traffic volume between v_i and v_j. We consider a static scenario in this paper, and thus, the values of t_i and $t_{i,j}$ can both be precomputed.

Let $\pi(i)$ be a mapping of the hybrid clouds of v_i. Let $x_{i,\pi(i)}$ denote the *binary indicator*, $x_{i,\pi(i)} \in \{0, 1\}$. $x_{i,\pi(i)} = 1$ implies that v_i is assigned to cloud site $\pi(i)$. The goal is to determine a migration policy $x = \{x_{i,\pi(i)}\}$ which minimizes the total costs subjected to given transaction delay constraint. We adopt the application migration plan model provided by Hajjat in [1] and extend it to fit our hybrid cloud infrastructure– in which more than one public cloud site is involved–in the following.

A. Cost Reduction

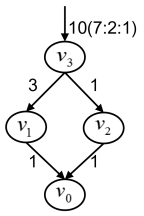
Typically, possessing servers in public clouds is much cheaper than in small- or medium-sized data centers due to Economies of Scale. Some reports claim 80% savings using public clouds versus on-premise private data centers [11]. The actual savings for a specific server depend on the resource requirement of the server, the server renting price of the cloud provider and the operation condition of the on-premise data center. Since we consider a static scenario in this paper, the operation cost benefit of each server v_i using each public cloud $\pi(i)$ is assumed to be a constant, denoted by $\alpha_{i,\pi(i)}$. Thus the total operation cost reduction with migration policy x can be represented as $\sum_{v_i \in V} \alpha_{i,\pi(i)}$. Obviously, the value of α_{i,h_0} equals zero for all servers.

In addition to the operation cost, the Internet communication cost is another financial cost relevant to server migration. This cost mainly depends on the total communication traffic volume and the per-unit Internet communication cost of traffic from the cloud site. Deploying communicating servers to hybrid clouds will change the Internet communication cost. The reasons are twofold: 1) when a server is migrated to a public cloud, the per-unit Internet communication cost will change; 2) when two communicating servers are migrated to two different cloud sites, they must use Internet to communicate, which will also increase the Internet communication cost. Let $\beta_{i,\pi(i)}$ denote the Internet communication cost increment of migrating server v_i to cloud $\pi(i)$ due to the first reason. Let $\beta_{i,j,\pi(i),\pi(j)}$ denote the Internet communication cost increment of migration server v_i and server v_j to two clouds due to the second reason. A linear cost model is used for computing the value of $\beta_{i,\pi(i)}$ and $\beta_{i,j,\pi(i),\pi(j)}$, which matches the business model of multiple cloud providers [1]. Note that the value of $\beta_{i,\pi(i)}$ is not necessarily positive and the value of $\beta_{i,j,\pi(i),\pi(j)}$ is zero when $\pi(i)$ and $\pi(j)$ are the same cloud site. Thus the Internet communication cost increase with the migration x can be formulated as $\sum_{i \in V} \beta_{i,\pi(i)} + \sum_{(i,j) \in E} \beta_{i,j,\pi(i),\pi(j)}$.

The total cost reduction with the migration x is the total operation cost reduction minus the total Internet communication cost increment. For the purpose of brevity, we use a new notation $\gamma_{i,\pi(i)}$ to represent $(\alpha_{i,\pi(i)} - \beta_{i,\pi(i)})$ in the following. Obviously the value of $\gamma_{i,\pi(i)}$ can also be precomputed. Let $\mathcal{H}(x)$ denote the total benefits leveraging the hybrid cloud infrastructure; thus it can therefore be expressed as $\mathcal{H}(x) = \sum_{v_i \in V} \gamma_{i,\pi(i)} - \sum_{(v_i,v_j) \in E} \beta_{i,j,\pi(i),\pi(j)}$.

B. Time Constraint

A workflow is a sequence of operations, which depicts the reaction of the application to a user request. Given a workflow $p : v_i \rightsquigarrow v_j$, an application and a migration policy x, the completion time of p, denoted by $d_p(x)$, can be computed as the sum of all the execution times on the servers (denoted by d_i^e), user access delay (denoted by d_i^a) and the communication time between servers (denoted by $d_{i,j}$). These delays can all be computed according to historic data. The reasons for completion time change due to migration are also twofold: 1) migrating a server will change its user access delay because each server is associated with different proportions of user requests, and moving a server to a region where most of its users are located will decrease user access delays; 2)



Plans	h_0	h_1	h_2	Time	Benefits	
1	All			25	0	A
2		All		30	40	NA
3			All	31	48	NA
4	v_1, v_3		v_0, v_2	28	20	A

Fig. 1. The application and four migration plans with the corresponding cost reduction and completion time.

placing two communicating servers in two different clouds will increase their communication time. The completion time of an application is defined as the completion time of the workflow whose completion time is the maximum among all workflows in G. We use $D(x)$ to represent the completion time with a migration policy x. We define that a feasible migration policy x should never violate the completion time constraints, i.e., $D(x) \leq D$, where D is a constant set by the administrator of the application.

C. Motivation Example

In order to describe the problem more clearly, we give a motivation example in this subsection. The hybrid cloud considered here involves an local cloud (h_0) and two public clouds (h_1, h_2). The users are from three regions (R_0, R_1 and R_2). h_0, h_1 and h_2 are located in regions R_0, R_1 and R_2, respectively. The application to be migrated is abstracted as a Directed Acyclic Graph (DAG), $G = (V, E)$ (in Figure 1), which is a widely adopted web application type. The traffic volume values between two servers are shown in Figure 1. The fractions of user requests from three regions to server v_3 are $7/10$, $2/10$ and $1/10$, respectively. For simplicity, we assume that $\alpha_{i,h_1} = 10, \alpha_{i,h_2} = 12$, and $\forall v_i \in V$. The unit Internet communication costs of the three clouds are the same. The service time of each server is 4. The unit communication time between the two clouds is 3. The unit access times from users in the local region and the non-local region are 1 and 2, respectively. The time constraint is set to 28.

We compare four migration plans in Figure 1. It can be seen that neither plan 2 nor plan 3 are applicable (NA) due to the violation of the time constraint. When all servers are migrated to public clouds (plan 2 and plan 3), the enterprise achieves the maximum benefits, but its time costs are high (30 and 31 respectively). In plan 4, both v_0 and v_2 are migrated to cloud 2, while the others remain in local cloud. It can be seen that when the application is migrated partially, the enterprise gets benefits of 20 while subjects to the time constraint.

III. The Framework

Not surprisingly, solving the EAM problem is NP-hard. In order to solve the EAM problem efficiently, a three-stage framework is proposed in this section. The framework first partitions the large EAM problem into multiple, smaller subproblems. Then, an algorithm based on the dynamic programming approach is proposed to generate a solution for each subproblem. Finally, all the solutions to the subproblems are collected to form an efficient solution to the EAM problem. The details of the framework are described in the following.

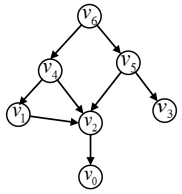


Fig. 2. The sequence of the graph is $(v_0, v_2, v_1, v_3, v_4, v_5, v_6)$.

A. The Applications

A typical multi-tier enterprise application can be decomposed into multiple isolated DAGs [2]. Since solving a smaller problem is much easier than a large one, we first divide the EAM problem into sub-EAMs to facilitate problem solving. For a given graph $G = (V, E)$, we use $G_q = (V_q, E_q), q \in [1, L]$, to denote each of its subgraphs, where L is total number of the subgraphs. The total number of servers in each subgraph G_q is N_q. We can easily construct a subproblem EAM$_q$ of the original EAM by replacing the input graph G with each DAG G_q. The union of all the L solutions of the subproblems is a feasible solution to the EAM.

B. Algorithm for EAM$_q$

In this subsection, we propose an efficient heuristic to solve the EAM$_q$ problem. Before going into detail, we define two node types: the child node and the parent node. For each link (v_i, v_j) in graph G_q, node v_i is called a parent node of node v_j, and v_j is a child node of v_i. We construct a sequence from G_q in the following two steps:

Step 1. Label all nodes with different indexes.

Step 2. Construct a sequence as follows: each time we find a node without any child, put it into the sequence and remove the node from the original graph. Continue the process until no node remains in the original graph. Then the process terminates.

An example of the process is illustrated in Figure 2. Finally, we form a sequence $< v_1, v_2, \cdots, v_{N_q} >$ for DAG G_q. The sequence ensures that each child node should only be listed before its parent node.

Then we derive an Algorithm based on Dynamic Programming (Algo_DP) to solve EAM$_q$ efficiently. A formal description of the algorithm can be found in Algorithm 1.

Let the array $c[i][j][k]$ denote the maximum cost reduction of the subgraph rooted at node v_i when node v_i is assigned to cloud $h_k \in H$ and when the total delay is no larger than j ($j \in [0, D]$). When using the dynamic programming approach to solve a problem, the key insight is that the original problem contains optimal substructures. In the EAM$_q$ problem, when all the child nodes of v_i have unique parent nodes, the assignment of node v_i that obtain the maximum cost reduction can be computed according to the optimal assignment of its subgraph. However, if a child node has multiple parent nodes– since the optimal assignment of the tree rooted at the child node is not the same to each parent node–it will lead to conflict. To tackle this problem, we divide all the nodes into two categories according to its number of parents and design strategies for each kind of node.

Case 1: In the case of node v_i whose child nodes have only one parent node, we use the standard dynamic programming approach to compute the cost tables. The computation method of $c[i][j][k]$ is described in equation (e1). S_i is the number of child node of v_i. $v_{i_{ih}}$ is the hth child node of node v_i. To facilitate the recording of the optimal assignment of each server, we build two arrays during the execution process. The first is $temp\_c[i][j][k]$, which is used to record the assignment of the child nodes of v_i when computes the value of $c[i][j][k]$; The second is $temp\_d[i][j][k]$, which is used to record the average delay of the subgraph rooted at the child node of v_i when computes the value of $c[i][j][k]$. By tracking the two arrays, we can get the final assignment of each server.

$$(e1) \quad c[i][j][k] = \begin{cases} \gamma_{i,k} + \sum_{1\le h\le S_i}\{\max_{1\le k_{ih}\le M}(c[i_{ih}] \\ [j_{ih}][k_{ih}] - \beta_{i,i_{ih},k,k_{ih}})\}, \text{if } \max_{1\le h\le S_i} \\ (j_{ih} + d_i^e + d_{i,k}^a + d_{i,i_{ih},k,k_{ih}}) \le j; \\ \text{no feasible solution, otherwise.} \end{cases}$$

Case 2: Now consider a situation in which at least one of the child nodes of v_i have more than one parent node. For example, node v_i has a child node v_k, which, at the same time, is also a child node of node v_j; node v_k has multiple parent nodes. For each assignment of parent v_i and parent v_j, the equation (e2) will output two totally different optimal assignments for the subgraph rooted at v_k. To avoid conflict, the basic idea is to fix the node with multiple parent nodes before migration. The key problem is where to locate this node. The location of multiparent nodes can be determined by the following means: 1) Random. Choose the location for each node randomly. The advantage of this strategy is its simplicity. 2) Enumeration. Compute the output for each combination of node and location, then choose the one with the maximum benefit under the time constraint. While this method achieves the maximum benefits among the three strategies, its running time overhead is the largest. 3) Greedy. Each time choose the location where the completion time of the subgraph rooted at the node is minimum. This is the strategy adopted in this paper.

C. Time Complexity

The complexity of the Algo_DP(G_q) is $O(N_q * D * M)$, where N_q is the number of nodes of G_q, D is the time constraint, and M is the number of hybrid clouds. Note that when G_q is a tree, as in case 1, the algorithm will output the optimal solution. The complexity of the Algo_EAM is $O(L{*}N_{max}{*}D{*}M)$, where $N_{max} = argmax_{q\in[1,L]}N_q$. It can be seen that the proposed Algo_EAM is a pseudo-polynomial time algorithm. However, the length of each workflow of an enterprise application with the number of tiers given is typically not large, which results in a limited time constraint D. Thus the algorithm can still work well in most cases.

IV. Evaluation Results

The effectiveness and efficiency of our proposed framework is evaluated through extensive simulations. In this section, we will give a detailed summary of our simulation findings.

Algorithm 1 Algo_DP(G_q):

Input: A DAG G_q, hybrid cloud H, a time constraint parameter D.

Output: the server migration policy x_q

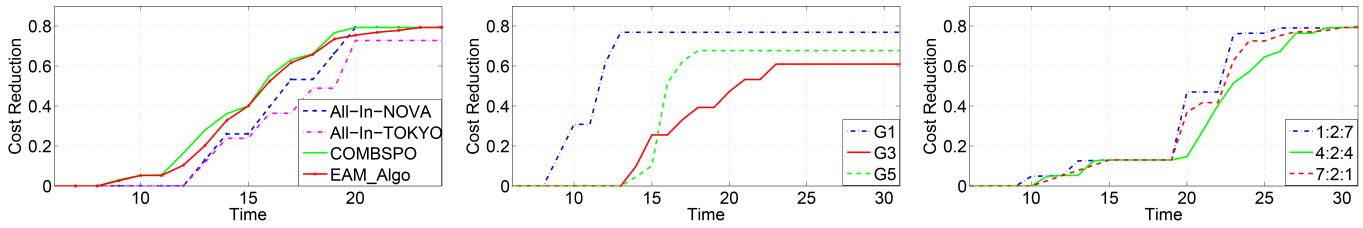
```
Form sequence < v_1, ..., v_{N_q} >.
for v_i ∈< v_1, ..., v_{N_q} > do
    if ExistMultiParentChild(v_i) = 1 then
        Fix the location for v_i, denoted by k_fix.
    for 0 ≤ j ≤ D do
        for 0 ≤ k ≤ M do
            if k ≠ k_fix then
                c[i][j][k] ← 0.
                Continue.
            Compute c[i][j][k] according to (5).
            Record temp_c[i][j][k] and temp_d[i][j][k].
max_c ← min_{1≤k≤M} c[N_q][D][k].
for all s ∈ S do
    Tracing the computing progress of max_c using
    temp_c[N_q][D][M] and temp_d[N_q][D][M] to get the
    server migration policy x_q.
```

A. Evaluation Settings

To evaluate our framework, we use an application that involves 6 DAGs in the simulations. The DAGs are randomly generated. Each DAG involves a number of nodes between 500 and $2K$. The total number of servers in the application is $7K$. Each communicating server pair is associated with unit traffic. Users of the application are from three regions (i.e., R_0, R_1 and R_2). The hybrid cloud considered in the simulation consists of one on-premise data center (h_0) and two public cloud sites (h_1 and h_2). Note that h_0 is located in region R_0, h_1 is located in region R_1 and h_2 is located in region R_2. The on-premise data center holds the entire application before migration. The capacity of both public clouds is assumed to be infinite (large enough to host the migrated applications). We adopt the Amazon EC2 cloud pricing [12] to calculate the cost of running a server in each cloud. We pick two AWS regions to compute the cost; one is in the east of US (i.e., Northern Virginia, use NOVA for short, denoted by h_1) and the other is in Tokyo (denoted by h_2). The cost of server rent in Tokyo is 1.5 times as much as in NOVA. Assume that migrating a server to NOVA can reduce costs by a factor of 7 according to the reports [1, 11]. Clouds are connected via the Internet. The unit communication cost and the delay between each cloud are 0.1 and 1, respectively. The execution time for each server is set to 2.

B. Simulation results

Cost reduction. We compare the cost reduction of our framework (EAM_Algo) with three strategies: All-In-NOVA assigns all servers to the cloud in NOVA; ALL-In-TOKYO assigns all servers to the cloud in Tokyo; COMBSPO is proposed by Altmann *et al.* in [6]. It can obtain the optimal solution using a brute-force approach. The simulation results



(a) (b) (c)

Fig. 3. Cost reduction is defined as the ratio of the total cost reduction after migration and the total cost before migration. (a) The average cost reduction of different strategies under different time constraints; (b) The cost reduction under different time constraints with different input DAGs; (c) The average cost reduction under different different user request fractions and different time constraints. The curve $1:2:7$ represents that 10% requests are from users in R_0, 20% requests are from users in R_1 and 70% requests are from users in R_2.

are displayed in Figure $3(a)$. It can be observed that our framework can bring up to 79.15% cost reduction to enterprises. Besides, the cost reduction obtained by our algorithm is close to that of the optimal solution solved by COMBSPO. Finally, our framework performs better in reducing enterprise costs leveraging the hybrid cloud architecture under controllable time overhead than the other two strategies. On average, EAM_Algo reduces 27.80% and 12.74% more costs compared with ALL-In-Tokyo and All-In-NOVA, respectively.

The benefit-time tradeoff. We vary the time constraint in order to explore dedicated tradeoff between the cost reduction and the time overhead. In this simulation, three DAGs are considered as the test graphs. The simulation results in Figure $3(b)$ show that by varying the value of time constraint, one can obtain a large cost reduction with large time overheads. Choosing a proper value for an application depends on the performance requirement of the application manager.

The effect of user location. We vary the fraction of user requests in different regions to explore the relationship between cost reduction and the location of user request. Figure $3(c)$ shows that migrating applications that have larger percentage of external user to cloud will bring more cost reduction than migrating the ones that have smaller percentage of external user. Besides, the cost reduction of migrating the applications with users evenly distributed in three regions is the least.

Running time. We measured the running time of our algorithm with 6 DAGs of different scale. The running time of our algorithm is at the level of seconds. Specifically, the running time of our algorithm increases linearly with the input graph size. Note that while COMBSPO can get the optimal solution, its running time increases exponentially with the size of the input graph. It cannot compute the solution for a DAG with nodes number larger than $1K$ in ten hours. In other words, our algorithm exhibits better scalability.

V. Conclusion

In this paper, we study the problem of migrating enterprise applications to hybrid cloud for cost benefits maximization. Unlike previous works, this work considers a more general hybrid cloud architecture involving multiple public clouds rather than one. By exploring the features of typical communicating applications, we propose a framework to derive an application migration plan for enterprises. We base our framework on dynamic programming which can achieve a near optimal solution. Besides, the computation of our framework is much faster than the centralized optimal solver. In our framework, a large application graph is partitioned into multiple small DAGs. Then an efficient algorithm is proposed to derive a migration plan for each DAG. The computation of different DAGs can be processed in a parallel manner, resulting in much less overhead in execution time overheads. Simulation results demonstrate that our framework can bring significant cost reduction to enterprises.

Acknowledgement

This work is supported partially by the National Natural Science Foundation of China (NSFC) Major International Collaboration Project 61520106005 and NSFC Project for Innovation Groups 61521092.

References

[1] M. Y. Hajjat, X. Sun, Y. E. Sung, D. A. Maltz, S. G. Rao, K. Sripanidkulchai, and M. Tawarmalani, "Cloudward bound: planning for beneficial migration of enterprise applications to the cloud," in *Proceedings of ACM SIGCOMM*, 2010.

[2] L. F. Bittencourt, E. R. M. Madeira, and N. L. S. D. Fonseca, "Scheduling in hybrid clouds," *IEEE Communications Magazine*, 2012.

[3] X. Qiu, H. Li, C. Wu, Z. Li, and F. C. M. Lau, "Cost-minimizing dynamic migration of content distribution services into hybrid clouds," in *Proceedings of IEEE INFOCOM*, 2012.

[4] F. D. Turck, L. P. Gaspary, and D. Medhi, "Workflow scheduling for saas/paas cloud providers considering two sla levels," in *Proceedings of IEEE NOMS*, 2012.

[5] L. F. Bittencourt and E. R. M. Madeira, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *J. Internet Services and Applications*, 2011.

[6] J. Altmann and M. M. Kashef, "Cost model based service placement in federated hybrid clouds," *Future Generation Comp. Syst.*, 2014.

[7] P. Hoenisch, C. Hochreiner, D. Schuller, S. Schulte, J. Mendling, and S. Dustdar, "Cost-efficient scheduling of elastic processes in hybrid clouds," in *Proceedings of IEEE CLOUD*, 2015.

[8] H. Yuan, J. Bi, W. Tan, and B. H. Li, "Temporal task scheduling with constrained service delay for profit maximization in hybrid clouds," *IEEE Trans. Automation Science and Engineering*, 2017.

[9] "https://aws.amazon.com."

[10] "https://azure.microsoft.com."

[11] Microsoft, "The economics of the cloud," 2010.

[12] "https://awstcocalculator.com."