Virtual Network Embedding with Virtual Topology Pre-Pruning

Cong Wang
School of CCE
Northeastern University at Qinhuangdao
Qinhuangdao, China
e-mail: congw1981@gmail.com

Ying Yuan
School of Information Science and Engineering
Northeastern University
Shenyang, China
e-mail: yuanying1121@gmail.com

Ying Yang
Liren College
Yanshan University
Qinhuangdao, China
e-mail: yyang_ysu@gmail.com

***Abstract*—Efficient network resource utilization is crucial in the field of Virtual Network Embedding (VNE). The diversity of virtual topologies belong to various services providers (SPs) severely affects the efficiency of VNE algorithms and fairness between SPs. To achieve high resource utilization of the physical substrate network and leverage repeatable features in virtual machine deployment, this paper proposes a VNE algorithm with a pre-pruning mechanism to virtual topologies. Before embedding a virtual network of a SP, if the virtual topology is too complex, i.e. has too many virtual nodes or high connectivity, such mechanism will reconfigure the topology to reduce complexity under the premise to guarantee isomorphism. Then the algorithm will felicitously map the virtual network onto substrate network through a Particle Swarm Optimization (PSO) based process. Simulation results show that due to the pre-pruning procedure the algorithm can achieve high revenue to cost ratio and more fairness than traditional VNE algorithms.**

Keywords-network virtualization; virtual network embedding; topological pre-pruning; network resource tuilization

I. INTRODUCTION

As the ingredient of future Internet architecture, Network virtualization allows multiple heterogeneous virtual networks (VNs) to coexist on the same substrate network by sharing the available resources. In Network Virtualization Environment (VNE), Virtual Networks (VNs) have their customized logical topologies and are composed of virtual nodes and links with different resource requirements [1].

The major challenge in network virtualization is the efficient mapping of virtual networks with constraints on both nodes and links onto substrate network, which known as the VNE problem. However, the VN embedding problem is known to be NP-hard even in the offline case. With constraints on virtual nodes and links, the VNE problem can be reduced to the NP-hard multi-way separator problem [2]. Therefore, its solutions mainly rely on heuristic algorithms.

Particularly, in virtual machine deployment, there are some techniques about ram data switch between VMs co-resident on the same single physical machine [3], such repeatable feature can provide a fully transparent and high performance data switch through ram channel instead of the traditional link switch. This provides a progressive way in VNE problem, the major advantage is that such repeatable techniques can save much more bandwidth between VMs of the same virtual network by mapping them on a same physical machine.

To make high utility of the physical resource, if the substrate network cannot support all the virtual networks running at the same time, current VNE algorithms must calculate many redundant requests and map feasible requests based free real-time physical resource and the resource needed by the virtual networks. For this reason, virtual networks with lighter resource requirement are easily to be mapped and bigger ones are hardly to be implement even they are in front of the queue than others. Furthermore, appropriate running VNs should be reconfigured to provide more reasonable idle resources, thus the substrate network's utilization will be further squeezed.

We argue that in addition to keep efficient use of substrate resource, some reconfiguration work can be moved ahead of embedding, which also helpful to reduce overhead on adjusting running VNs. Moreover, the fairness of the virtual network requests should also be guaranteed. So this paper presents a VNE algorithm with virtual topology pre-pruning mechanism, which also can increase utility of the substrate network at the same time.

II. RELATED WORK

Many algorithms have been proposed for the VNE problem. Most works [4-10] formulates the VN mapping as an optimization problem with mapping cost as the objective. A number of constraints imposed by substrate network components are often associated with this optimization problem, such as link bandwidth capacity, node computing

© 2013. The authors - Published by Atlantis Press

resource amount, node memory resource capacity, etc. They can be classified to one-stage VNE algorithm and two stage VNE algorithm. The main different of the two styles is whether the node mapping and the link mapping for a VN is completed at the same time.

In one-stage VNE solution, Lischka et al. [4] proposed a backtracking-based VN embedding algorithm using subgraph isomorphism detection that extensively searches the solution space in a single stage. I. Houidi [5] proposed a distributed algorithm for mapping the VN. However, they only consider an offline version of the problem and assume the substrate network has enough resources to handle all the VN requests.

In two-stage VNE solutions, Minlan Y et al. [6] have provided a two stage algorithm for embedding the VNs. Firstly, they embedding the virtual nodes. Secondly they proceed to map the virtual links using shortest paths and multi-commodity flow (MCF) algorithms in order to increase the acceptance ratio and the revenue. N. Chowdhury et al. [7] introduce co-ordination among the node mapping and link mapping phases to improve the performance of the mapping algorithm. They solve the problem by formulating a mixed integer program and further relaxing the constraints to obtain the linear programming formulation. Xiang, C et al. [8] present a Particle Swarm Optimization based algorithms named VEN-R-PSO, as a heuristic algorithm they did not consider the repeatable node mapping and so there is make no sense to improve the efficiency by adjust the position.

Some other researchers aim at virtual network reconfiguration [9-11] which means to adjust running virtual networks to re-allocate resource of substrate resource. Their ultimate goal is also to improve resource utilization. Butt et al. [8] propose a VN reconfiguration algorithm, which first detects the unmapped virtual nodes and links causing the VN request rejection then migrates these unmapped virtual nodes to one node among the potential candidate nodes. Marquezan et al. [9] propose a distributed self-organizing algorithm to manage the substrate network resources. The main idea is to shorten the physical path embedding a virtual link that overloads at least one substrate link according to its traffic. Fajjari et al. [10] propose a new greedy virtual network reconfiguration algorithm. The main idea is to re-assign canonical star virtual topologies hosted in the overloaded substrate nodes and links.

Mostly, our solution is fall within the VNE field and is also Particle Swarm Optimization based like [11]. Different from previous works discussed above, the algorithm presented in this paper moves some reconfiguration process onto mapping step. In other words, in the algorithm some "big" virtual topologies of some VN requests will be changed before embedding. This mechanism can improve the utilization of substrate network and provide better fairness.

III. VNE Problem Description

Before further discussion, we first introduce general description of the VNE problem. We model the substrate network as a weighted undirected graph and denote it by $G^S = (N^S, E^S, A_N^S, A_E^S)$, where N^S is the set of substrate nodes and E^S is the set of substrate link. We denote the set of loop-free substrate paths by P^S .The notations A_N^S denote the attributes of the substrate nodes, including CPU capacity, storage, and location. The notations A_E^S denote the attributes of the substrate edges, including bandwidth and delay. In this paper, each substrate node $n^s \in N^S$ is associated with the CPU capacity. Each substrate link $e^s(i, j) \in E^S$ between two substrate nodes *i* and *j* is associated with the bandwidth.

Similar to the substrate network, a virtual network can be represented by a weighted undirected graph $G^V = (N^V, E^V, C_N^V, C_E^V)$, where N^V and E^V denote the set of virtual nodes and virtual link , respectively. Virtual nodes and edges are associated with constraints on resource requests, denoted by C_N^V and C_E^V , respectively.

We model the problem of cost efficient reconfiguration and embedding of a virtual network as a mathematical optimization problem using integer linear programming (ILP). The goal is to minimize the usage of the substrate resources. Because node resources cannot be reduced through certain mechanism, however leveraging the advantage of ram switch between VMs host on same physical machine, we can map virtual nodes of same VN onto same machine as far as possible instead of allocation them a physical link capacity for their communication, i.e. try to embed each VNR to the least number of physical machines to save physical bandwidth. Thus the object of our optimization problem just needs to calculate link cost. It is defined as follows:

$$\text{Minimize} \sum_{(i,j)\in P^s} \varphi_{ij}^w \times bw(e^w)$$

$$\text{s.t.} \quad \forall n \in N^V,\ \forall j \in N^S \quad Cpu(j) - \sum_{n^v \to j} Cpu(n^v) \geq Rcpu(n)$$

$$\forall w \in E^v, \quad \forall (i,j) \in p^s,\ w \to p^{ij} \quad \min_{e^s \in p^{ij}} Cbw(e^s) \geq Rbw(w)$$

Where φ_{ij}^w is a binary variable, when the virtual link is mapped onto the physical link it equals to 1, otherwise it equals to 0. The first qualification is node resource constraints, where $\sum Cpu(n^v)$ is the total amount of CPU capacity which has already been allocated; $Cpu(j)$ is the total amount of CPU capacity of the substrate node *j*. The second qualification is link resource constraints, where $\min Cbw(e^s)$ is the minimum bandwidth of links in the path p^{ij} , that means if a virtual link *w* is embedded onto a substrate path p^{ij} , the capacity of each link in this path must be higher than the request bandwidth of virtual link *w*.

Then we can use long-term revenue to cost ratio shown in equation (1) to measure the stand or fall of a VNE algorithm. In the equation, *R* is resource request of virtual network *V*; *C* is real physical resource provided by substrate network to implement the virtual network *V*.

$$R/C = \lim_{T\to\infty} \frac{\sum_{t=0}^{T} R(V(t))}{\sum_{t=0}^{T} C(V(t))} \quad (1)$$

IV. VIRTUAL TOPOLOGY PRE- PRUNING

For the propose to make high utility ratio of the physical resource, VNE algorithms must calculate many redundant requests and map feasible requests based free real-time physical resources and required virtual resources. Then select several but not all from the redundant ones. For this reason, virtual networks with smaller topologies are easily to be accepted and bigger ones are hardly to be mapped even they are in front of smaller ones in the queue. Even some big virtual network is mapped. They may cause some recourse fragment which can result in lower utilization and VN acceptance ratio.

To reduce topological differences among virtual requests in the waiting queue, we introduce a pre-configuration step in VNE solution. With repeatable features, when a virtual network request is added into waiting queue, its topology will be pruned, i.e. converted to a smaller one but logically equivalent to the original one by merging virtual nodes. As shown in Figure 1, the number on node represents CPU request and on link represents bandwidth request.

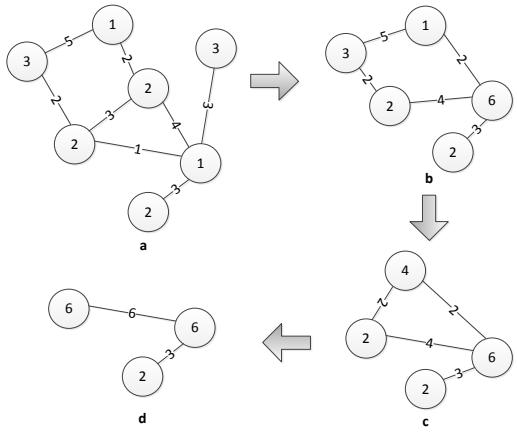

Figure 1. Virtual topology pre-purnning

Another advantage of such pre-configuration is that. We use repeatable mechanism before embedding. Because some virtual nodes are merged into one big virtual node, the virtual links between them can be ignored, thus can save some calculate works in embedding step and is also helpful to improve the mapping acceptance ratio. The main idea of the algorithm is to eliminate the virtual node with high connectivity as far as possible, then we can change the old virtual topology to a logically equivalent and small enough one. The essence of the process is pruning topology in order to reduce part of the virtual links and the complexity of the topology. The details of such mechanism are shown as follows:

Algorithm 1 Virtual topology pre-pruning

1: **For** each VNR in the waiting queue **do**
2: Calculate the number of nodes n_i according to VNR_i
3: **If** n_i>*nodes_threshold* **then**
4: Find the virtual node *vn'* which has biggest connectivity;
5: Sort neighbor nodes of vn' in descending order according to request bandwidth between them and vn' into a queue *sorted_nodes*;
6: Merge nodes from *sorted_nodes* to vn' one by one until the node CPU capacity request or link bandwidth request be equal or greater than the given threshold *C_threshold* or *B_threshold*;
7: If the new virtual topology can be pruned sequentially, go to step 4;
8: **End If**
9: **End For**

In algorithm 1, *nodes_threshold* indicates which virtual topologies should be pruned. *C_threshold* or *B_threshold* indicates allowed maximum CPU capacity and bandwidth value of the merged virtual node and link respectively. Both of them are empirical value. Too big value will cause no solution to embed the virtual topology; on the other hand, small value will reduce the effectiveness of topological pruning.

V. PSO BASED SOLUTION

Standard PSO is not directly applicable to the optimal reconfiguration problem, so we use Disperse Particle Swarm Optimization (DPSO) which is a variant of PSO to solve the VNE optimal problem described in section III.

For a VNR, the search space is *N*-dimensional, where *N* is the number of node of the VN. Then a particle swarm is used to search the optimal position $X^i = [x_1^i, x_2^i, ..., x_N^i]$ to map the virtual nodes of a VN. To VNE problem the position and velocity of particles are determined according to the following velocity and position update recurrence relations:

$$V^{k+1} = \varphi_1(X_p^k - X^k) + \varphi_2(X_g^k - X^k)$$
$$X^{k+1} = X^k \oplus V^{k+1} \quad (2)$$

Where $V^{k+1} = [v_1^i, v_2^i, ..., v_N^i]$ is velocity of a particle, where v_i^k is a binary variable. For each v_i^k, if $v_i^k = 1$, the corresponding virtual node's position in the current VNE solution should be preserved; otherwise, should be adjusted by selecting another substrate node.

We also need to give the relevant discrete quantity operation definitions:

Definition 1: Subtraction of Position $X_* - X$ If X_* and X have the same values at the same dimension, the resulted value of the corresponding dimension is 1, otherwise, the resulted value of the corresponding dimension is 0.

Definition 2: Addition of Multiple $\varphi_1 X' + \varphi_2 X''$ a new velocity that corresponds to a new virtual network embedding solution, where $\varphi_1 + \varphi_2 = 1$. If X' and X'' have

the same values at the same dimension, the resulted value of the corresponding dimension will be kept; otherwise, keep X' with probability φ_1 and keep X'' with probability φ_2.

Definition 3: Addition of Position and Velocity $X^k \oplus V^k$ a new position that corresponds to a new virtual network embedding solution. If the value of v_i^k equals to 1, the value of x_i^k will be kept; otherwise, the value of x_i^k should be adjust by selecting another substrate node.

Eq. (2) is used to update position and velocity in our DPSO algorithm to calculate optimal position for each VN. After every update, particles calculate their fitness according to equation (1). In each round, if the position cannot match the two qualifications the fitness will be set to be $+\infty$. In our solution, link mapping is implement in fitness calculate phase simultaneously, according to the position we use FloydWarshall shortest path algorithm to calculate virtual link mapping between every virtual node pairs, if all shortest path of the node pairs are exit, the fitness is gain by the object function in VNE optimization problem discussed in section III; otherwise, the fitness is set to be $+\infty$. Then for each VNR, after several rounds implementation of the algorithm, a particle swarm can find an optimal mapping solution for it. The detail of our algorithm is shown as follows:

Algorithm 2 PSO based embedding

Input: virtual network requests;
Substrate network G_s;
Output: an optimal solution to embed G_v;
1: **For each** VNR in the waiting queue **do**
2: Generate an undirected graph Gv;
3: Prune the topology of it using algorithm 1;
4: Generate particles for Gv;
5: In G_s, remove nodes and links which capacity is less than minimal request of *Gv*;
6: Randomly initial each Particle's position
7: **For** a preset maximum iterative number **do**
8: **For** each particle **do**
9: Calculate fitness according to VNE object function in section III;
10: Update speed and position according to equation (2);
11: Update global and private best position;
12: **End For**
13: If the global best position not changed in last 15 rounds then break;
14: **End For**
15: If has a solution return the solution;
16: **End For**

In algorithm 2, remove nodes and links which capacity is less than minimal node and link capacity request of VNR is to reduce the search space for particles. The terminate condition of the algorithm is it has implement more than maximum iterative round number which is pre-assigned or the global optimal fitness unchanged in 15 rounds of iteration. The second condition is to improve the efficiency of the algorithm.

VI. PERFORMANCE EVALUATION

We implemented the algorithm presented in this paper using the CloudSim3.0.1 simulator [12] on a high level PC which has one Intel Core i7-3770 CPU and 20G RAM. Both topologies of substrate network and virtual network are generated randomly by a topology generator write in java. We analyzed the performance of the new algorithm in this paper and the VEN-R-PSO algorithm which present in [10].

The parameters of substrate network are: the number of nodes is 60; connectivity in topology generator is 0.2; both node capacity and link bandwidth are set to 100 units. The parameters of virtual network are: the number of nodes is uniform distributed between 2 to 10; connectivity is 0.5; both node capacity and link bandwidth are uniform distributed between 3 to 30 units. Each virtual network's living time uniformly distributed between 100 and 1000 time units. *C_threshold* and *B_threshold* in algorithm 1 are both set to 6, $\varphi_1 = \varphi_2 = 0.5$ in equation (2).

In the experiment, we simulate 1500 VNRs. Each test run 10 times, every implement take about more than 3 hours. For a VNR the max iterative count of particle swarm is 30. If the best position is invariant more than 15 rounds, the algorithm will terminate. When the substrate network accepts about 18 VNRs it reaches full load condition, then the other VNRs should wait to be implement. Because of the connectivity of VN is bigger than substrate network, which may lead VNE solution is hard to be calculated, thus we can measure the extreme effect of the algorithm. Firstly, we plot sample average revenue/cost ratio according to equation (1) of the algorithm with and without pre-pruning and the VEN-R-PSO presented in [10]. Note that because the connectivity of virtual network is bigger than the substrate network, some virtual network cannot be mapped, so we just plot 1000 accept VNs.

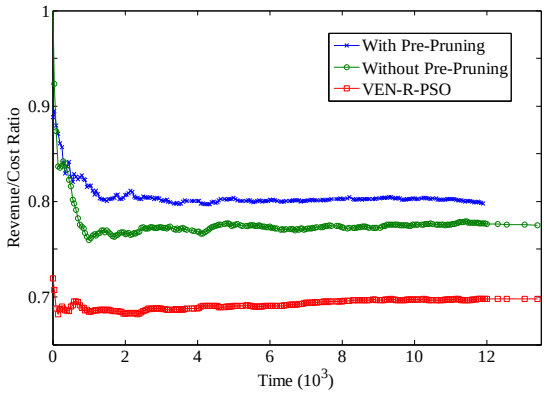

Figure 2. The revenue/cost ratio of each algorithm

From Fig. 2 we can see that due to the repeatable features the revenue/cost ratio of substrate network can significantly be improved. No matter with and without pre-pruning mechanism the algorithm present in this paper always perform better than VEN-R-PSO at the same time. The reason is that repeatable mapping can save substrate bandwidth, many virtual nodes from same VN are mapped on to single substrate node and the links between them are

just instead by ram switch; in VEN-R-PSO, the bottleneck is the link resource limitation, if the substrate nodes which can satisfy the connectivity of a VNR are all full load, the VNR must wait for completion of running ones. In addition, the pre-pruning mechanism can improve the ratio because it can make the topology of big VNs be simpler as much as possible.

Fig. 3 shows the performance on fairness with or without pre-pruning mechanism. It plots the original number of nodes of virtual networks which are accepted at each time unit. Fig. 3 (a) is without pre-pruning and Fig. 3 (b) with it. We can see that pre-pruning can gain a more smooth average number of nodes than without it. That is to say there is little difference in embedding difficulty between big and small topology of virtual network. Through this mechanism we can improve the fairness during VNE. The reason is for bigger virtual topology we pruned it to a logically equivalent but smaller one. Note that though the two topologies ate logically equivalent, but by repeatable features the requirement of virtual link capacity is reduced. Thus some virtual link between merged virtual nodes is ignored. This can reduce mapping difficulty of big virtual networks.

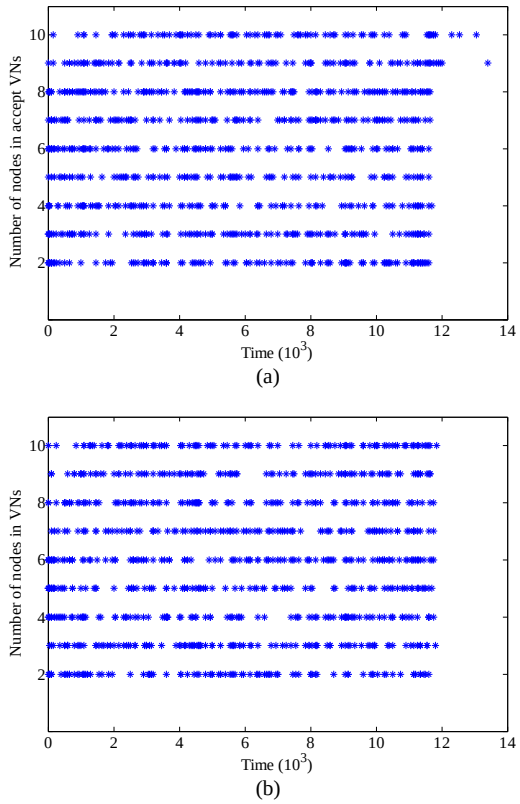

Figure 3. Performance on fairness with or without pre-pruning mechanism

VII. CONCLUSION

This paper introduced a VNE algorithm with virtual topology pre-pruning mechanism, the major characteristics of our work is that we move some reconfiguration work to embedding steps and leverage the advantage of inter ram switch techniques. Such mechanism can achieve higher revenue/cost ratio of substrate network and more fairness of virtual networks during embedding. We also present a whole repeatable DPSO based VNE solution and test the effect of our algorithm. In our future work, we will continually study the pre-configuration mechanism and try to design a distribute algorithm for searching the mapping solution for VNE problems.

ACKNOWLEDGMENT

This work was supported by The Central University Fundamental Research Foundation, under Grant. N110323009, N110323007.

REFERENCES

[1] S. Qing, J. Liao, J. Wang, X. Zhu., Q. Qi, “Hybrid virtual network embedding with K-core decomposition and time-oriented priority,” Proc. IEEE International Conference on Communications 2012 (ICC 12), IEEE Press, Jun. 2012, pp. 2695-2699, doi: 10.1109/ICC.2012.6363761.

[2] M. Chowdhury , M. Rahman, R. Boutaba, “ViNEYard: Virtual Network Embedding Algorithms With Coordinated Node and Link Mapping,” IEEE/ACM Transac. J. Networking, vol. 20(1), pp. 206-219 , February, 2012, doi: 10.1109/TNET.2011.2159308.

[3] J. Wang, K. Wright, K. Gopalan, “XenLoop: a ransparent high performance inter-vm network loopback,” Proc. the 17th international symposium on High performance distributed computing (HPDC 08), ACM, Jun. 2008, pp.109-118, doi: 10.1007/s10586-009-0079-x.

[4] J. Lischka, H. Karl, “A virtual network mapping algorithm based on subgraph isomorphism detection,” Proc. the 1st ACM workshop on Virtualized infrastructure systems and architectures (VISA 2009), ACM, Aug. 2009,pp.81-88, doi: 10.1145/1592648.1592662.

[5] Ines, H., Wajdi, L.,Djamal, Z, “A distributed virtual network mapping algorithm,” Proc. IEEE International Conference on Communications 2008(ICC 2008), IEEE Press , May. 2008, pp. 5634-5640, doi: 10.1109/ICC.2008.1056.

[6] M. Yu, Y. Yi, J. Rexford, M. Chiang, “Rethinking virtual network embedding: substrate support for path splitting and migration,” ACM SIGCOMM. J. Computer Communication Review,vol. 38(2), pp. 17-29, Spril, 2008, doi: 10.1145/1355734.1355737.

[7] Chowdhury, N. M. M. K., Boutaba, R, “Network virtualization: state of the art and research challenges,” EEE Communications Society . J. Communications magazine, vol. 47(7), pp. 20-26, July, 2009, doi: 10.1109/MCOM.2009.5183468.

[8] X. Cheng, Z. Zhang, S. Su., F. Yang, “Virtual Network Embedding Based on Particle Swarm Optimization,” CIE. J. Acta Electronica Sinica,vol. 39(10), pp.2240-2244, October, 2011, doi: 10.1016/j.comnet.2012.01.022.

[9] N. F. Butt, M. Chowdhury, R. Boutaba, "Topology-awareness and reoptimization mechanism for virtual network embedding," Springer. J. LNCD, Vol. 6091, pp. 27-39, May 2010, doi: 10.1007/978-3-642-12963-6_3.

[10] C. C. Marquezan, L. Z. Granville, G. Nunzi and M. Brunner, "Distributed autonomic resource management for network virtualization," Proc. IEEE Network Operations and Management Symposium (NOMS 10), IEEE Press, Apr. 2010, pp. 463-470, doi: 10.1109/NOMS.2010.5488490.

[11] I. Fajjari, N. Aitsaadi, G. Pujolle, H. Zimmermann,"VNR Algorithm: A Greedy Approach For Virtual Networks Reconfigurations," Proc. IEEE Globecom, IEEE Press, Dec. 2011, pp. 1-6, doi:10.1109/GLOCOM.2011.6134006.

[12] CloudSim: A Framework For Modeling And Simulation Of Cloud Computing Infrastructures And Services, http://www.cloudbus.org/cloudsim/ 2013.