INFORMATION TECHNOLOGY JOURNAL

Information Technology Journal 12 (19): 5064-5069, 2013
ISSN 1812-5638 / DOI: 10.3923/itj.2013.5064.5069
© 2013 Asian Network for Scientific Information

Saving Energy in Data Center Networks with Traffic-Aware Virtual Machine Placement

[1]Chen Zhi and [2]Huang Guowei
[1]School of Computer Science and Software Engineering, Tianjin Polytechnic University, 300387, Tianjin, China
[2]College of Computer, Shen Zhen Institue of Information Technology, 518172, China

Abstract: As the rapid development of cloud computing techniques, a large number of data centers have been deployed recently. The power costs of data centers have become a practical issue and have attracted significant attention. Existing works on green data center have focused on computer servers and cooling systems. However, saving network energy also plays an important role on energy efficiency of data centers. In this study, we formally define the Network Power Saving VM Placement Problem (NPS-VMPP), analyze and highlight the resemblance between NPS-VMPP and a variant of Quadratic Assignment Problem (QAP) for Fat-Tree topology data centers. Inspired by this observation, we propose an extended robust tabu search based approach (eRTS) to optimize virtual machine placement and adopt a simple topology-aware heuristic to allocate traffic flow so as to turn off as many unneeded network devices as possible. Experiment results demonstrate the efficacy of this approach.

Key words: Cloud computing, data center, VM placement, energy saving, tabu search

INTRODUCTION

Cloud computing is now the propelling force for the development and deployment of data centers that hold a large number of computer servers and that support a variety of online applications (e.g., Google) as well as infrastructural services (e.g., Hadoop platform for big data). However, the energy cost of data center is also skyrocketing. It has been reported that data center power usage in US doubled between year 2000 and 2006 to nearly 61 billion kWh (1.5% of total US electricity consumption) and is predicted to double again by 2011 to more than 100 billion kWh (US Environmental Protection Agency, 2007). The computer servers and cooling systems account for about 70% of a data center's total power budget and most research efforts focus on making them even more energy efficient. In contrast, the underlying network infrastructure, such as routers and switches, takes up about 10-20% of energy bill and has received little attention.

Conventional data centers usually employ a three-tier topology proposed by Cisco as the common network architecture. At the bottom level (known as the access tier), servers are organized in racks and each server in a rack connects to one Top-of-Rack (ToR) switche. Each ToR switch connects to one (or more) switches at the aggregation tire and each aggregation switch connects with multiple switches at the top level (known as the core tier). As services scale up, poor scalability and insufficient bandwidth inherent in three-tier topology have motivated researchers to propose new data center network architecture, e.g., Fat-Tree (Al-Fares *et al.*, 2008),VL2 (Greenberg *et al.*, 2009) and BCube (Guo *et al.*, 2009).

Recently, some works have focused on reducing the power consumption of network elements in cloud data centers. ElasticTree (Heller *et al.*, 2010) consolidates traffic flows in the data center network onto a small set of switches and links such that unused network elements can be turned off for power saving. But ElasticTree doesn't take VM placement into consideration. Some works have studied VM placement for optimizing network traffic in data centers. Meng *et al.* (2010) investigated traffic-aware virtual machine placement in data center networks. However, this work does not consider network power optimization.

In this study, we solve the power-saving issue of data center networks by exploring traffic-aware VM placement and topology-aware traffic allocation strategy. We formulate a topology independent optimization problem (Network Power Saving VM Placement Problem, NPS-VMPP) to minimize network power cost. To leverage cheap commodity Ethernet switches and to simplify the analysis and discussion, we mainly focus on data center

Corresponding Author: Chen Zhi, School of Computer Science and Software Engineering, Tianjin Polytechnic University, 300387, Tianjin, China

network with the state-of-the-art Fat-Tree topology. By transforming NPS-VMPP to a variant of classical Quadratic Assignment Problem (QAP), we are inspired to exploit robust tabu search which is an efficient meta-heuristic for QAP. So we propose an extended robust tabu search (eRTS) based framework to minimize energy bill and verify its effectiveness with experiments.

PROBLEM FORMULATION

The data center network topology can be modeled as a simple undirected graph G(V,E), where V is the set of vertices and E is the set of edges. There are two types of vertices in V: the physical machines (PM in abbreviation and PMs are denoted by upper letter, e.g., I, J) N, the network switches S and $V = N \cup S$. The edge $(u, v) \in E$ represents a communication link between a server u and a switch v, or between a pair of switches u and v. The bandwidth capacity of link (u, v) is denoted by c(u, v). S_u represents a set of nodes connected to a switch u. Services hosted by the data center demands a set of virtual machines (VM in abbreviation and VMs are denoted by lower letter, e.g., i, j) M. The traffic rate from VM i to j is denoted by d_{ij}.

Let P_u denote the power consumption of switch u and Y_u denote the binary decision variable indicating whether switch u is powered on. We can formally define the Network Power Saving VM Placement Problem (NPS-VMPP) and denote it as Problem P for short in the rest of the study. The objective of Network Power Saving VM Placement Problem is to find a VM placement scheme such that the power consumption of networking devices can be minimized.

Problem P:

$$\text{Minimize} \sum_{u \in S} P_u \times Y_u$$

subject to the following constrains:

$$\pi(i) = I,\ \ \pi_T(I) = i \tag{1}$$

$$\begin{aligned} &\forall i, j \in M, \forall (u,v) \in E, f_{ij}(u,v) \le d_{ij} \\ &\forall I, J \in N, \forall (u,v) \in E, f_{IJ}(u,v) \le d_{\pi_T(I)\pi_T(J)} \end{aligned} \tag{2}$$

$$\forall (u,v) \in E, f(u,v) = \sum_{i,j=1}^{n} f_{ij}(u,v) = \sum_{I,J=1}^{n} f_{IJ}(u,v) \le c(u,v) \tag{3}$$

$$Y_u = \begin{cases} 1 & \text{if} \sum_{v \in S_u} f(u,v) \ne 0 \\ 0 & \text{if} \sum_{v \in S_u} f(u,v) = 0 \end{cases}, \forall u \in S \tag{4}$$

We assume that some VM consolidation mechanism (Nathuji and Schwan, 2007) has already employed so that the number of compact VMs is no larger than the number of PMs, so the term VM in this study represent "compact, virtual" VM. Moreover, we create dummy VMs that consume no resource at all to make it possible to map VMs and PMs on a one-to-one (1:1) basis. As a result, we can define a mapping π from VM to PM: $[1,\ldots i,\ldots,j,\ldots,n] \rightarrow [1,\ldots,i,\ldots,j,\ldots,n]$ and another mapping π_T from PM to VM: $[1,\ldots I,\ldots,J,\ldots,n] \rightarrow [1,\ldots,i,\ldots,j,\ldots,n]$. Constraint 1 states that any virtual machine i must be hosted by a physical server $I = \pi(i)$ and any physical machine I only hosts one virtual machine $i = \pi_T(I)$. $f_{ij}(u, v)$ and $f_{IJ}(u, v)$ represent the traffic flow on link(u,v) imposed by the traffic between VM i and j (i.e., d_{ij}) and between PM I and J, respectively. f(u, v) denotes the total traffic on link (u,v). Constraint (2) implies that the flow is splittable among different links. Constraint 3 ensures that the total traffic along each link must not exceed its bandwidth capacity. Constraint 4 means that when all links connecting to a switch are off, the switch is powered off, otherwise, the switch is powered on.

ALGORITHM DESIGN FOR FAT-TREE TOPOLOGY

In Fat-Tree topology, all switching elements are identical, enabling us to leverage cheap commodity Ethernet switches to deliver scalable bandwidth for large-scale clusters at significantly lower cost than existing techniques. Fat-Tree topology, however, requires more networking devices and thus consumes more and may waste more, energy than other existing network topologies. So in this study we analyze and solve the energy saving for Fat-Tree topology. Notice that although the analysis and algorithm given in this study are based on Fat-Tree, the basic idea and designs can be applied to data center with other hierarchical network topologies (e.g. 3-tier tree, VL2).

A k-ary Fat-Tree is built with k-port switches which are divided into k pods, with k/2 access switches and k/2 aggregation switches in each pod. Each pod is connected with $k^2/4$ core switches and with $k^2/4$ servers. Thus in total, there are 5 $k^2/4$ switches that interconnect $k^3/4$ servers. Figure 1 shows one such network with k = 4. The meanings of the symbols used to describe Fat-Tree scenario are given in Table 1.

We employ a simple topology-aware heuristic (Heller *et al.*, 2010) to allocate traffic flow to different link on each switch. The number of active ToR switches is defined as:

$$|S_{ToR}'| = \sum_{u \in S_{ToR}} Y_u$$

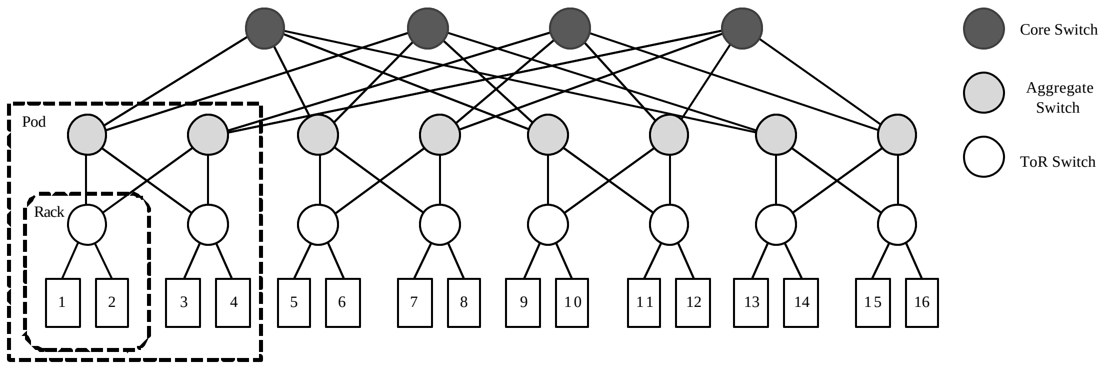

Fig. 1: Fat-Tree architecture with k = 4

Table 1: Key symbols for Fat-Tree scenario and their descriptions

S_{ToR}, S_{Aggr}, S_{Care}	Set of ToR, Aggregation and Core switches, respectively. Note that $S = S_{ToR} \cup S_{Aggr} \cup S_{Core}$
$\|S_{toR}'\|$, $\|S_{aggr}'\|$, $\|S_{Care}'\|$	No. of power-on ToR, Aggregation and Core switches, respectively.
T_{Rock}, T_{Pod}, T_{Care}	Total traffic of ToR, Aggregation and Core switches, respectively. Note that $T_{Rack} \supset T_{Pod} \supset T_{Core}$ and $T_{Rack} = \sum_{i,j=1,\cdots,n} d_{ij}$
$T^{I}_{Rock}(p)$	Internal traffic of Rack p $T^{I}_{Rack(p)} = \sum_{I,J \in Rack(p), I \neq J} d_{\pi_I(I)\pi_I(J)}$
$T^{E}_{Rock}(p)$	External traffic of Rack p $T^{E}_{Rack(p)} = \sum_{I \in Rack(p), J \in V-Rack(p)} d_{\pi_I(I)\pi_I(J)}$
$T^{I}_{Pod}(q)$	Internal traffic of Pod q $T^{I}_{Pod(q)} = \sum_{I,J \in Pod(q), \lfloor \frac{2I}{k} \rfloor \neq \lfloor \frac{2J}{k} \rfloor} d_{\pi_I(I)\pi_I(J)}$
$T^{E}_{Pod}(q)$	External traffic of Pod q $T^{E}_{Pod(q)} = \sum_{I \in Pod(q), J \in V-Pod(q)} d_{\pi_I(I)\pi_I(J)}$

The number of active aggregate switches in pod q can be calculated according to Eq. 5, where c denotes the identical capacity for all the links in a Fat-Tree. So, the total number of active aggregate switches is calculated by Eq. 6. The total number of active Core switches derives from Eq. 7. Consequently, the energy cost of data center network can be derived from the number of active switches:

$$|S_{Aggr\_Pod(q)}'| = \max_{p \in [\frac{(q-1)k}{2}+1, \frac{qk}{2}]} \{T^{E}_{Rack(p)} / c\} \tag{5}$$

$$|S_{Aggr}'| = \sum_{q \in [1, \frac{k^2}{4}]} |S_{Aggr\_Pod(q)}'| = \sum_{u \in S_{Aggr}} Y_u \tag{6}$$

$$|S_{Core}'| = \max_{q \in [1, \frac{k^2}{4}]} \{T^{E}_{Pod(q)} / c\} = \sum_{u \in S_{Core}} Y_u \tag{7}$$

For Fat-Tree topology, the objective function of problem P can be reformulated as problem P'.

Problem P':

$$\text{Minimize} \sum_{u \in S_{ToR}} P_u \times Y_u + \sum_{u \in S_{Pod}} P_u \times Y_u + \sum_{u \in S_{Core}} P_u \times Y_u \Leftrightarrow$$
$$\text{Minimize} |S_{ToR}'| + |S_{Aggr}'| + |S_{Core}'|$$

We also define problem Q (Meng *et al*., 2010).

Problem Q:

$$\text{Minimize} \sum_{i,j=1,\cdots,n} d_{ij} C_{\pi(i)\pi(j)}$$

where, C_{IJ} represents the number of switches on the routing path from PM I to J. With such a definition, the objective function is to minimize the traffic sum perceived by each switch. Obviously, problem Q falls into the category of Quadratic Assignment Problem (QAP) in the Koopmans-Beckmann form (Loiola *et al*., 2007) which is a known NP-hard (Sahni and Gonzalez, 1976) problem.

For Fat-Tree topology, $C_{IJ}^{Fat-tree}$ is calculated as Eq. 8 which implies the Eq. 9. Since:

$$T_{Rack} = \sum_{i,j=1,\cdots,n} d_{ij}$$

so T_{Rack} is a constant given a permutation π.

$$C_{IJ}^{Fat-tree} = \begin{cases} 0 & \text{if } I = J \\ 1 & \text{if } \lfloor \frac{2I}{k} \rfloor = \lfloor \frac{2J}{k} \rfloor \\ 3 & \text{if } \lfloor \frac{2I}{k} \rfloor \neq \lfloor \frac{2J}{k} \rfloor \wedge \lfloor \frac{4I}{k^2} \rfloor = \lfloor \frac{4J}{k^2} \rfloor \\ 5 & \text{if } \lfloor \frac{4I}{k^2} \rfloor \neq \lfloor \frac{4J}{k^2} \rfloor \end{cases} \tag{8}$$

$$\sum_{i,j=1,\cdots,n} d_{ij} C^{\text{Fat-tree}}_{\pi(i)\pi(j)} = T_{\text{Rack}} + T_{\text{Pod}} + T_{\text{Core}} \quad (9)$$

Therefore, good solutions to Problem Q are also good solutions to both Problem P’ and P. Such observation motivated us to adopt effective algorithms for Problem Q to obtain high quality solutions to Problem P.

Although various methods for solving QAP (Problem Q in this study) have been proposed (Loiola *et al*., 2007), there is a general agreement that finding the optimality of QAP problems with size >15 is practically impossible. However, there are some effective meta heuristic algorithms for QAP problem, for example, Robust Tabu Search (RTS) (Taillard, 1991). The resemblance between NPS-VMPP (Problem P) and QAP inspired us to extend Robust Tabu Search to minimize power consumption of data center network.

Any VM placement can be obtained after a serial of transposition of two VMs, regardless of the initial placement. If we randomly select two PMs, I and J and exchange the VMs on them, random transpositions fall into 3 different categories: intra-rack transposition, intra-pod transposition and inter-pod transposition. Intra-rack transposition, exchanging PM I and J within a rack, affects neither internal (T^I_{Rack}) nor external rack traffic (T^E_{Rack}). So, we won't probe this type of permutation (Line 9-10). Intra-pod transposition, exchanging PM I and PM J within a pod, may rearrange internal rack traffic, external rack traffic and internal pod traffic (T^I_{Pod}), leaving external pod traffic (T^E_{Pod}) unchanged. Inter-pod transposition, exchanging PM I and PM J from different pods, adjusts both internal and external traffic of corresponding pods and racks.

Particularly, for an inter-pod transposition between PM I and J, the alteration to external and internal traffic for the rack hosting I can be calculated by the following equations:

$$\Delta T^E_{\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} = \sum_{I'\in N-\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} d_{\pi_T(I')\pi_T(J)} - \sum_{J'\in N-\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} d_{\pi_T(I)\pi_T(J')} + \sum_{J''\in \text{Rack}(\lfloor \frac{2I}{k} \rfloor), J''\neq I} d_{\pi_T(I)\pi_T(J'')} - \sum_{I''\in \text{Rack}(\lfloor \frac{2I}{k} \rfloor), I''\neq J} d_{\pi_T(I'')\pi_T(J)}$$

$$\Delta T^I_{\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} = \sum_{I'\in \text{Rack}(\lfloor \frac{2I}{k} \rfloor), I''\neq J} d_{\pi_T(I')\pi_T(J)} - \sum_{J''\in \text{Rack}(\lfloor \frac{2I}{k} \rfloor), J''\neq I} d_{\pi_T(I)\pi_T(J'')}$$

The alteration to external and internal traffic for the pod hosting I can be calculated by the following equations:

$$\Delta T^I_{\text{Pod}(\lfloor \frac{4I}{k^2} \rfloor)} = \sum_{I''\in \text{Pod}(\lfloor \frac{4I}{k^2} \rfloor)-\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} d_{\pi_T(I'')\pi_T(J)} - \sum_{J''\in \text{Pod}(\lfloor \frac{4I}{k^2} \rfloor)-\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} d_{\pi_T(I)\pi_T(J'')}$$

$$\Delta T^E_{\text{Pod}(\lfloor \frac{4I}{k^2} \rfloor)} = \Delta T^E_{\text{Rack}(\lfloor \frac{2I}{k} \rfloor)} - \Delta T^I_{\text{Pod}(\lfloor \frac{4I}{k^2} \rfloor)}$$

We omit the cases for the rack and pod hosting PM J and the case for intra-pod transposition, because they can be easily derived from the above equations.

Our proposed algorithm which is extended from the framework of Robust Tabu Search and thus denoted as eRTS, is described in Algorithm eRTS. The purpose of eRTS is to probe different VM placements and find out the one π with minimal energy cost P_{best}.

Algorithm eRTS takes a random VM placement π’ as input and initiates accordingly (Lines 1-4) and then iterates the processes of probe (Lines 6-20) and update (Lines 24-32).

During the initiation phase, we calculate the current traffic sum S_{current} and the current power consumption P_{current} according to the methods mentioned above. For VM placement problem, appropriate move strategy is to transpose two VMs on different PMs. Each move is associated with movement cost which is measured, in this study, by the power difference $\Delta P_{I,J}$ and the traffic sum difference $\Delta S_{I,J}$ if VMs on PM I and J are transposed. The impact on network power cost of a move depends on the adjustment to both external and internal traffic for the corresponding rack and pod and can be quantified accordingly. TabuList(I,j) maintains the tabu period for VM j to be assigned to PM I.

During the probe phase, the procedure selects a best-quality solution π’ among a subset of the neighbors obtained by probing qualified moves. In case of two movements with the same power consumption, we use traffic sum to break the tie (line 13).

We define a set of Boolean variables to facility the understanding of the update criteria. B_{imp} (line 13) indicates whether the current movement is better than any other movements in term of power consumption and traffic sum. B_{auth} (line 12) is true if the current movement is not forbidden by tabu list.

Classical aspiration function allows a tabu move to be selected if it yields a better solution. According to Taillard (1991), longer term diversification process is beneficial for problem with very heterogeneous flows, as our problem is. Thus, a move will trigger aspiration and be chosen, irrespective of its solution quality, if the exchange places both VMs in PMs that have not been examined during the last TA iterations. We define $B_{\text{asp\_t}}$ (line 14) to account for both aspiration criteria. $B_{\text{asp\_a}}$ (line 6 and 15) indicates whether aspiration has been triggered in the current iteration.

Therefore, the unified update criteria consist of three conditions (1) Meeting the aspiration criterion for the first time (i.e., triggering no aspiration yet), (2) Meeting the aspiration criterion and yielding better improvement over any other movements and (3) Qualifying for the movement and yielding better improvement over any other movements.

During the update phase, the current solution is substituted by the newly selected one and the tabu list and the movement cost are updated correspondingly. In order to escape from local optimum, the reverse moves (assigning VM i to PM J and assigning VM j to PM I) are put into tabu list and forbidden for a period of time which is selected randomly between 0.9n and 1.1n. Readers could refer to Glover (1990) and Taillard (1991) for more details on Tabu Search and robust tabu search.

In summary, the complexity of Algorithm eRTS is O(iter_num.n^2).

PERFORMANCE EVALUATION

We have implemented eRTS algorithm using C/C^{++} and Python to evaluate its performance. eRTS is compared with the state-of-the-art network power saving algorithm ElasticTree. The results demonstrated in the study are average of multiple runs. Based on the traces obtained by Meng *et al.* (2010), the traffic distribution for individual VMs is highly uneven. A relatively large proportion of VMs have a relatively small average rate while a relatively small proportion of VMs have a relatively higher average rate. To capture the essence of traffic characteristic, we used the partitioned traffic model in the experiment. Under this model, each VM which belongs to a partition of VMs, sends traffic only to other VMs in the same partition, with the pairwise traffic rate following a normal distribution with variance of 0.75. The proportion of VMs within partition of large traffic rate varies from 2-40%. Since, Fat-Tree comprises homogeneous commodity GigE switches, we use the number of active switches as the indicator of network power consumption. In addition, important parameters of Fat-Tree topology are summarized in Table 2.

Table 2: Experiment parameters for fat-tree topology

k	PM/VM No.	Rack size	Pod size	Pod No.	ToR No.	Aggr No.	Core No.
16	1024	8	64	16	128	128	64

Algorithm eRTS

Input: Initial VM placement π'
Output: Optimal VM placement π, optimal energy cost P_{best}

```
1:  $\pi = \pi'$
2:  Calculate $S_{current}$, $P_{current}$, $S_{best} = S_{current}$, $P_{best} = P_{current}$
3:  Calculate movement cost $\Delta P_{I,J}, \Delta S_{I,J}, \forall I, J \in [1,n], I < J$
4:  Initiate $TabuList(I, j) = -\infty, \forall I, j \in [1,n]$
5:  For i =1 to iter_num do
6:   $I = J = -1, \Delta P_{min} = \Delta S_{min} = \infty$  $B_{asp\_a}$ = false
7:   For I' =1 to n-1 do
8:   For J' =I'+1 to n do
9:   If: $\lfloor \frac{2I'}{k} \rfloor = \lfloor \frac{2J'}{k} \rfloor$ then //I' and J' in the same rack
10:  Continue
11:  End if
12:  $B_{auth}$ = $TabuList(I', \pi'_T(J)) < i$ or $TabuList(J', \pi'_T(I)) < i$
13:  $B_{imp}$ = $\Delta P_{I',J'} < \Delta P_{min}$ or ( $\Delta P_{I',J'} = \Delta P_{min}$ and $\Delta S_{I',J'} < \Delta S_{min}$ )
14:   $B_{asp\_t}$ = $TabuList(I', \pi'_T(J')) < i - T_A$ or $TabuList(J', \pi'_T(I')) < T_A$ or
($P_{current}+\Delta P_{I', J'}<P_{best}$) or (($P_{current}+\Delta P_{I', J'} = P_{best}$) and ($S_{current}+\Delta P_{I', J'}<S_{best}$))
15:  $B_{asp\_a}$= $B_{asp\_a}$ or $B_{asp\_t}$
16:   if ($B_{imp}$ and $B_{auth}$) or ((not $B_{asp\_a}$) and $B_{asp\_t}$) or ($B_{asp\_t}$ and $B_{imp}$) then
17:   $I = I', J = J', \Delta P_{min} = \Delta P_{I',J'}, \Delta S_{min} = \Delta S_{I',J'}$
18:  End if
19:  End for // J'
20:  End for // I'
21:  If I = = -1 then // All moves are in TabuList
22:  Continue
23:  Else
24:  Transpose $(\pi'_T(I), \pi'_T(J))$
25:  $S_{current} = S_{current}+\Delta S_{I', J'}$, $P_{current} = P_{current}+\Delta P_{I', J'}$
26:   TabuList (I, $\pi'_T(J)$) = I+Random (0.9n, 1.1n), TabuList (J, $\pi'_T(I)$)
= I+Random (0.9n, 1.1n)
27:  Update $\Delta P_{I,J}, \Delta S_{I,J}, \forall I, J \in [1,n], I < J$
28:  If $P_{current} < P_{best}$ then
29:  $S_{best} = S_{current}$, $P_{best} = P_{current}$, $\pi = \pi'$
30:  Else if ($P_{current} = P_{best}$ and $S_{current} < S_{best}$) then
31:  $S_{best} = S_{current}$, $\pi = \pi'$
32:  End if
33:  End for //iteration i
34:  Return $\pi$
```

Figure 2 shows the number of active network switches of eRTS and ElasticTree under different traffic intensity. We can observe that the number of active network switches grows up as the proportion of heavy

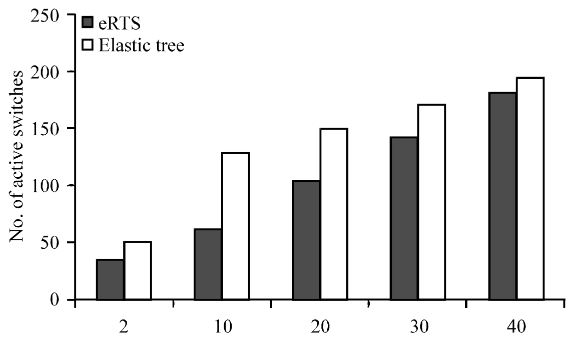

Fig. 2: No. of active switches under partitioned traffic model

traffic partition increases for both eRTS and ElasticTree. This result complies with the desirable property that fewer traffic volume requires less active switches. In general, eRTS outperforms ElasticTree under all circumstances. In particular, the performance advantages of eRTS over ElasticTree are relatively significant when the traffic distribution can be better optimized by VM placement, e.g., when the partition proportion is 10, 20 or 30%. However, the improvement is less prominent when the partition proportion is 2 or 40%. The reason is that the optimization space for traffic distribution is relatively small when the traffic variance is very small or very large. Consequently, eRTS is more beneficial to data centers with heterogeneous traffic flows among VMs. In addition, eRTS only uses approximately 11% of the switches when the partition proportion is 2 and about 66% of the switches when the partition proportion is 40%.

CONCLUSION

Based on the insight that an appropriate VM placement can localize large portion of traffic and thus reduce load at high-level switches, this study explores the idea of manipulating VM placement to save power cost in modern data center networks. We formulate the Network Power Saving VM Placement Problem (NPS-VMPP) which is network topology independent. As a preliminary work towards general approaches for network power saving, we focus on data center network with Fat-Tree topology in this study. By transforming NPS-VMPP to a variant of QAP, we propose a framework based on extended robust tabu search (eRTS) to optimize VM placement and thus minimize energy consumption of network devices. Compared with another state-of the art network power saving algorithm ElasticTree, eRTS can substantially cut the network energy bill especially for services with intermediate and heterogeneous traffic load. In the future, we are going to extend this approach to accommodate the other modern data center network topologies.

RREFERENCES

Al-Fares, M., A. Loukissas and A. Vahdat, 2008. A scalable, commodity data center network architecture. Proceedings of the ACM SIGCOMM, August 17-22, 2008, Seattle, Washington, USA., pp: 63-74.

Glover, F., 1990. Tabu search: A tutorial. Interfaces, 20: 74-94.

Greenberg, A., J.R. Hamilton, N. Jain, S. Candula and C. Kim *et al*., 2009. VL2: A scalable and flexible data center network. Proceegings of the ACM SIGCOMM Computer Communication Review, August 17-21, 2009, Barcelona, Spain pp: 51-62.

Guo, C., G. Lu, D. Li, X. Zhang and Y. Shi *et al*., 2009. BCube: A high performance, server-centric network architecture for modular data centers. Proceedings of the ACM SIGCOMM Computer Communication Review, August 17-21, 2009, Barcelona, Spain, pp: 63-74.

Heller, B., S. Seetharaman, P. Mahadevan, Y. Yiakoumis, P. Sharma, S. Banerjee and N. McKeown, 2010. ElasticTree: saving energy in data center networks. Proceedings of the 7th USENIX conference on Networked systems design and implementation, April 28-30, 2010, San Jose, CA, USA, pp: 1-17.

Loiola, E.M., N.M.M. de Abreu, P.O. Boaventura-Netto, P. Hahn and T. Querido, 2007. A survey for the quadratic assignment problem. Eur. J. Operat. Res., 176: 657-690.

Meng, X., V. Pappas and L. Zhang, 2010. Improving the scalability of data center networks with traffic-aware virtual machine placement, Proceedings of the 29th IEEE Conference on Information Communications (INFOCOM), March 14-19 ,2010, San Diego, CA., pp: 1-9.

Nathuji, R. and K. Schwan, 2007. Virtual power: Coordinated power management in virtualized enterprise systems. Proceedings of the 21thACM SIGOPS Operating Systems Review, October 5, 2007, ACM New York, pp: 265-278.

Sahni, S. and T.L. Gonzalez, 1976. P-complete approximation problems. J. Assoc. Comp. Machin, 23: 555-565.

Taillard, E., 1991. Robust taboo search for the quadratic assignment problem. Parallel Comput., 17: 443-455.

US Environmental Protection Agency, 2007. Report to Congress on Server and Data Center Energy Efficiency Public Law 109-431. ENERGY STAR Program, August, 2007. http://hightech.lbl.gov/documents/data_centers/epa-datacenters.pdf