Elastic Scaling of Virtual Clusters in Cloud Data Center Networks

Shuaibing Lu[a,b], Zhiyi Fang[a], and Jie Wu[b]
[a]College of Computer Science and Technology, Jilin University
[b]Department of Computer and Information Sciences, Temple University

Outline

- Background
- Problem Formulation
- Single Virtual Cluster Scaling
- Multiple Virtual Cluster Scaling
- Online Multiple Virtual Cluster Scaling
- Evaluation
- Conclusion

Background

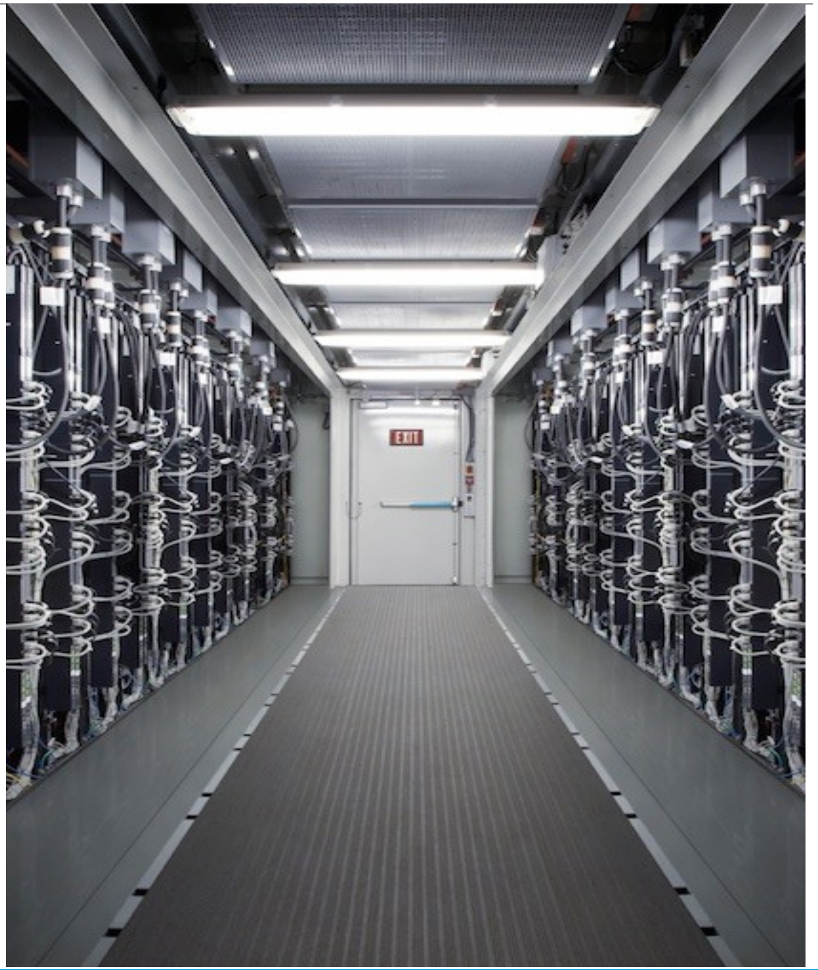
- **Cloud Data Center Networks**

  Supporting cloud-based applications for large enterprises

- **Virtual Cluster Placement**

  Solving the resource utilization problem in a cloud DCN.

- **Motivation**
  - Balancing the allocation on physical resource to virtual clusters.
  - Guaranteeing both computation and communication demands for users.



Problem Formulation

- **Definition**
  - **Data Center Network:** Fat-tree.
  - **Virtual Cluster (VC):**
  $$V_i = \langle N_i, B_i \rangle$$
  - **Hose Model:**
  $$f_i(\cdot) = min\{x, N_i - x\} \cdot B_i$$
  - **Communication Cost:**
  $$m(V_i) = \sum_{j=1}^{k} |T_{S_{ij}}| \cdot H_j \cdot \gamma$$

$T_{S_{ij}}$: denotes the total amount of VMs under the subtree S_{ij} of V_i;
H_j: the hops between PMs that holding the VMs of V_i;
γ: is a constant value which denotes the communication cost between each pair of VMs in V_i;
$f_i(\cdot)$: communication demand;

Problem Formulation

- **Elasticity (E): $\boldsymbol{min\{E_M, E_L\}}$**
  - E_M : minimum percentage of available slots among PMs of V_i.
  - E_L : minimum percentage of available bandwidth among all PLs.
- **Challenges**
  - Balancing $\boldsymbol{E_M}$ and $\boldsymbol{E_L}$ to maximize $\boldsymbol{E}$.
  - Trade-off between $\boldsymbol{E}$ and the $\boldsymbol{m}$.

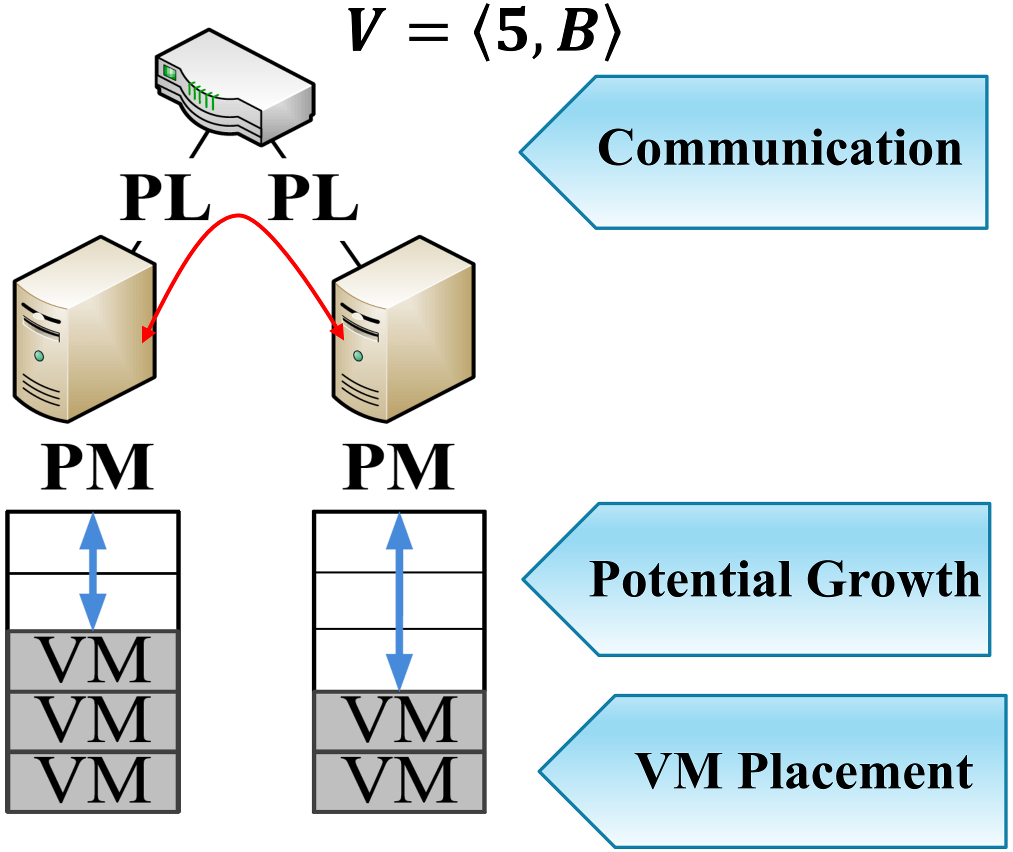


Fig. 1. Fat-tree and Virtual Cluster.

Problem Formulation

- **Problem:** Determine the placement for the scaling VCs.

$$\boldsymbol{V_i = \langle N_i, B_i \rangle -> V_i = \langle N_i + N_i', \delta B_i \rangle}$$

- **Objective:** Maximize $\boldsymbol{E}$ for V_i under the constraints;

$$maxmize\ E = \min\{E_M, E_L\}$$

subject to

$$0 \le m(V_i) \le \Phi_i$$

$$C_i^* + C_i' \le C_i$$

$$f_i(\sum_{C_i \in S_{ij}} (C_i^* + C_i')) \le L_{ij}$$

Notations:

$m(\cdot)$: communication cost;

C_i: PM capacity;

L_i: PL capacity;

Single Virtual Cluster Scaling (VCS)

- **Step 1:** Initialize Φ_i, S_{ij} and $R_{S_{ij}}$;
- **Step 2:** Update the locality S'_{ij} based on Φ_i.
- **Step 3:** Hierarchically place N'_i VMs into PMs into $T_{s_{ij}}$ based on S'_{ij};
  - Update PLs according to the scaling request $B_i \rightarrow \delta B_i$;
  - Update PMs according to the scaling VMs $N_i \rightarrow N_i + N'_i$;

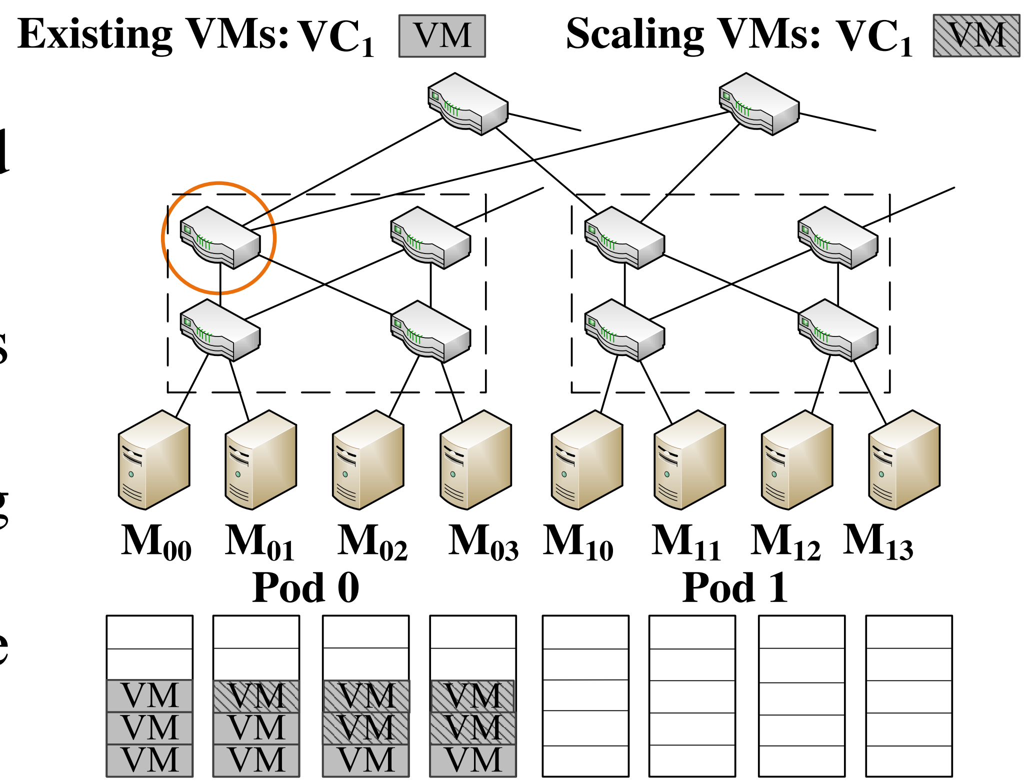


Fig. 1. An example of different placements for single virtual cluster scaling.

Multiple Virtual Cluster Scaling (MVCS)

- **Problem:** $V = \{V_1, V_2, \ldots, V_\varpi\}$
- **Objective:** Maximize over time elasticity in time period $[0, T]$;

$$\boldsymbol{maxmize\ E = \sum_{i=0}^{T} E_i}$$

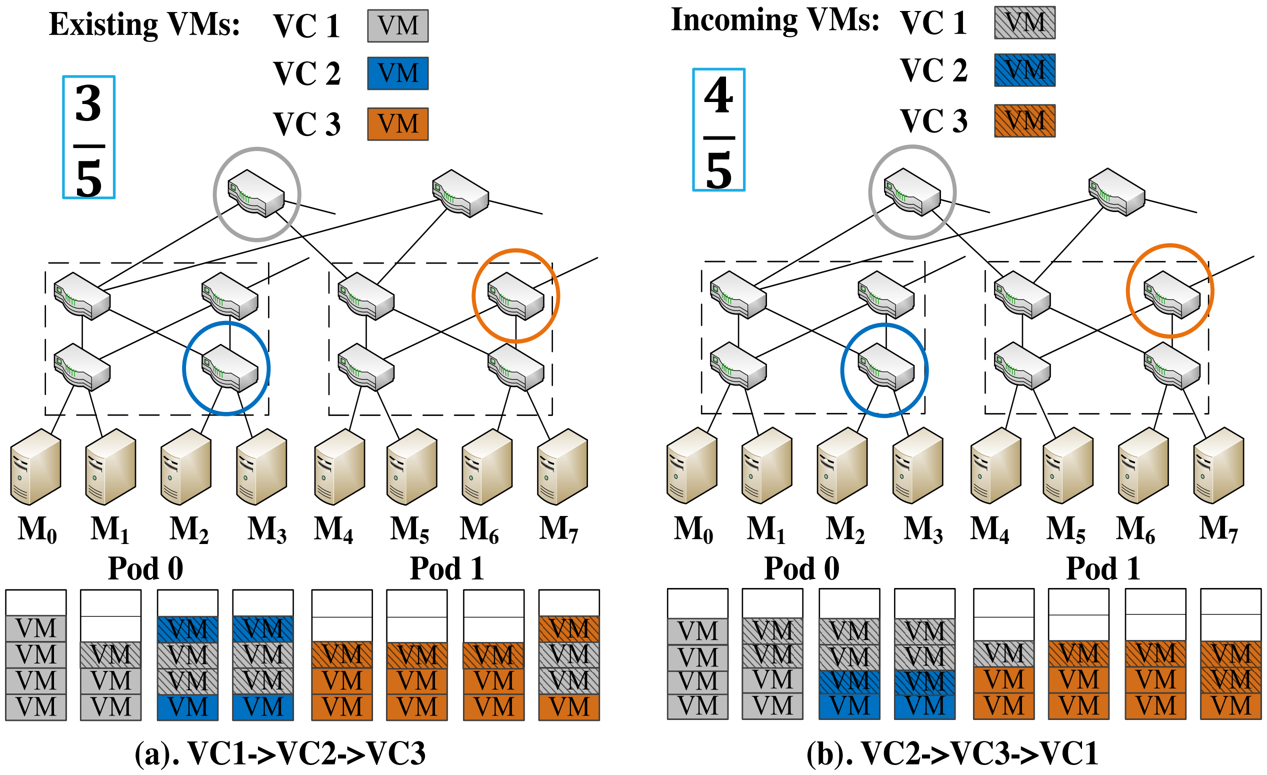


Fig. 2. An example of different placements for multiple VCs scaling.

Multiple Virtual Cluster Scaling (MVCS)

- **Problem:** $V = \{V_1, V_2, \dots, V_\varpi\}$
- **Objective:** Maximize over time elasticity in time period $[0, T]$;

Step 1: Initialize Φ for each V.

Step 2: Calculate each scaling ratio $\rho_i = \frac{V_i}{Rs_{ij}}$.

Step 3: Place the VCs prioritize in the ascending order of scaling ratio ρ_i.

Online Multiple Virtual Cluster Scaling (OMVCS)

- **Problem:** Online condition for the multiple VCs scaling;
- **Objective:** Maximize the over time elasticity in time period $[0, T]$;

Step 1: Estimate the fluctuating mean based on Bayesian parameter estimation;

Step 2: Calculate the future scaling ratio ρ_i^*;

Step 3: Relocate the locality for V_i based on ρ_i^*;

Step 4: Sort VCs in the set V to V' by localities $i = \arg min_i S_{ij}^*$;

Step 5: For VCs with the same locations in the order of ascending scaling ratio $i = \arg min_i \rho_i^*$;

Evaluation

■Single Virtual Cluster Scaling

- **Compare Algorithm: Equally Scaling (ES)** and **Greedy Scaling (GS)**;

Equally Scaling (ES): scaling request of V_i is evenly divided into several pieces depending on the amount of PMs in the sub-tree.

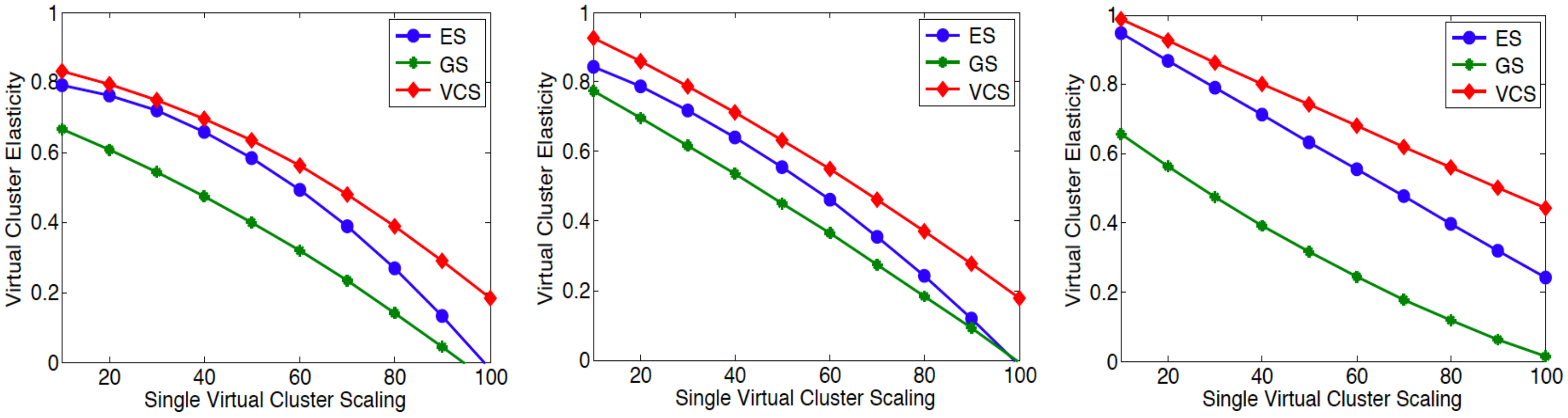
Greedy Scaling (GS): scaling request of V_i for the PMs depends on the amount of available resource in the sub-tree.

- **Setting:** The number of the switches' ports: θ= 4 , θ= 6, θ= 8;

Evaluation

Conclusion:

- The elasticity of the scaling VC depends on the architectures of the fat-tree.
- The elasticity for the scaling VC depends on various placement algorithms, 25% improvement for ES, 11% improvement for GS.



(a) $\theta = 4$ (b) $\theta = 6$ (c) $\theta = 8$

Fig. 3. The elasticity for single VC scaling under various Fat-trees.

Evaluation

■Multiple Virtual Cluster Scaling

- **Compare Algorithm:**
  - ➢Random Schedule Scaling (RSS);
  - ➢Decreasing Schedule Scaling (DSS);
  - ➢Increasing Schedule Scaling (ISS);
- **Setting:**
  - ➢The number of the switches' ports : $\theta = 4$, $\theta = 6$, $\theta = 8$;
  - ➢The VMs of the VCs scaled are evenly distributed between 0 and 50;

Evaluation

■Conclusion

- The volatility of the multiple scaling VCs is stable.
  - As shown in Fig. 4, the mean value of under are marked by red lines, which are close with each other under different algorithms.
- The over-time elasticity for the multiple VCs depends on the scheduling order.
  - MVCS has the best performance in the over-time elasticity.

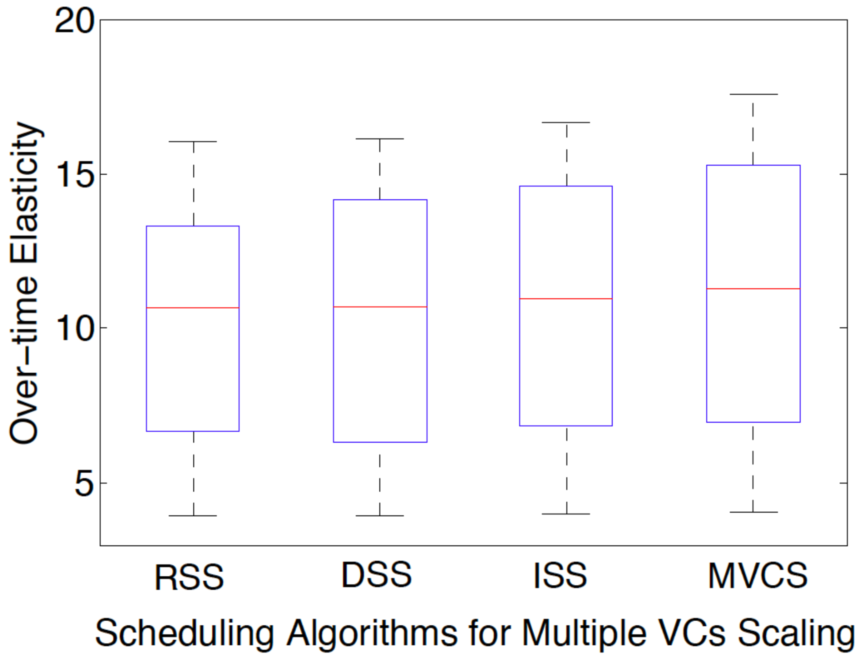


Fig. 4. The elasticity for multiple VCs scaling.

Evaluation

■Online Multiple Virtual Cluster Scaling

- **Compare Algorithm:** online multiple scaling without prediction.
- **Setting:**
  - ➢The number of the switches' ports θ= 4 , θ= 6, θ= 8, θ= 12;
  - ➢Scaling amount of VCs are randomly determined by the tenants;
  - ➢Set scaling frequency to 1, each time slot has to process the scaling or releasing requests.

Evaluation

■Conclusion

- When the size of the Fat-tree is not very large (θ= 4 and θ = 6), the advantage of online scheduling with prediction is not obvious.
- When the size of the Fat-tree is scaling, such as $\theta = 8$, $\theta = 10$ and $\theta = 12$, the gap between these two solutions will increase with the scale of the Fat-tree.

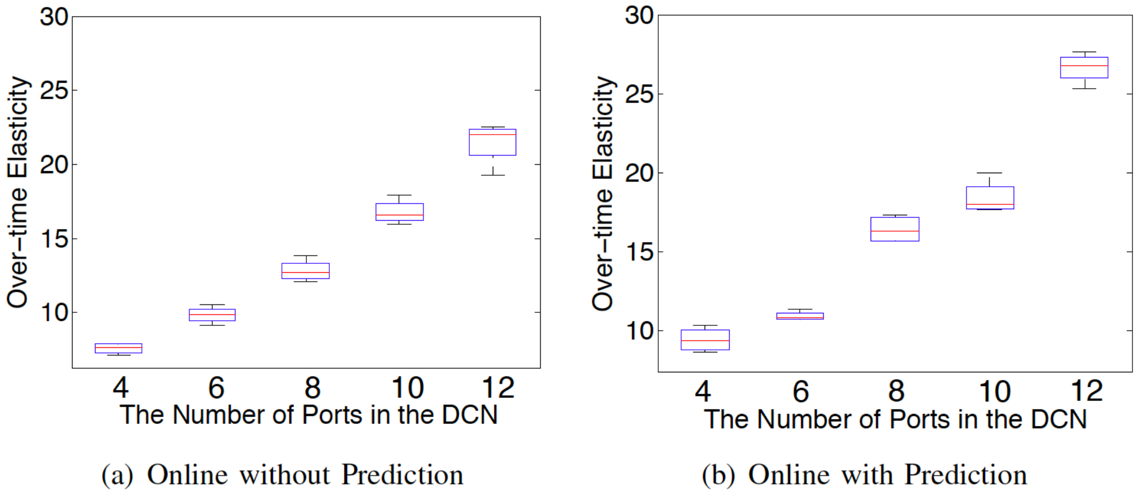


Fig. 5. The elasticity for online multiple VCs scaling.

Conclusion

- We first show that there is a trade-off between elasticity and the communication cost for VC scaling problem.
- We propose an algorithm, VCS, for the scaling request of an existing VC under the constraints of resource and communication costs;
- We extend the single VC scaling placement problem into multiple VCs and prove that it is an NP-hard problem.
- We propose MVCS and OMVCS algorithms for both offline and online cases;
- Extensive simulations demonstrate that our elastic VCs scaling placement schemes outperform existing state-of-the-art methods in terms of elasticity in the DCN.

Thank you very much!