Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement

Xiaoqiao Meng, Vasileios Pappas, Li Zhang
IBM T.J. Watson Research Center

Presented by: Peter Izsak

December 8, 2010
236635 - On the Management and Efficiency of Cloud Based Services
INTRODUCTION

- Modern virtualization Data Centers are hosting a wide spectrum of applications.
- Bandwidth usage VMs is rapidly growing.
- Scalability of data center networks becomes a concern.
- This issue has already attracted attention, techniques suggested:
  - Rich connectivity at the edge of the network
  - Dynamic routing protocols
  - ...
INTRODUCTION

This paper tackles the scalability issue from a different perspective, by optimizing the placement of VMs on host machines.
INTRODUCTION

This paper tackles the scalability issue from a different perspective, by optimizing the placement of VMs on host machines.

Normally, VM placement is decided by Capacity Planning tools:

- VMware Capacity Planner, IBM WebSphere CouldBurst, Novell PlateSpin Recon etc.
- These tools focus on CPU, RAM and power consumption consolidation but ignore network resource consumption.
DATA CENTER TRAFFIC PATTERN EXAMINATION

To better understand the traffic patterns in a data centers, we examine traces from two data-center-like systems:

- DWH hosted by IBM Global Services - server resource utilization from hundreds of server farms.
- An “unnamed” server cluster with hundreds of VMs - aggregate traffic of 68 VMs

The traces were collected over the period of 10 days
DATA CENTER TRAFFIC PATTERN EXAMINATION

Based on the collected traces, the following trends were identified:

1. Uneven distribution of traffic volumes from VMs
   While 80% of VMs have average rate less than 800 KBytes/min, 4% of them have a rate x10 higher.

(a) CDF of mean traffic rate
Data Center Traffic Pattern Examination

2. Stable per-VM traffic at large timescale

(b) Distribution of (mean, standard deviation)
3. Weak correlation between traffic rate and latency

Fig. 2. Traffic matrix  
Fig. 3. Latency matrix
Network Architectures

Current data centers follow to a great extend a common network architecture, known as the three-tier architecture:

1. **Access tier** - each server connects to 1 or 2 access switches.
2. Each access switch connects to 1 or 2 switches at the **Aggregation tier**.
3. Each aggregation switch connects with multiple switches at the **Core tier**.
Network Architectures
**NETWORK ARCHITECTURES**

![Diagram of VL2 network architecture](image)
**Network Architectures**

![Fat-Tree Diagram]
**NETWORK ARCHITECTURES**

$\text{BCube}_0 = n$ servers with 1 $n$-port switch.

$\text{BCube}_k = n \text{BCube}_{k-1}$ connected with $n^k$ $n$-port switches.
TVMPP PROBLEM DEFINITION

- We refer to the problem as a VM to slot assignment. A host can accommodate multiple VMs.
- We consider a scenario where there are $n$ VMs and $n$ slots.
- We assume the routing is static and single-pathed.
- $C_{i,j} =$ Communication cost between slot $i$ and $j$
- $D_{i,j} =$ Traffic rate from VM $i$ to $j$
- $e_i =$ External traffic rate from VM $i$
- $g_i =$ Communication cost between VM $i$ and the gateway
- For any assignment of VMs to slots, there’s a permutation function $\pi : [1, \ldots, n] \rightarrow [1, \ldots, n]$
**Problem Definition**

TVMPP’s main goal is to find a $\pi$ to **minimize** the following objective function:

$$\sum_{i,j=1,...,n} D_{ij} C_{\pi(i)\pi(j)} + \sum_{i,j=1,...,n} e_{i\pi(i)}$$

We can also define it as:

$$\min_{X \in \Pi} tr(DX^T C^T X) + eX^T g^T$$

- $tr(A) = \sum_i A_{ii}$
- $\Pi$ is the set of all valid permutation matrices
- $X$ is a permutation matrix: $X_{ij} \in \{0, 1\} (\forall i, j), \sum_{j=1}^n X_{ij} = 1(\forall i), \sum_{i=1}^n X_{ij} = 1(\forall j)$
**PROBLEM DEFINITION**

The TVMPP framework is very general and can be applied to both offline and online scenarios.

**Offline** - multiple customers request VMs. DC operators gather input and solve VM placement.

**Online** - periodically collect data and solve TVMPP and decide whether a VM placement shuffle is needed.
**Complexity Analysis**

- TVMPP falls into the category of *Quadratic Assignment Problem (QAP)*.
- QAP is known to be NP-hard.
- QAP is one of the most difficult problems in the NP-hard class. (even finding an $\sigma$-approximation algorithm is NP-hard).
Since solving TVMPP optimaly is not an easy job, we’ll focus on two models:

1. Global traffic model - VM communication is at a constant rate.
2. Partitioned traffic model - VMs form isolated partitions, only VMs in each partition communicate with each other.
**COST MATRIX**

We need to formulate the Cost matrix, which is dependant on the network architecture

![Tree and VL2 diagrams with cost matrices](attachment:image.png)
**COST MATRIX**

**Fat-Tree**

Cost Matrix:

```
0 1 3 3 5 5 5 5 5 5 5 5 5 5 5 5
1 0 3 3 5 5 5 5 5 5 5 5 5 5 5 5
3 3 0 1 5 5 5 5 5 5 5 5 5 5 5 5
3 3 1 0 5 5 5 5 5 5 5 5 5 5 5 5
5 5 5 5 0 1 3 3 5 5 5 5 5 5 5 5
...```

**BCube**

Cost Matrix:

```
0 1 1 1 1 3 3 3 1 3 3 3 1 3 3 3
1 0 1 1 3 3 3 1 3 3 3 1 3 3 3 1
1 1 0 1 3 3 3 1 3 3 3 1 3 3 3 1
1 1 0 3 3 3 1 3 3 3 1 3 3 3 1
1 3 3 3 0 1 1 1 3 3 3 1 3 3 3 1
...```
GLOBAL TRAFFIC MODEL

- Each VM sends traffic to every other VM at equal and constant rate.
- Traffic matrix $D$ consists of constant row vectors.
- For any permutation matrix $X$, $DX^T = D$ holds.
- TVMPP is simplified:

$$\min_{X \in \Pi} S_{opt} = tr(DC^T X)$$

(Classical Linear Sum Assignment problem $\rightarrow O(n^3)$)

- Random placement:

$$S_{rand} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} (i, j) \text{ entry in } DC^T$$
Fig. 5. Optimal objective value vs objective value achieved by random placement (global traffic model, different traffic variance)

Evaluated on 1024 VMs
PARTITIONED TRAFFIC MODEL

- Each VM belongs to a group of VMs and it sends traffic only to other VMs in the same group, with pairwise traffic rate following a normal distribution.
- Computing $S_{opt}$ requires an exact solution, which becomes expensive for problem sizes larger than 15 VMs/slots.
- We’ll replace $S_{opt}$ by the classical Gilmore-Lawler Bound (GLB).
- The GLB is a lower bound for the optimal objective value of a QAP problem.
  Objective is close to GLB → little room for improvement.
  Objective is far from GLB → we don’t know anything, the potential for improvement is high.
Fig. 6.  GLB vs objective value achieved by random placement (partitioned traffic model, 16 partitions of 64 VMs each)
Fig. 7. GLB vs objective value achieved by random placement (partitioned traffic model, 10 partitions with $2^i$ VMs in each, $i = 1, \ldots, 10$)
Fig. 8. GLB vs objective value of random placement (with different partition size)
Algorithm Design

TVMPP is NP-hard, thus, no exact solution can be found for the size of current DCs.
ALGORITHM DESIGN

TVMPP is NP-hard, thus, no exact solution can be found for the size of current DCs.

Design principle 1

Proposition:
Suppose \(0 \leq a_1 \leq a_2 \ldots \leq a_n\) and \(0 \leq b_1 \leq b_2 \ldots \leq b_n\), the following inequalities hold for any permutation \(\pi\) on \([1, \ldots, n]\):

\[
\sum_{i=1}^{n} a_i b_{n-i+1} \leq \sum_{i=1}^{n} a_i b_{\pi(i)} \leq \sum_{i=1}^{n} a_i b_i
\]
ALGORITHM DESIGN

TVMPP is NP-hard, thus, no exact solution can be found for the size of current DCs.

Design principle 1

Proposition:
Suppose $0 \leq a_1 \leq a_2 \ldots \leq a_n$ and $0 \leq b_1 \leq b_2 \ldots \leq b_n$, the following inequalities hold for any permutation $\pi$ on $[1, \ldots, n]$:

$$\sum_{i=1}^{n} a_i b_{n-i+1} \leq \sum_{i=1}^{n} a_i b_{\pi(i)} \leq \sum_{i=1}^{n} a_i b_i$$

Solving TVMPP is equivalent to finding a mapping of VMs to slots such that VM pairs with heavy mutual traffic be assigned to slot pairs with low-cost connections.
Algorithm design

Design principle 2

Divide-and-conquer

- partition VMs into VM-clusters and partition slots into slot-clusters.
- Map each VM-cluster to a slot cluster.
- For each VM-cluster and its associated slot-cluster, solve another TVMPP problem but with a smaller problem size.

VM-clusters are obtained via classical min-cut graph algorithm which ensures that VM pairs with high mutual traffic rate are within the same VM-cluster.

Slot-clusters are obtained via standard clustering techniques which ensures slot pairs with low-cost connections belong to the same slot-cluster.
Algorithm design

SlotClustering
Partition $n$ slots into $k$ clusters.
Can be done manually by the Data Center operators, or by running an algorithm based on the cost matrix.
This becomes the Minimum $k$-clustering Problem (NP-hard), solved by an algorithm with approx. ratio 2.
**Algorithm Design**

**VMMinKcut**
Partition $n$ VMs into $k$ VM-clusters with minimum inter-cluster traffic.

Use the Gomory-Hu algorithm to find all min-cut between every VM pair.

- There are $n - 1$ distinct min-cuts.
- Sort min-cut in increasing order.
- Find a subset such that their removal from $G$ leads to a partition with the requested size.
**Algorithm 1 - Cluster-and-Cut**

Input: D,C,k (no. of clusters)

1. \( n \leftarrow \) size of \( D \) \{find out VM count\}
2. **SlotClustering(C,k)** \{partition slots into \( k \) clusters\}
3. Sort \( \{r_i\} \) in decreasing order of the cost of edges having one endpoint in \( \{r_i\}\)
4. **VMMinKcut\((D, |\{r_1\}|, ..., |\{r_k\}|)\)** \{partition \( n \) VMs into \( k \) clusters\}
5. Assign \( s_i \) to \( \{r_i\}, \forall i = 1, ..., n \) \{1-to-1 mapping between slot cluster and VM cluster\}
6. for \( i = 1 \) to \( k \) do
   7. if \( |s_i| > 1 \) then \{Multiple VMs in \( s_i \}\}
   8. Cluster-and-Cut\((D(s_i), C(r_i), |s_i|)\) \{recursively call Cluster-and-Cut\}
9. end if
10. end for
Algorithm 2 - VMMinKcut

Input: G(Graph weight matrix), \( \{b_1, ..., b_n\} \) (size of each cluster)

1. \( n \leftarrow \) size of \( G \)
2. Compute Gomory-Hu tree for \( G \) and obtain \( n - 1 \) cuts \( \{g_i\} \) \{These \( n - 1 \) cuts contains the minimum weight cuts for all server pairs\}
3. Sort \( \{g_i\} \) by increasing weight
4. for \( i = 1 \) to \( k \) do
5. Clear \( s_i \)
6. Find the minimum \( j \) such that removing \( \{g_1, ..., g_j\} \) will partition \( G \) into two components: \( c_1 \) with size \( |b_i| \) and \( c_2 \) with size \( n - |b_i| \)
7. \( s_i \leftarrow c_1 \)
8. \( G \leftarrow c_2 \)
9. \( n \leftarrow n - |b_i| \)
10. end for
11. Return \( \{s_i\} \)
EXPERIMENT SETTINGS

There are many heuristics for general QAP problems, we’ll select two to be compared to Cluster-and-Cut:

1. Local Optimal Pairwise Interchange (LOPI)
2. Simulated Annealing (SA)

We’ll consider a *hybrid traffic model* which combines real traces and the classical *Gravity model*:

\[ D_{ij} = \frac{D_{out}^i D_{in}^j}{\sum_k D_{in}^k} \]
<table>
<thead>
<tr>
<th>Topology</th>
<th>Algorithms</th>
<th>Gilmore-Lawler bound</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td><strong>Best value</strong></td>
</tr>
<tr>
<td>Tree</td>
<td>LOPI</td>
<td>4.63e+10</td>
<td>8.22e+10</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td></td>
<td>8.35e+10</td>
</tr>
<tr>
<td></td>
<td>Cluster-and-Cut</td>
<td></td>
<td>8.13e+10</td>
</tr>
<tr>
<td>VL2</td>
<td>LOPI</td>
<td>7.03e+10</td>
<td>1.09e+11</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td></td>
<td>1.12e+11</td>
</tr>
<tr>
<td></td>
<td>Cluster-and-Cut</td>
<td></td>
<td>1.05e+11</td>
</tr>
<tr>
<td>Fat-tree</td>
<td>LOPI</td>
<td>6.43e+10</td>
<td>1.07e+11</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td></td>
<td>1.12e+11</td>
</tr>
<tr>
<td></td>
<td>Cluster-and-Cut</td>
<td></td>
<td>0.97e+11</td>
</tr>
<tr>
<td>BCube</td>
<td>LOPI</td>
<td>5.55e+10</td>
<td>1.43e+11</td>
</tr>
<tr>
<td></td>
<td>SA</td>
<td></td>
<td>1.41e+11</td>
</tr>
<tr>
<td></td>
<td>Cluster-and-Cut</td>
<td></td>
<td>1.21e+11</td>
</tr>
</tbody>
</table>

**TABLE I**

**Algorithm performance with hybrid traffic model**
**EXPERIMENT RESULTS**

1. Objective function if Cluster-and-Cut is !10% lower.
2. CPU time halved.
CONCLUSIONS

- TVMPP is formulated and proven to be NP-hard.
- Careful VM placement can reduce traffic at DC switches.
- Traffic patterns and network topology affect the potential a network to scale by solving TVMPP.
LOOKING FORWARD...

- Combine VM migration with dynamic routing protocols.
- VM placement by joint network and server resource optimization.

Personal thoughts:
- Why didn’t they use the real traces in all evaluations?
- Why didn’t they show us the performance of TVMPP on network Arch.?
- Why didn’t they offer a complete model of applying TVMPP?
- Impact is low