# 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: Zhao Jian

# Introduction

- Scalability of modern data center has become a practical concern.
- The techniques suggested to solve the issue:
  - Network architecture with rich connectivity
  - Dynamic routing protocols
  - Traffic-aware virtual machine placement

    The bandwidth usage between VMs is rapidly growing

# Introduction

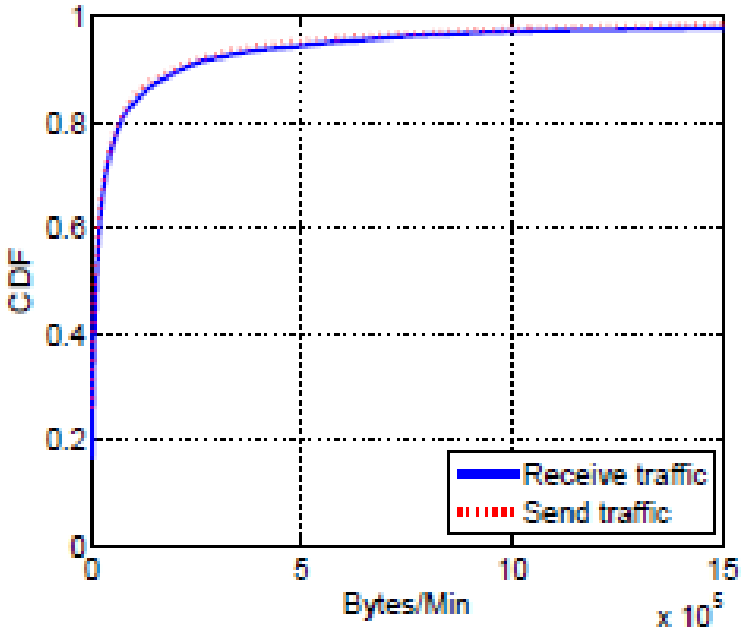
- This paper tackles the scalability from a different perspective, by optimizing the placement of VMs on host machines.
- Normally, VM placement is decided by Capacity Planning tools:
  - VMware Capacity Planner, IBM WebSphere CloudBurst, Novell PlateSpin Recon etc.
  - These tools focus on CPU, RAM and power consumption consolidation but ignore network resource consumption.

# Background

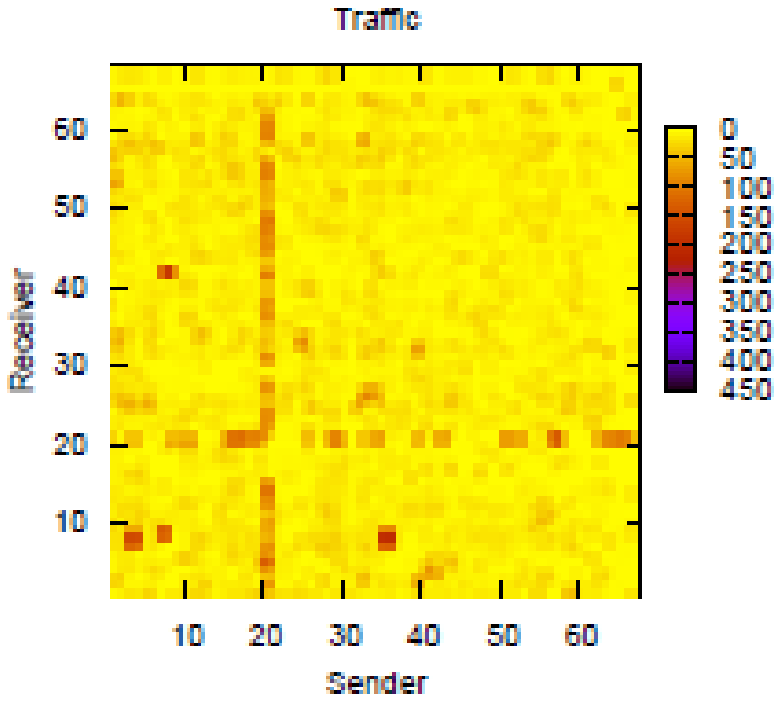
- Data Center Traffic Pattern:
  - Examine traces from two data-center-like systems to better understand data center traffic patterns.
    - Incoming and outgoing traffic rates for 17 thousand VMs in a data warehouse hosted by IBM Global Services
    - Incoming and outgoing TCP connections for 68 VMs in an “unnamed” server cluster with about hundreds of VMs.

# Background

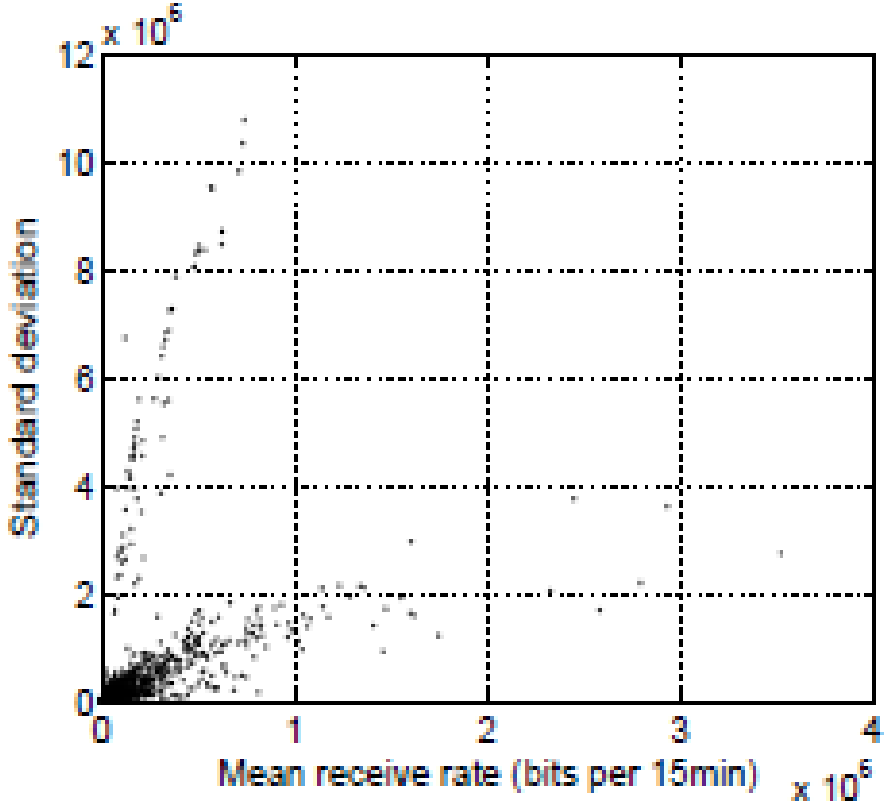
- Observed three trends regarding traffic patterns in data centers:
  1. Uneven distribution of traffic volumes from VMs:




(a) CDF of mean traffic rate



# 2. Stable per-VM traffic at large timescale




(b) Distribution of (mean, standard deviation)

# 3. Weak correlation between traffic rate and latency:

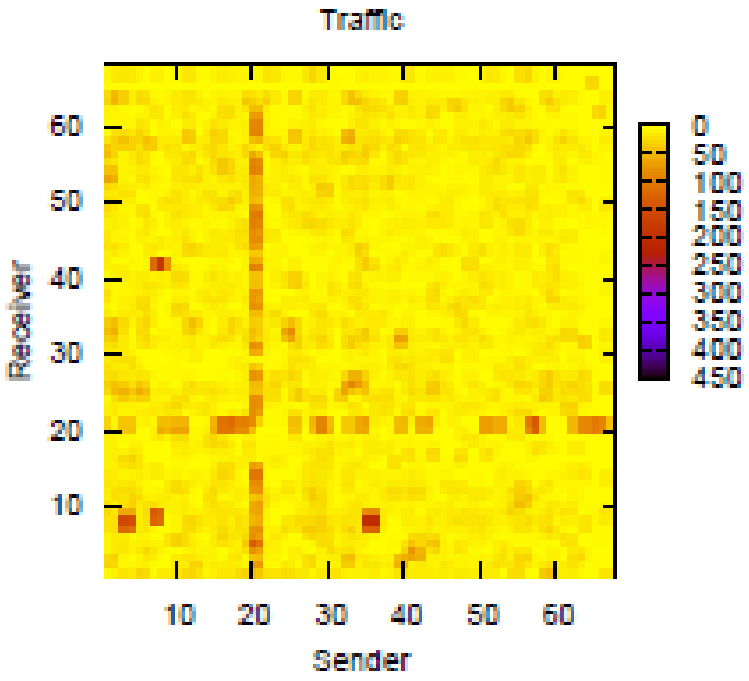



Fig. 2. Traffic matrix

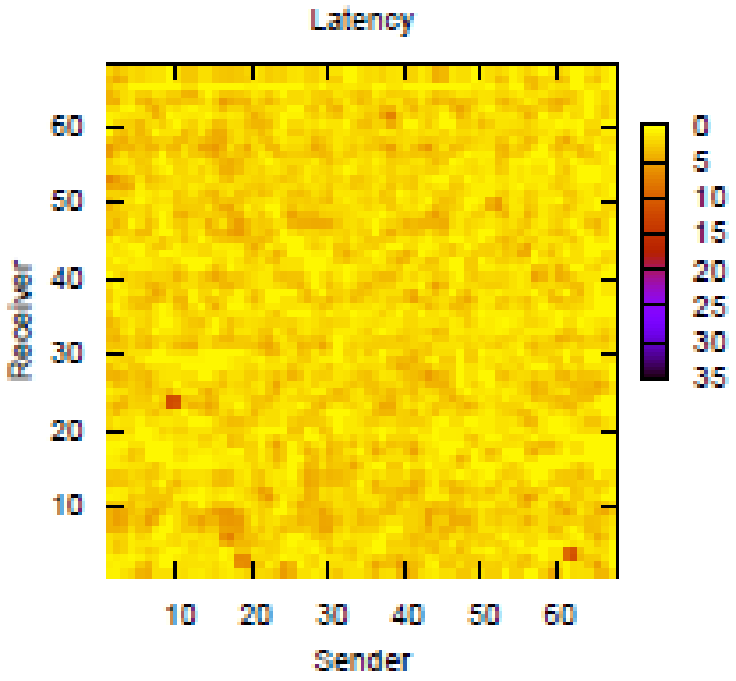



Fig. 3. Latency matrix

# VM placement problem definition

- Refer to the problem as a VM to slot assignment . A host can accommodate multiple VMs.
- Consider a static scenario where there are n VMs and n slots.
- Assume the routing is static and single-pathed.
- Ci,j = communication cost between slot i and j.
- Di,j = Traffic rate from VM i to VM j.
- *ei* = External traffic rate from VM i.
- $g_i$ = Communication cost between slot i and the gateway.
- For any assignment of VMs to slots, there is a permutation function $\pi: [1, \dots, n] \rightarrow [1, \dots, n]$

# Problem definition

- Traffic-aware VM Placement Problem (TVMPP) is to find a $\pi$ to minimize the following objective function:

$$\sum_{i,j=1,\ldots,n} D_{ij} C_{\pi(i)\pi(j)} + \sum_{i=1,\ldots,n} e_i g_{\pi(i)}$$

- It is equivalent to:

$$\min_{X \in \Pi} \; tr\left(D X^T C^T X\right) + e X^T g^T$$

  - $tr(A) = \sum_i A_{ii}$
  - $\Pi$ 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 -approximation algorithm is NP-hard).

# Algorithm design

- Cluster-and Cut: two design principles
  - Design principle 1:

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 $[1, \ldots, n]$.

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

2. 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 connection.

- 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.

- **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.

- **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.

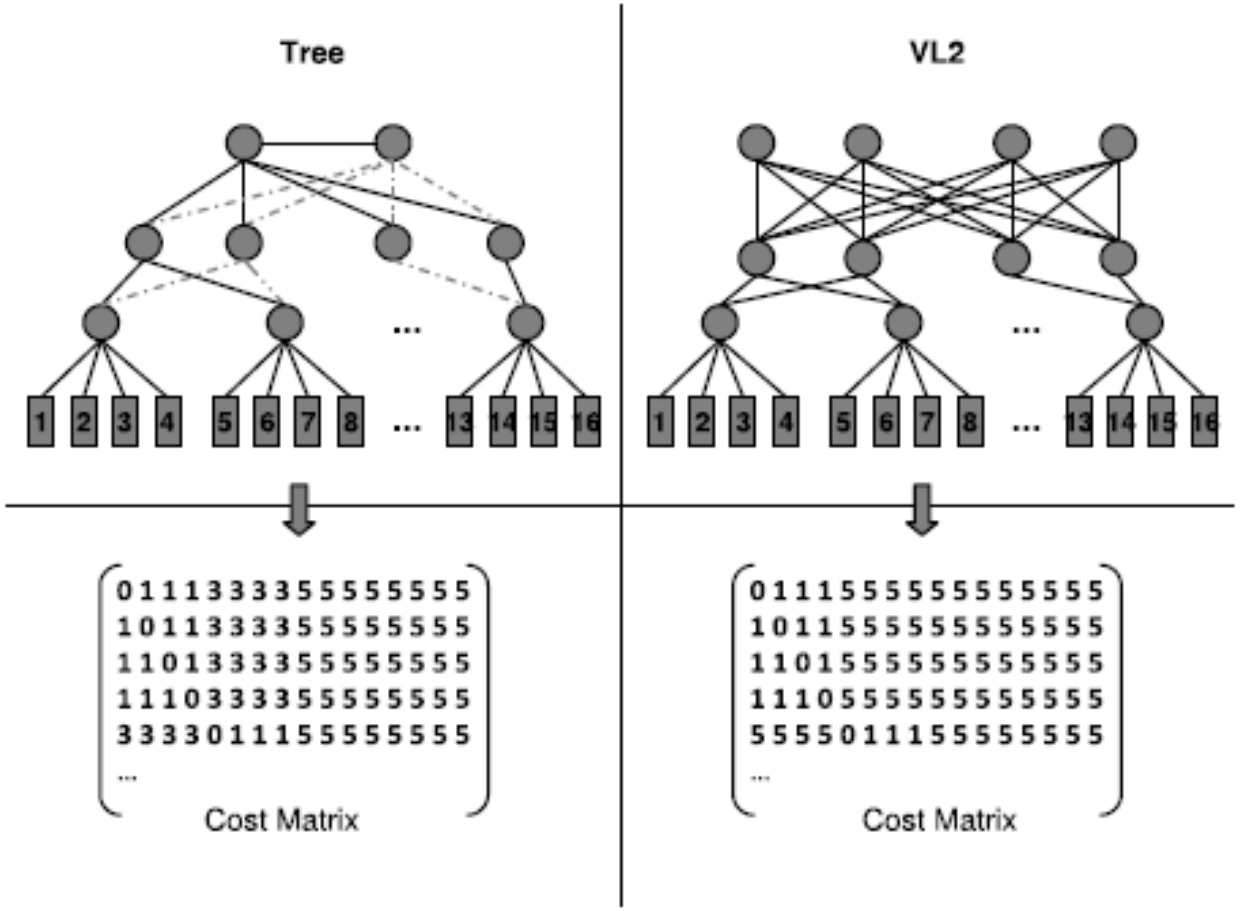
# Cluster and Cut

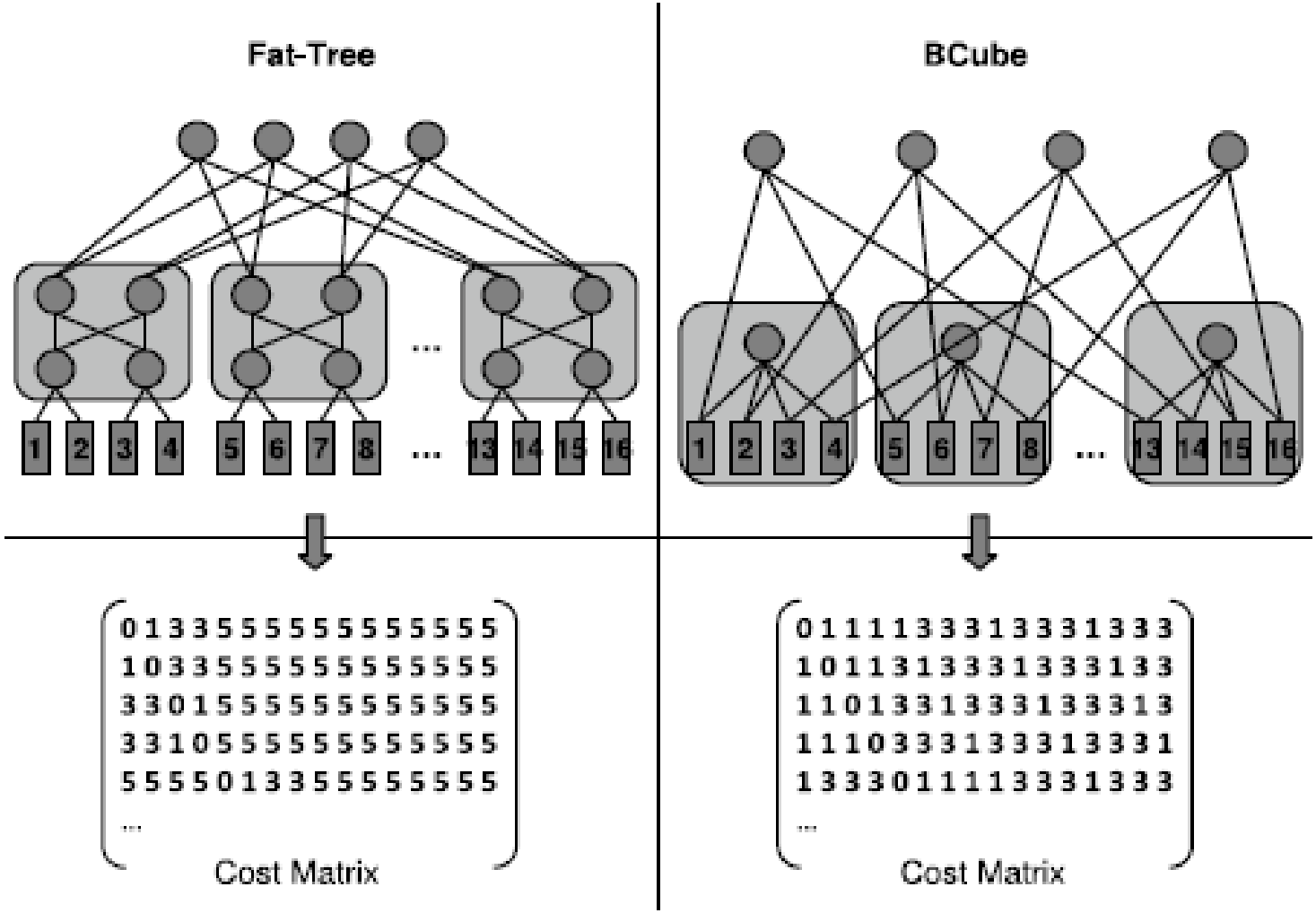
- Input D,C, k(no. of clusters)

1: $n \leftarrow$ size of $D$ {Find out VM count}
2: **SlotClustering(C,k)** {Partition slots into $k$ clusters: $\{r_i\}$}
3: Sort $\{r_i\}$, in decreasing order of the cost of edges having one endpoint in $r_i$. Each edge only counts once
4: **VMMinKcut(**$D, \{|r_1|, \ldots, |r_k|\}$**)** {Partition $n$ VMs into $k$ clusters $\{s_i\}$, $|s_i| = |r_i|$}
5: Assign $s_i$ to $r_i$, $\forall i = 1, \ldots, n$ {One-to-one 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($D(s_i)$, $C(r_i)$, $|s_i|$){$D(s_i)$: traffic matrix for $s_i$. $C(r_i)$: cost matrix for $r_i$. $|s_i$: recursively call Cluster-and-Cut}
9: **end if**
10: **end for**

# Impact of Network Architectures and Traffic Patterns on Optimal VM Placement

- Cost matrix




Fat-Tree
BCube
1 2 3 4 5 6 7 8 ... 13 14 15 16
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
...
Cost Matrix
0 1 1 1 1 3 3 3 1 3 3 3 1 3 3 3
1 0 1 1 3 1 3 3 3 1 3 3 3 1 3 3
1 1 0 1 3 3 1 3 3 3 1 3 3 3 1 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 3
...
Cost Matrix

# 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:

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

- Random placement:

$$S_{rand} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{n} (i,j) \text{ entry in } DC^T$$

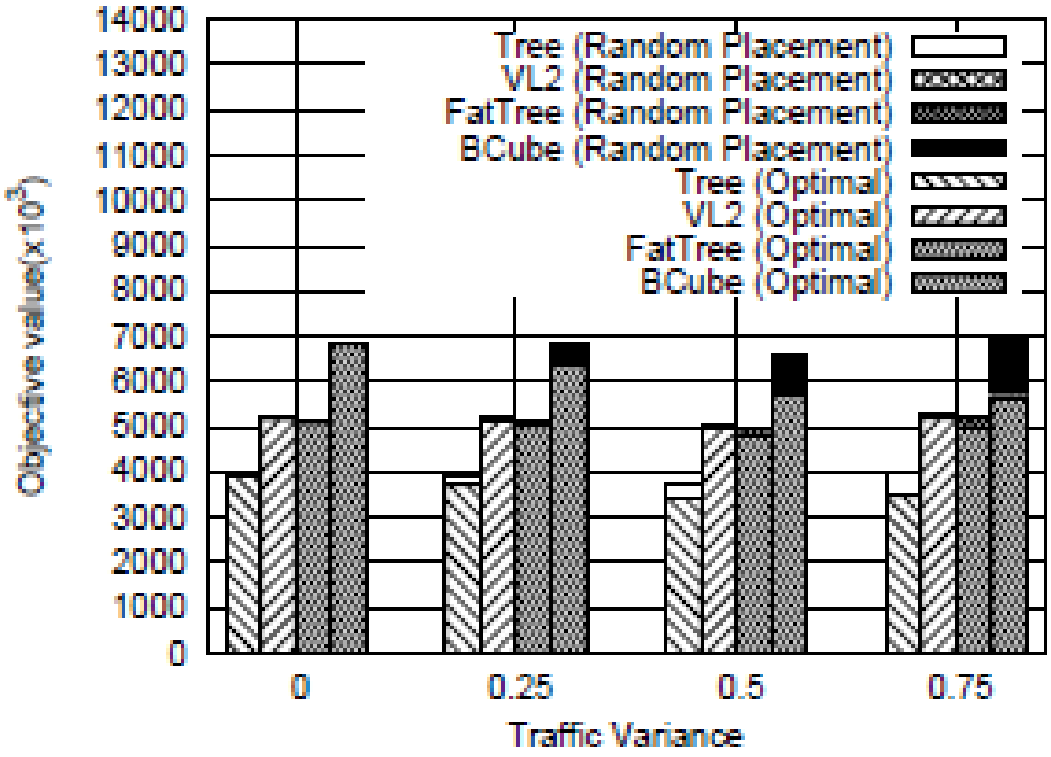



Fig. 5. Optimal objective value *vs* objective value achieved by random placement (global traffic model, different traffic variance)

# 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 do not know anything, the potential for improvement is high.

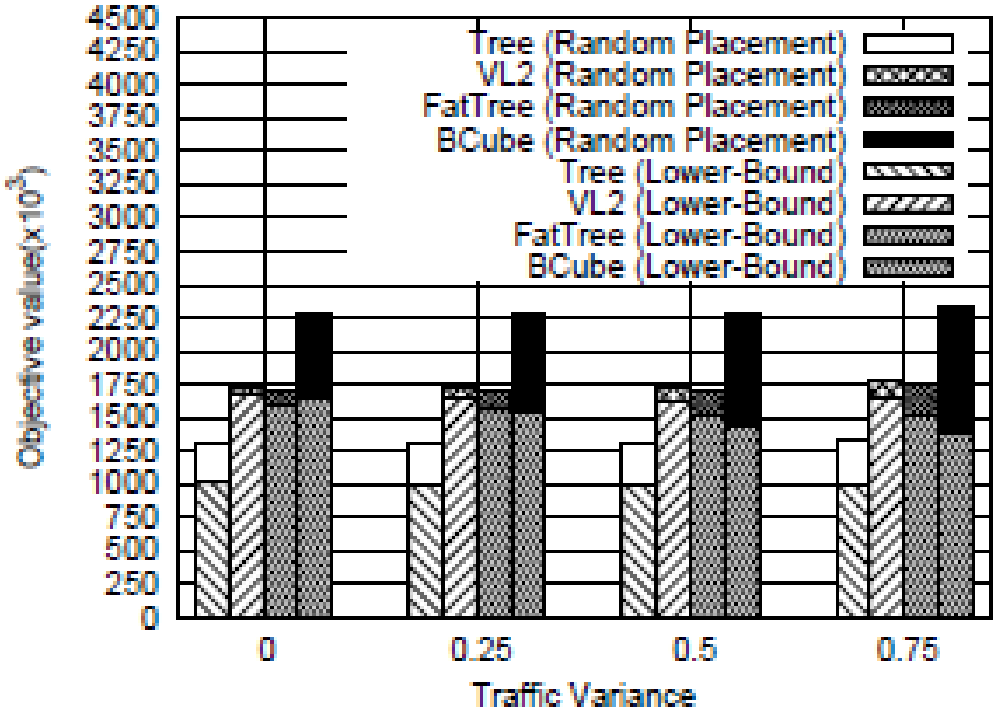



Fig. 6. GLB *vs* objective value achieved by random placement (partitioned traffic model, 16 partitions of 64 VMs each)

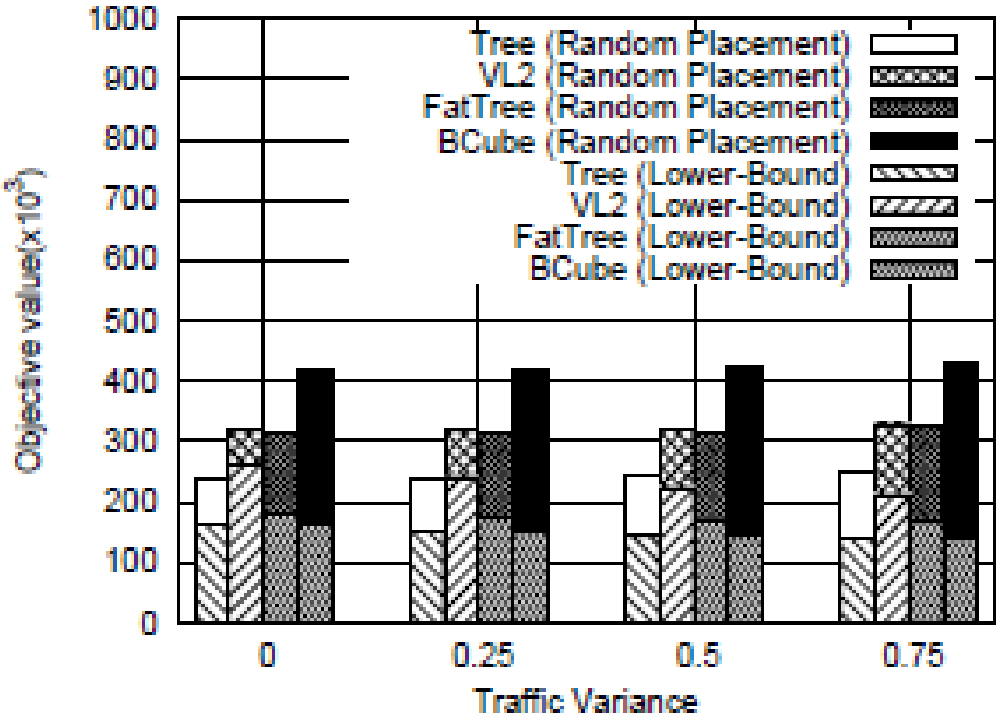



Fig. 7. GLB *vs* objective value achieved by random placemente (partitioned traffic model, 10 partitions with $2^i$ VMs in each, $i = 1, .., 10$)

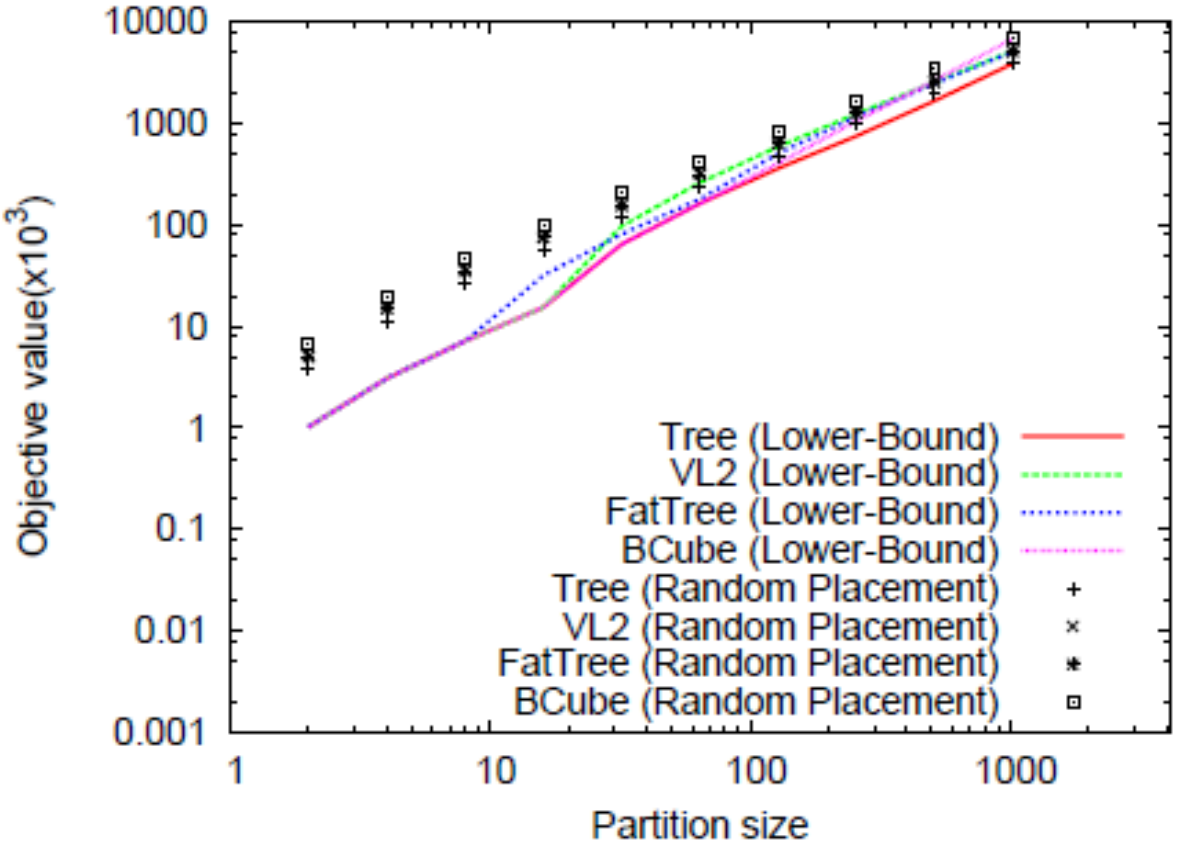



Fig. 8. GLB *vs* objective value of random placement (with different partition size)