A Scalable Algorithm for Placement of Virtual Clusters in Large Data Centers

Asser N. Tantawi
IBM T.J. Watson Research Center
Yorktown Heights, NY

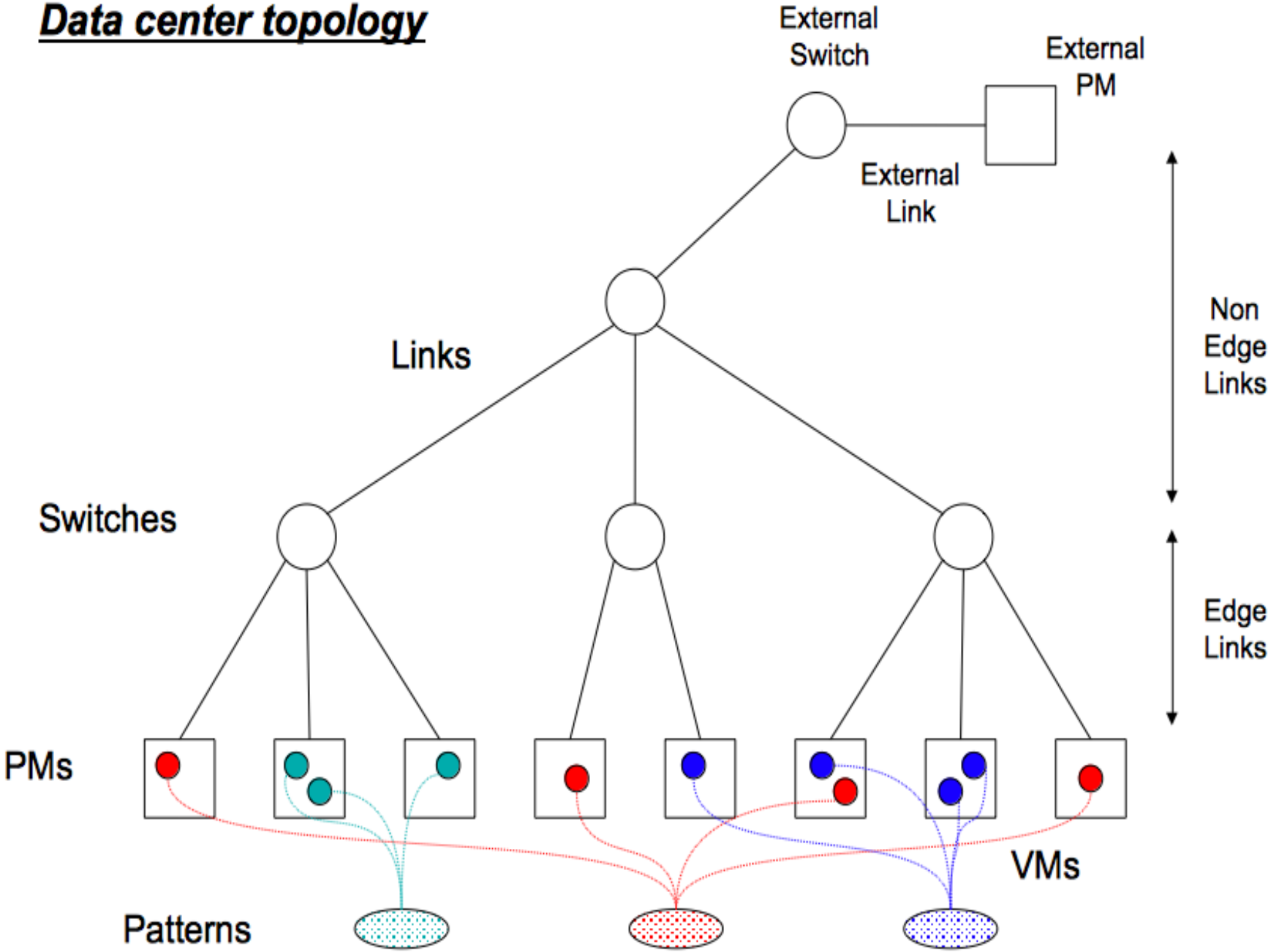
tantawi@us.ibm.com

August 7, 2012

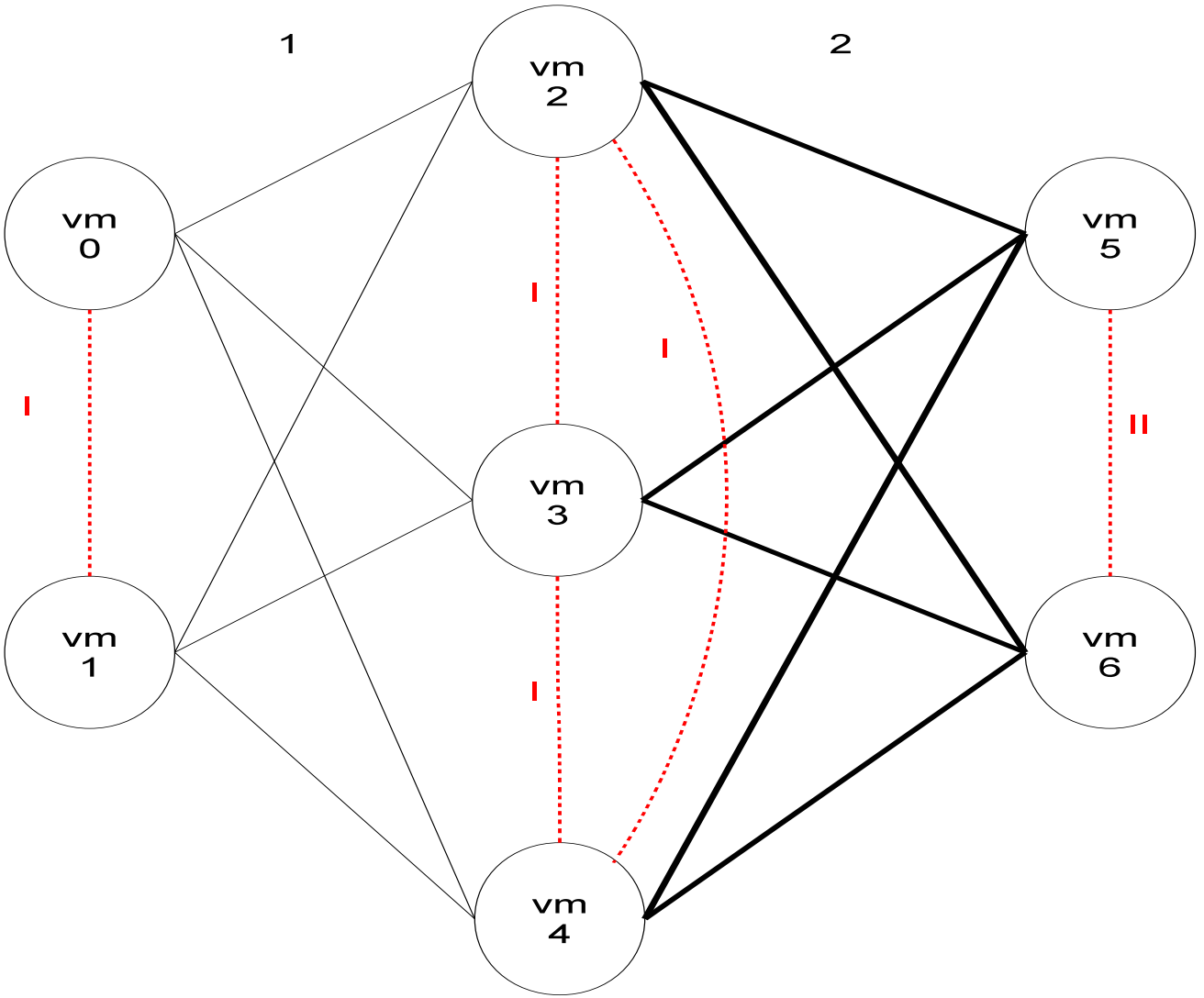
Summary

- We consider the problem of placing virtual clusters, each consisting of a set of heterogeneous virtual machines (VM) with some interrelationships due to communication needs and other dependability-induced constraints, onto physical machines (PM) in a large data center.
- We introduce a statistical approach based on importance sampling (also known as cross-entropy).
- We considerably enhance the basic method by biasing the sampling process to incorporate constraints of virtual clusters to yield an efficient algorithm that is linear in the size of the data center.
- We investigate the quality of the results of using our algorithm on a simulated system.

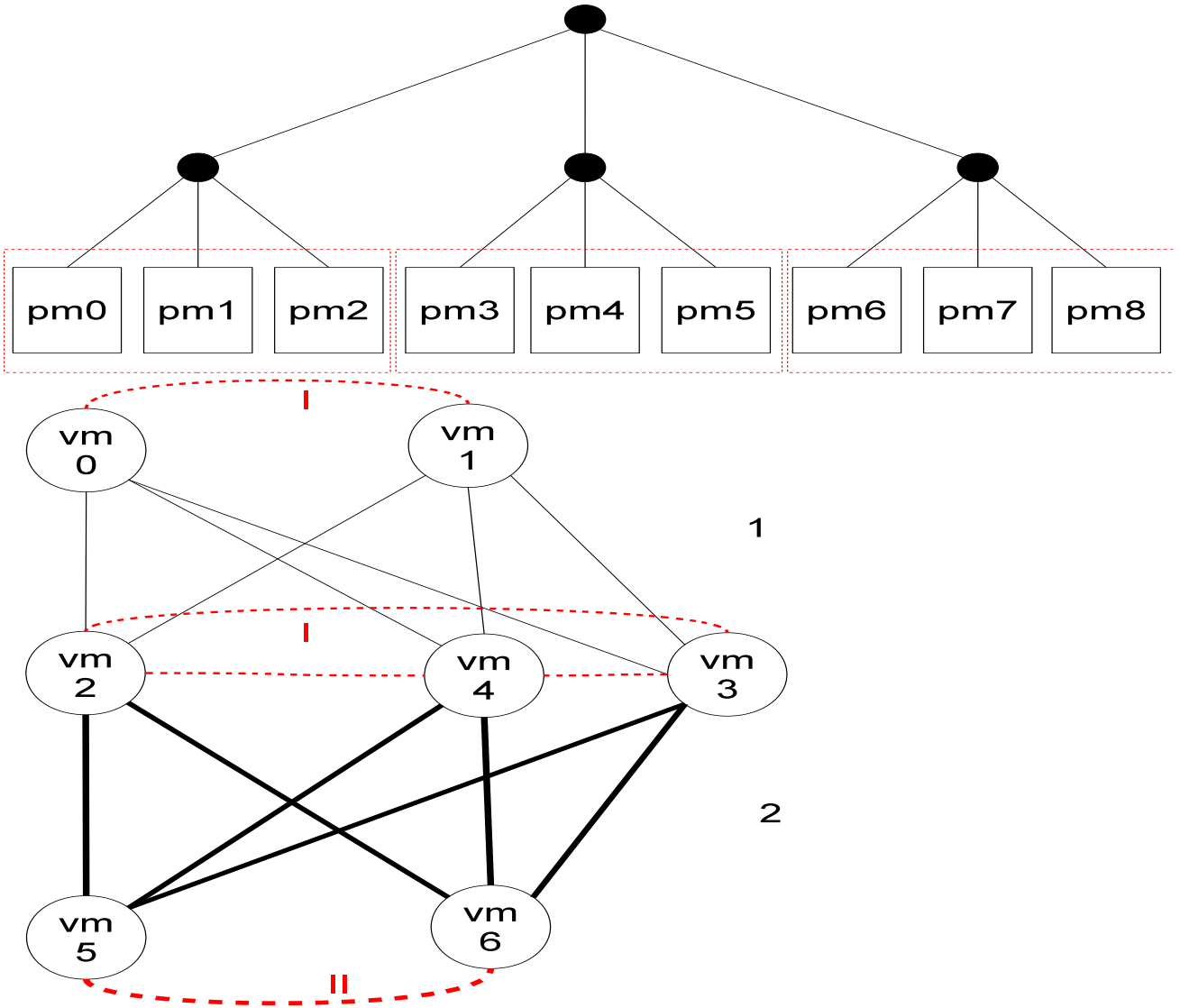
Data center topology



An example multi-tier pattern



and its placement ...



System Description

Physical machines

- Let $\mathcal{PM}$ denote the set of n_{pm} physical machines in the cloud, $n_{pm} = |\mathcal{PM}|$.
- We will refer to an element in the set as $pm_i, i = 1, \cdots, n_{pm}$.
- Each PM provides a set of resources, $\mathcal{R}$, consisting of resources $r_k, k = 1, \cdots, n_r$.
- The total capacity of resource r_k on pm_i is denoted by $c_{i,k}$.

System Description

Physical network

- The interconnection network forms a graph where the vertices are PMs and SWs, and the edges are LKs.
- Each link provides bandwidth for communication.
- We assume that the switches are fast and that communication delay is solely due to link congestion.
- A path $h_{i,j}$ between pm_i and pm_j consists of an ordered set of links $\{lk_{\pi_{i,j}(1)}, lk_{\pi_{i,j}(2)}, \cdots\}$, with path length denoted by $\eta_{i,j} = |h_{i,j}|$.
- We define $w_i(l), i = 1, 2, \cdots, n_{pm}$, and $l = 0, \cdots, L$ as the set of PMs such that for $pm_j \in w_i(l)$ we have $\eta_{i,j} = l$.

System Description

System availability

- We imagine that a data center is partitioned into a hierarchy of availability zones, where PMs in the same zone have similar availability characteristics. (A tree model)
- Let $v_{i,j}$ be the probability that at least one of pm_i or pm_j is available.
- We associate an availability level, V_l, $l = 0, \cdots, L$, for a node at level l in the tree, where $l = 0$ represents the leaves and $l = L$ represents the root of the tree with height L. ($V_0 < V_1 < \cdots < V_L$)
- Two PMs pm_i and pm_j with the lowest common ancestor at level l have $v_{i,j} = V_l$. (Clearly, $v_{i,i} = V_0$).
- We define $g_i(l), i = 1, 2, \cdots, n_{pm}$, and $l = 0, \cdots, L$ as the set of PMs such that for $pm_j \in g_i(l)$ we have $v_{i,j} = V_l$.

System Description

Virtual machines

- Each VM is characterized by a set of resource demands, one per resource type in the set $\mathcal{R}$.
- We refer to the PM which hosts vm_i as $pm(vm_i)$.

System Description

Virtual clusters (Patterns)

- A virtual cluster is a collection of VMs that make up a deployable unit which we refer to as *pattern*, $\mathfrak{p}$.
- The VMs in $\mathfrak{p}$ form the set $\mathcal{VM}(\mathfrak{p}) = \{vm_1(\mathfrak{p}), vm_2(\mathfrak{p}), \cdots, vm_{n_{vm}(\mathfrak{p})}(\mathfrak{p})\}$.
- The communication bandwidth demand of $\mathfrak{p}$ may be represented by a matrix $[\lambda_{i,j}]$, where $1 \leq i, j \leq n_{vm}(\mathfrak{p})$.
- Let $\mathcal{S}(\mathfrak{p}) \subset \mathcal{VM}(\mathfrak{p}) \times \mathcal{VM}(\mathfrak{p})$ be a set of distinct pairs of VMs in $\mathfrak{p}$.
- A pair $(vm_i, vm_j) \in \mathcal{S}(\mathfrak{p})$ has an availability constraint specified as $v_{pm(vm_i),pm(vm_j)} = \alpha$.
- This availability requirement is satisfied if $\alpha \leq V_l$ and $pm(vm_j) \in g_k(l)$, where $pm(vm_i) = pm_k$ for some $l, 0 \leq l \leq L$.

System Description

Pattern Placement

- We denote by $\pi(\mathfrak{p})$ a particular placement of pattern $\mathfrak{p}$.
- $\pi(\mathfrak{p})$ maps each VM in$\mathcal{VM}(\mathfrak{p})$ to a PM in such a way that
  1. the resource requirement of the VM is satisfied by this PM,
  2. the communication demand between this VM and other VMs in $\mathfrak{p}$ is satisfied by the links of the communication network, and
  3. the pairwise availability requirements are satisfied.
- We write such a placement as
  $\pi(\mathfrak{p}) = \{(vm_i, pm_m) \mid pm(vm_i) = pm_m, \forall vm_i \in \mathcal{VM}(\mathfrak{p})\}$.

Performance Measures

System performance

- PM utilization, $\rho_{i,k}$, of resource r_k on pm_i as $\rho_{i,k} = u_{i,k}/c_{i,k}$.
- Link utilization, ν_i, as $\nu_i = a_i/b_i$.
- The network delay between pm_i and pm_j is the sum of the link delays along the path $h_{i,j}$.
- Denoting the delay factor between pm_i and pm_j by $T_{i,j}$, we write

$$T_{i,j} = \sum_{k=1}^{\eta_{i,j}} \frac{1}{1 - \nu_{h_{i,j}(k)}}.$$

- The delay index $\delta_{i,j}$ between pm_i and pm_j along path $h_{i,j}$ is given by

$$\delta_{i,j} = 1 - \frac{\eta_{i,j}}{T_{i,j}},$$

Performance Measures

Pattern performance

- We define a weighted path length (distance) for pattern $\mathfrak{p}$, denoted by $\eta(\mathfrak{p})$, as

$$\eta(\mathfrak{p}) = \frac{\sum_{vm_i,vm_j} \lambda_{i,j} * \eta_{pm(vm_i),pm(vm_j)}}{\sum_{vm_i,vm_j} \lambda_{i,j}},$$

  where vm_i and vm_j go over the set $\mathcal{VM}(\mathfrak{p})$.

- Similarly, we define the weighted delay index for pattern $\mathfrak{p}$, denoted by $\delta(\mathfrak{p})$, as

$$\delta(\mathfrak{p}) = \frac{\sum_{vm_i,vm_j} \lambda_{i,j} * \delta_{pm(vm_i),pm(vm_j)}}{\sum_{vm_i,vm_j} \lambda_{i,j}}.$$

Optimization Problem

Optimization problem

- Given a cloud in state C, we are concerned with the placement $\pi(\mathfrak{p})$ of pattern $\mathfrak{p}$ so as to minimize the objective function $F(\pi(\mathfrak{p})|C)$.

$$\textit{Given } C, \textit{find } \pi(\mathfrak{p}) \mid \textit{min}\{F(\pi(\mathfrak{p})|C)\}.$$

- An optimal placement algorithm attempts to find PMs in the cloud
  - that have enough capacity to host the VMs in the pattern,
  - while making sure that there is enough bandwidth in the network to accommodate inter-VM bandwidth requirements,
  - as well as satisfying any pairwise VM availability constraints.

Placement algorithm

Importance sampling in a nutshell

- Find an optimal solution to a combinatorial (maximization) problem.
- Generate many samples of solutions using a parametrized probability distribution.
- Order the samples in their attained values of the objective function.
- The top small fraction of samples (*important samples*), are used to adjust the values of the parameters of the generating probability distribution so as to skew the generation process to yield samples with large objective values.
- The method iterates a few times until a good solution is obtained.

Placement algorithm

Pattern Placement

- We define the parameters for the sample generation process as a matrix $\mathbf{P} = [p_{i,j}]$ of probabilities, where $i = 1, 2, \cdots, n_{vm}$, is the i^{th} VM in the pattern, and $j = 1, 2, \cdots, n_{pm}$, is pm_j in the cloud.
- $p_{i,j}$ represents the probability of assigning vm_i in the pattern to pm_j.
- Starting from some initial $\mathbf{P}$, the algorithm proceeds to modify $\mathbf{P}$ until a solution is reached, represented by a dominant (close to 1) entry in each row of $\mathbf{P}$ and all other entries are negligibly small (close to 0).
- In every iteration of the importance sampling algorithm, the entries in $\mathbf{P}$ that correspond to a large objective value are reinforced and amplified, at the cost of other solutions that are away from optimality.

Sample biasing

We sequentially consider the VMs in the pattern without backtracking, i.e. once $vm_i, i = 1, 2, \cdots, n_{vm}$ is placed, the choice for placement is only left for $vm_{i+1}, \cdots, vm_{n_{vm}}$.

Initial biasing

- The initial setting of $\mathbf{P}$ should reflect the state C.
- A simple choice that is only based on PM resources is $p_{i,j} \propto 1/\rho_{j,k}$, where k is the bottleneck resource.

Sample biasing

Availability constraint biasing

- Once vm_i is placed on say $pm_i = pm(vm_i)$, we examine any availability constraint with $vm_m, m = i + 1, \cdots, n_{vm}$ in a look-ahead fashion.
- Let's say that vm_i and vm_m have an availability constraint of level l. Then, we need to bias $p_{m,j}$ positively towards $pm_j \in g_i(l)$ and negatively to all other PMs.
- If the constraint is soft, the negative biasing becomes more negative for $pm_j \in g_i(l-1) \cup g_i(l+1)$, $pm_j \in g_i(l-2) \cup g_i(l+2)$, and so on.
- The way biasing is done is through multiplying $p_{m,j}$ by a factor $f_{m,j}$ and normalize the $p_{m,\cdot}$ after all biasing factors are applied.

Sample biasing

Communication biasing

- We bias the probabilities depending on the number of hops between a given PM and the other PMs in the cloud.
- Once vm_i is placed on say $pm_i = pm(vm_i)$, consider the communication demand $\lambda_{i,m}$ between vm_i and vm_m, $m > i$.
- In order to keep vm_m placed close to pm_i we positively bias $p_{m,j}$ towards $pm_j \in w_i(0)$, i.e. pm_i, then less positively towards $pm_j \in w_i(1)$, and so on until we reach a most negative bias towards $pm_j \in w_i(L)$, the farthest away PMs from pm_i.
- The way biasing is done is through multiplying $p_{m,j}$ by a factor $f_{m,j}$ and normalize $p_{m,..}$

Results

Optimality verification

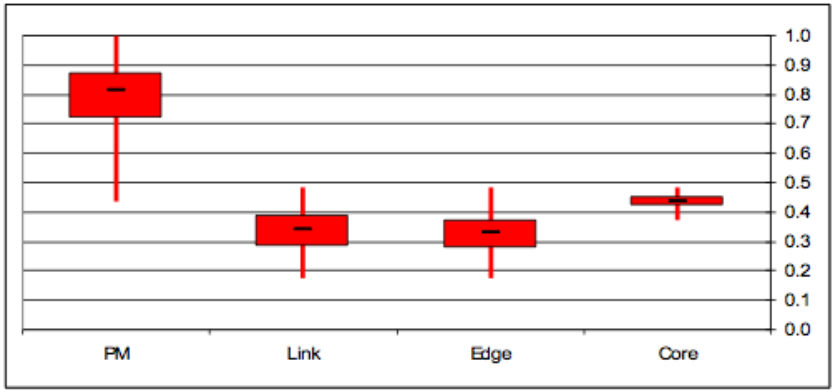
	Our Algorithm	Optimal (ILOG)
PM utilization	0.77	0.77
Edge link utilization	0.51	0.52
Core link utilization	0.20	0.20
Pattern path length	1.21	1.22
Pattern delay index	0.26	0.28
Pattern rejection prob.	0.02	0.02
Placement time *(msec)*	**3**	**10,329**

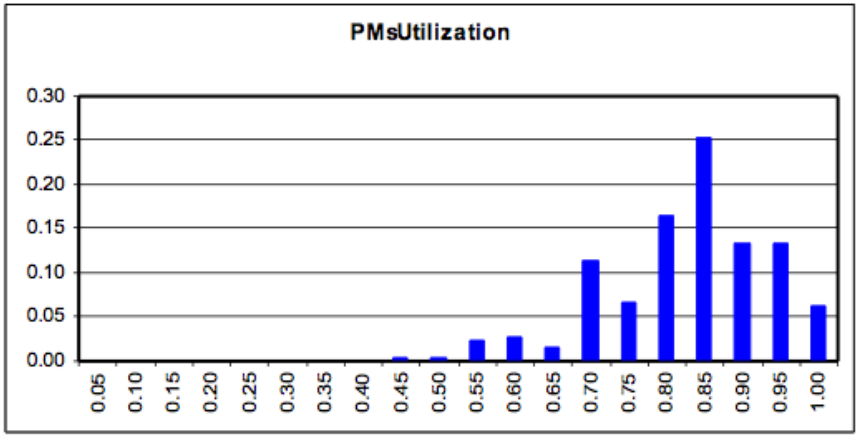
Results

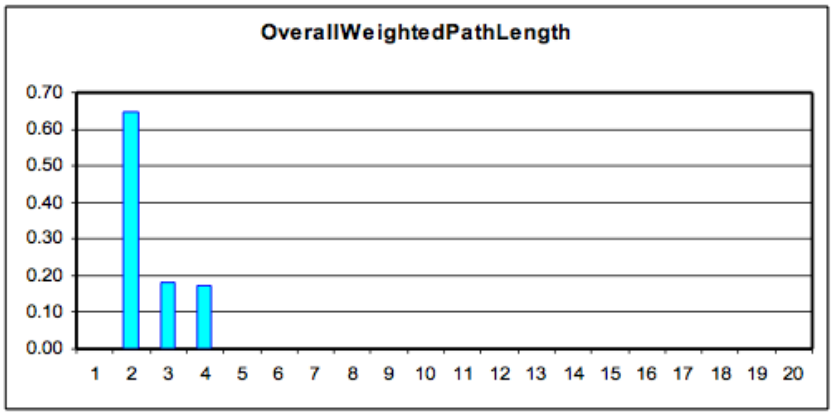
Unbiased importance sampling

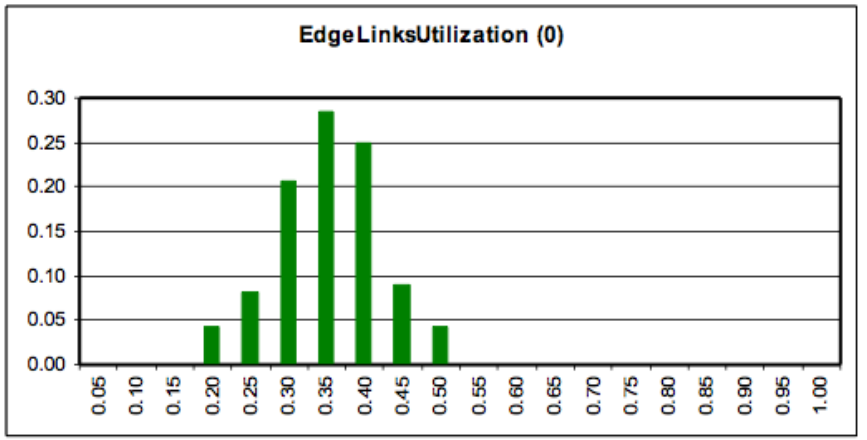
	Our Algorithm (with biasing)	Plain Importance Sampling (without biasing)
PM utilization	0.80	0.22
Edge link utilization	0.35	0.06
Core link utilization	0.44	0.17
Pattern path length	1.99	2.99
Pattern delay index	0.28	0.13
Pattern rejection prob	**0.0005**	**0.6382**
Placement time *(msec)*	115	354

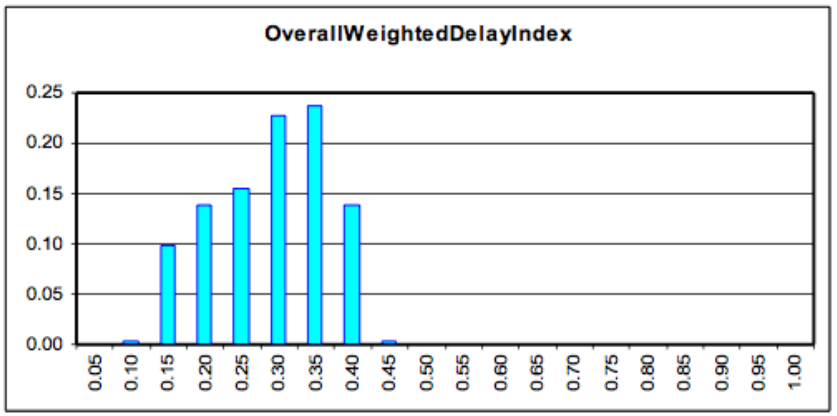
Base configuration results

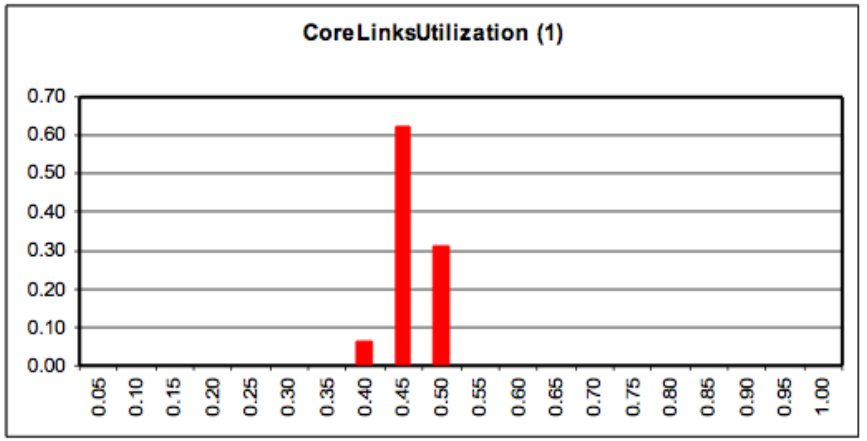




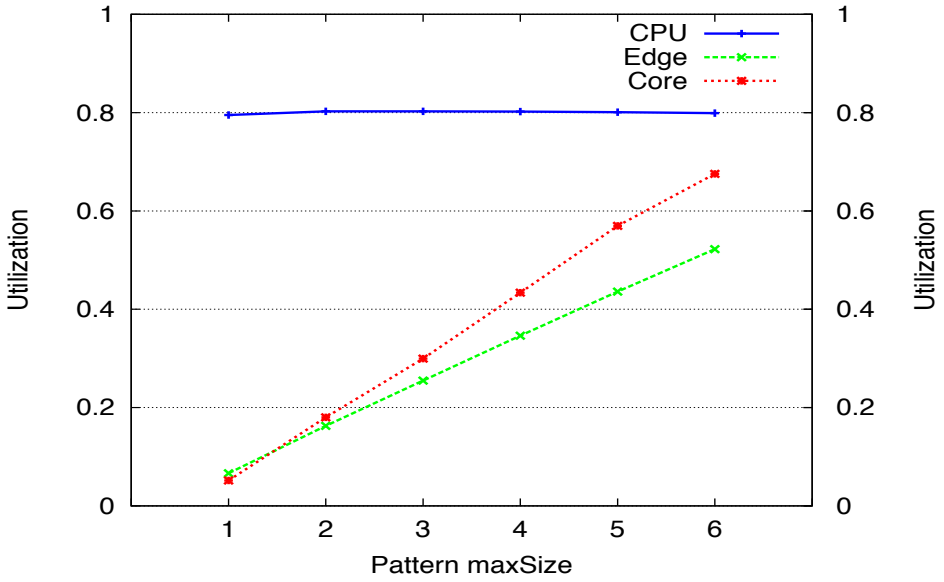




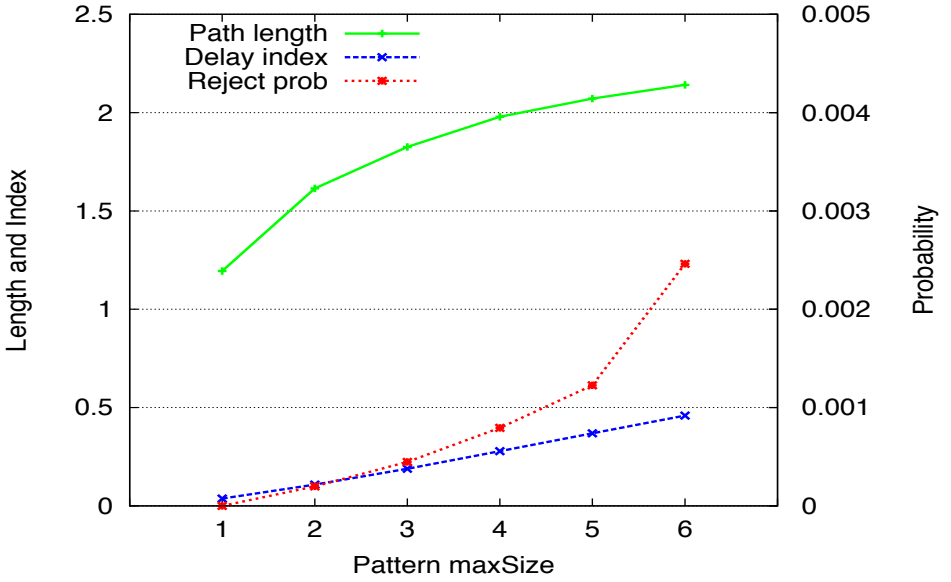




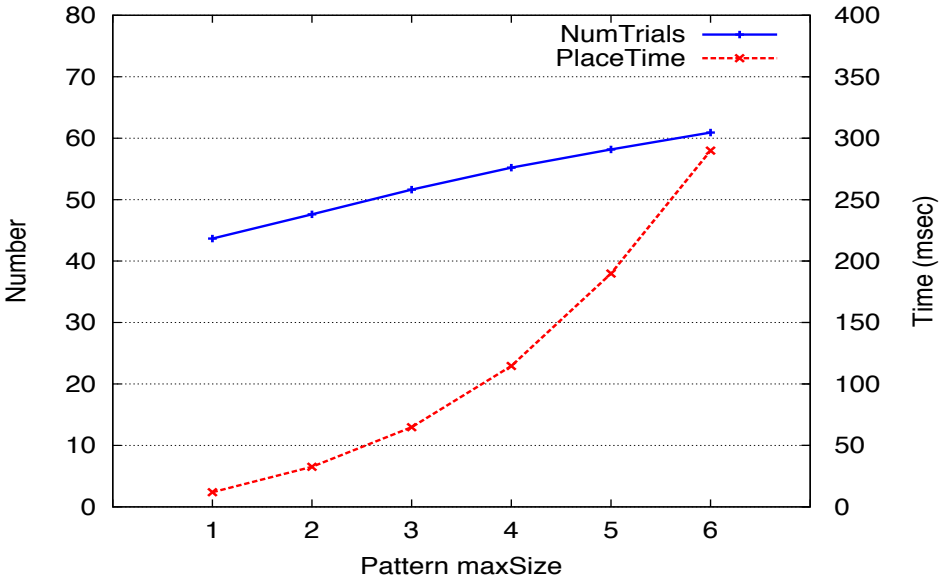
Effect of pattern size



(a) PM and network utilization.

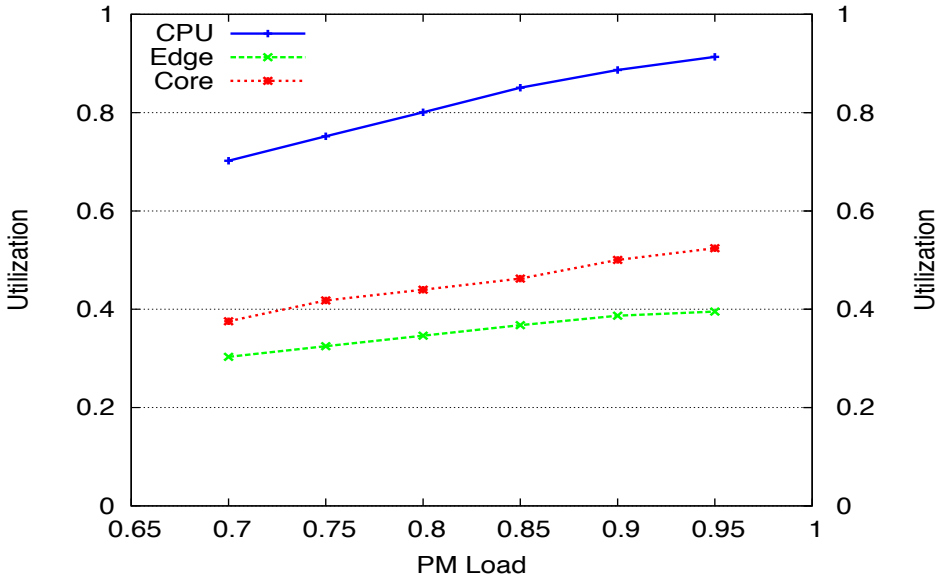


(b) Pattern performance.

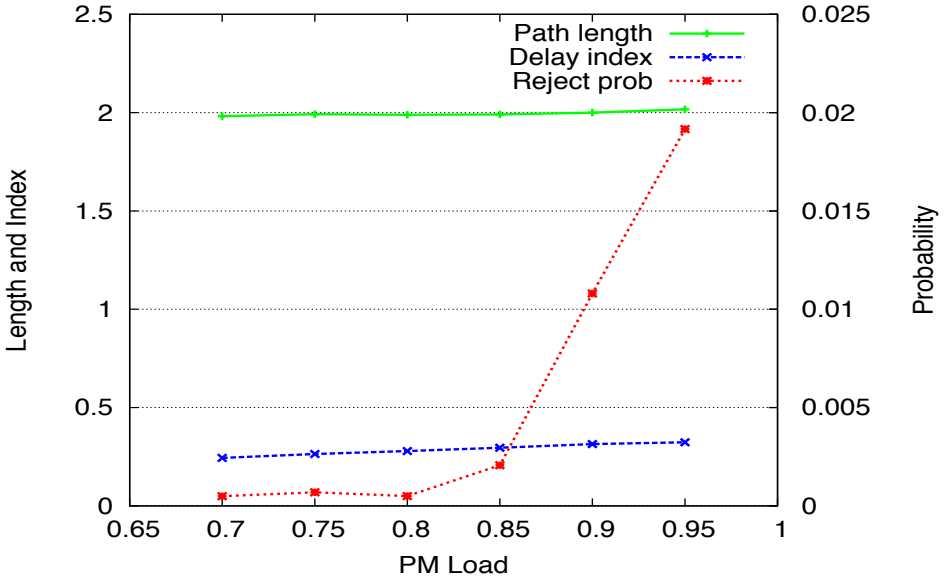


(c) Placement performance.

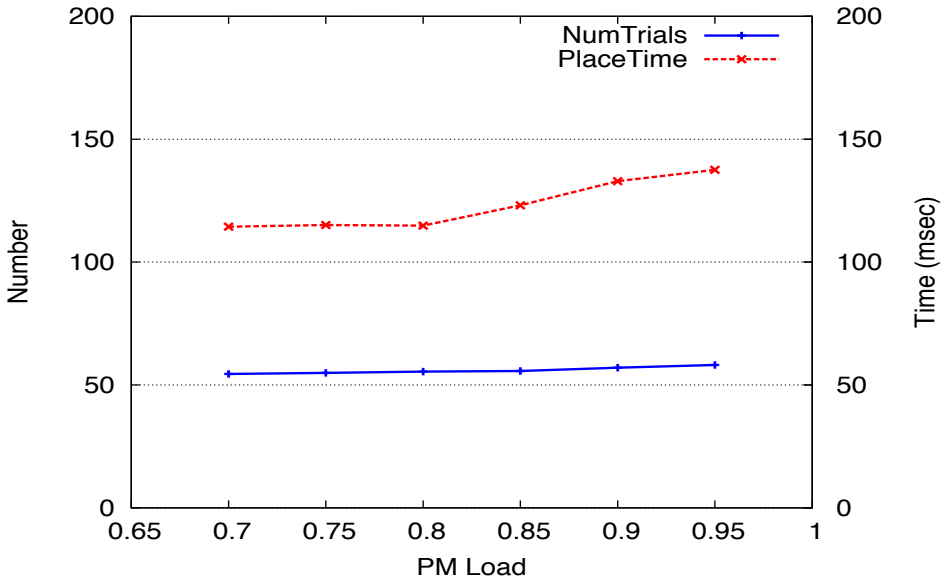
Effect of load



(a) PM and network utilization.

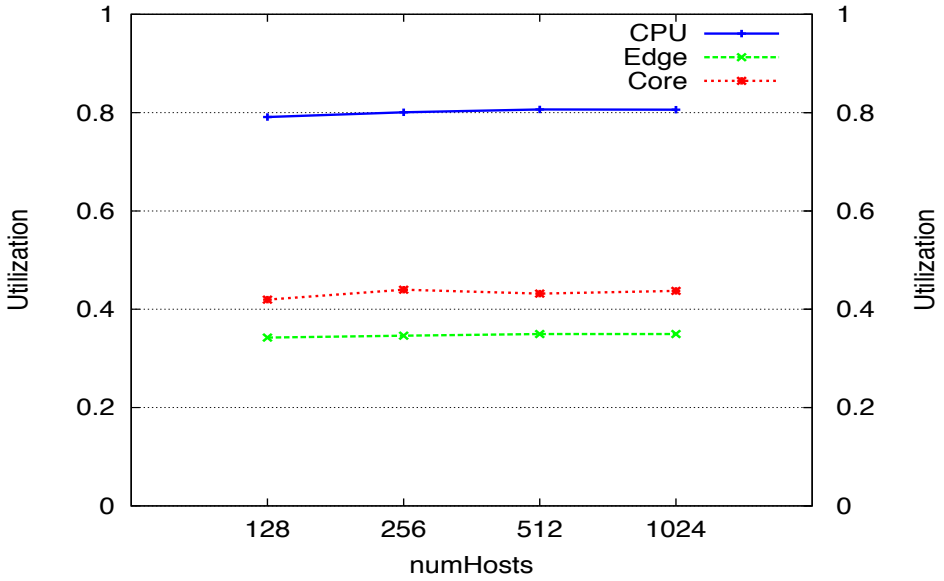


(b) Pattern performance.

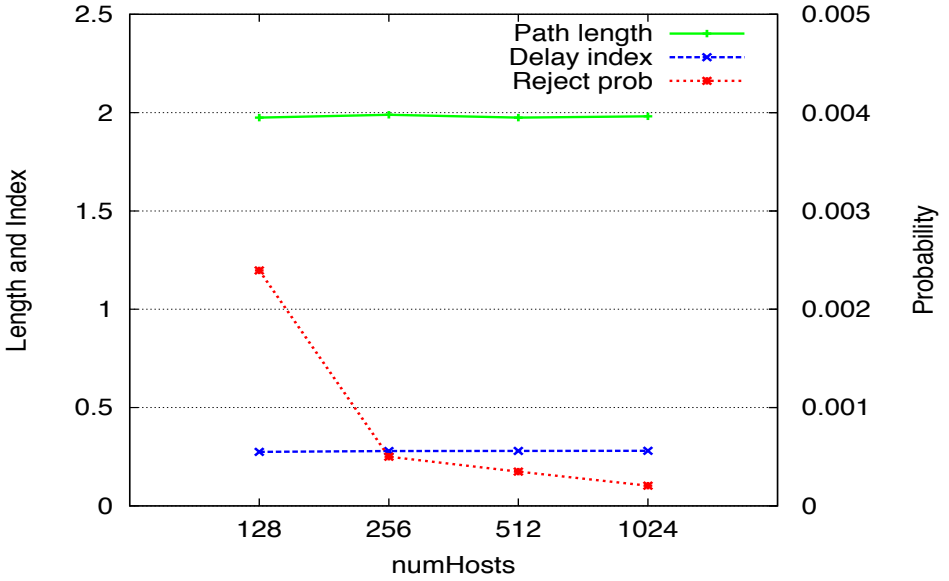


(c) Placement performance.

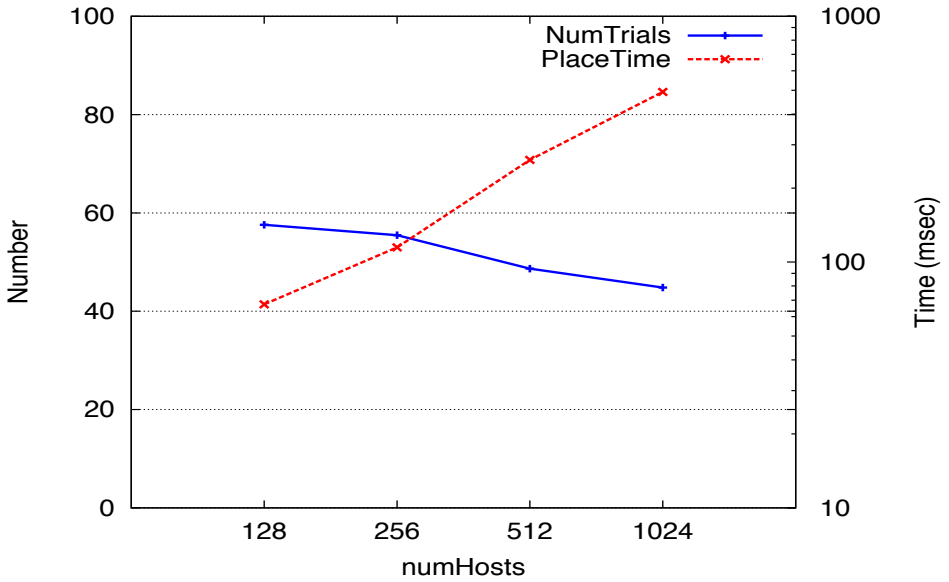
Effect of number of hosts



(a) PM and network utilization.



(b) Pattern performance.



(c) Placement performance.

Thank You!