Joint Virtual Machine Placement and Migration in Dynamic Policy-Driven Data Centers

•••

Hugo Flores J Lucas
California State University Dominguez Hills
Department of Computer Science

Presentation Overview

1. Introduction
2. Related Works
3. System Model
4. Virtual Machine Migration
5. Virtual Machine Placement
6. Performance Evaluation
7. Conclusion

Introduction

What is a Dynamic Policy Driven Data Center (PDDC)?

- Data Center
  - Physical Machines (PMs)
  - Switches
  - Virtual Machines (VMs)
- Policy Driven
  - Middleboxes (MBs)
  - Policy Chains (Ordered or Unordered)
- Dynamic
  - Communication Frequencies

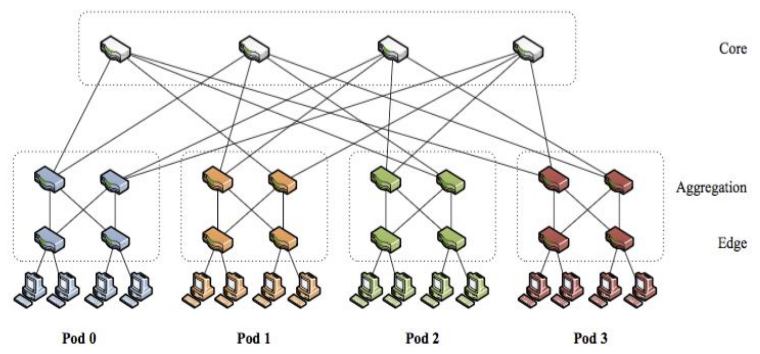
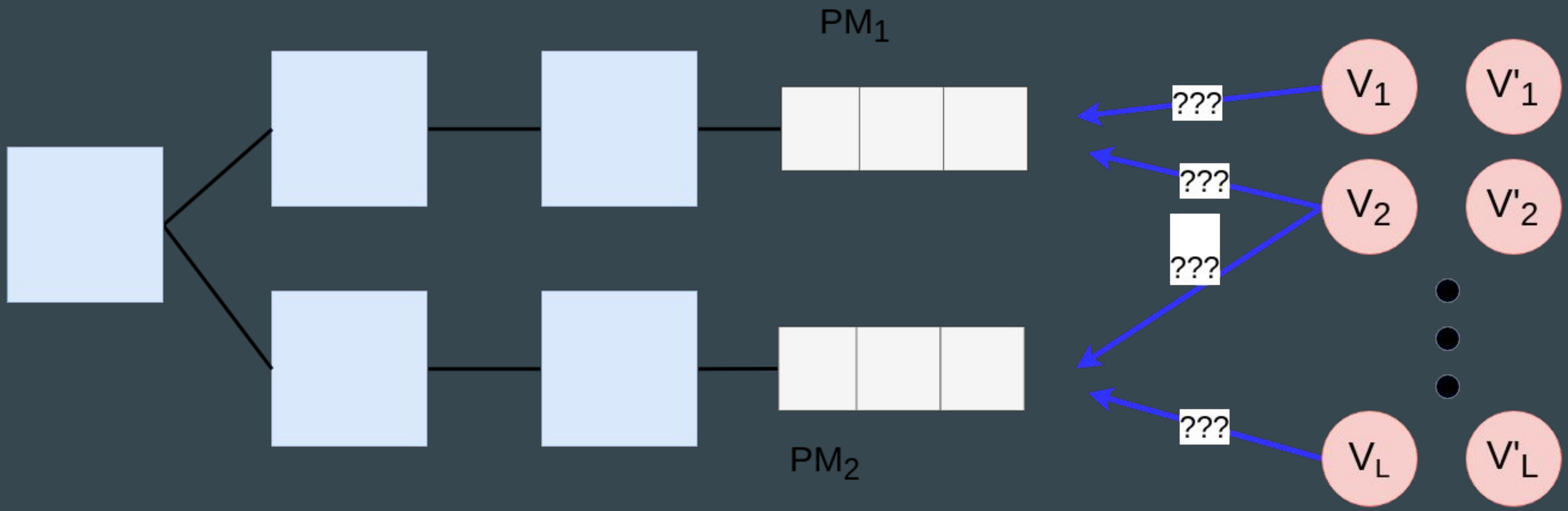
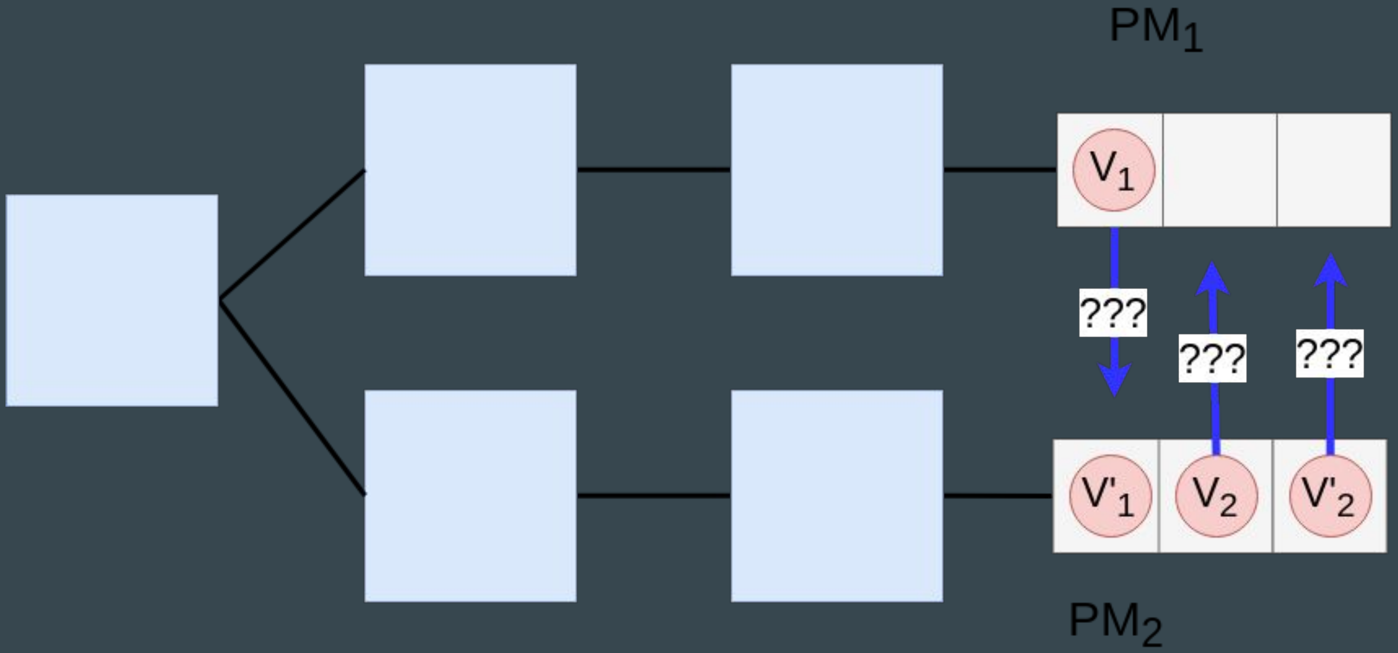


Figure 1: Sample fat tree topology.

What is VM Placement?



What is VM Migration?



Goals

Virtual Machine Placement

- Given:
  - An empty PDDC
  - Policies (Ordered or Unordered)
  - Unplaced VM Pairs with Comm. Frequency
- Output:
  - VM Placement with minimum Comm. cost
- How:
  - Optimal Algorithm
  - Placement Approximation Algorithm

Virtual Machine Migration

- Given:
  - A PDDC
  - Policies (Ordered or Unordered)
  - Placed VM Pairs with new Comm. Frequency
- Output:
  - VM Migration with minimum Comm. & Migration cost
- How:
  - MCF Algorithm
  - Migration Approximation Algorithm

Related Works

Virtual Machine Placement or Migration

- Improving the Scalability of Data Center Networks with Traffic-aware Virtual Machine Placement
    - 2010 Proceedings IEEE INFOCOM
    - X. Meng, V. Pappas, & L. Zhang
    - TrafficAware Algorithm
- PACE: Policy-Aware Application Cloud Embedding
    - 2013 Proceedings IEEE INFOCOM
    - L. E. Li et al.
- PLAN: Joint Policy- and Network-Aware VM Management for Cloud Data Centers
    - 2016 IEEE Transactions on Parallel and Distributed Systems
    - L. Cui et al.
    - PLAN Algorithm

Virtual Machine Placement and Migration

- Joint Virtual Machine Placement and Migration Scheme for Data Centers
  - 2014 IEEE Global Communications Conference
  - T. Duong-Ba, T. Nguyen, B. Bose, & T. Tran
- Traffic-Aware Virtual Machine Migration in Topology-Adaptive DCN
  - 2017 IEEE/ACM Transactions on Networking
  - Y. Cui et al.

System Model

Datacenter

- Fat Tree Topology
  - K-parameter determines number of PMs & switches
- PDDC:
  - Undirected Graph G(V, E)
  - $V = V_P \cup V_S$
  - E is the set all edges
- Physical Machines:
  - i-th PM has $m(i)$ resource slots
  - Each VM requires 1 slot

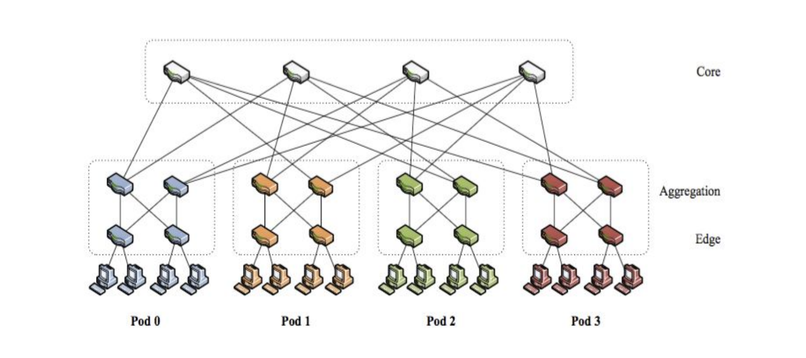
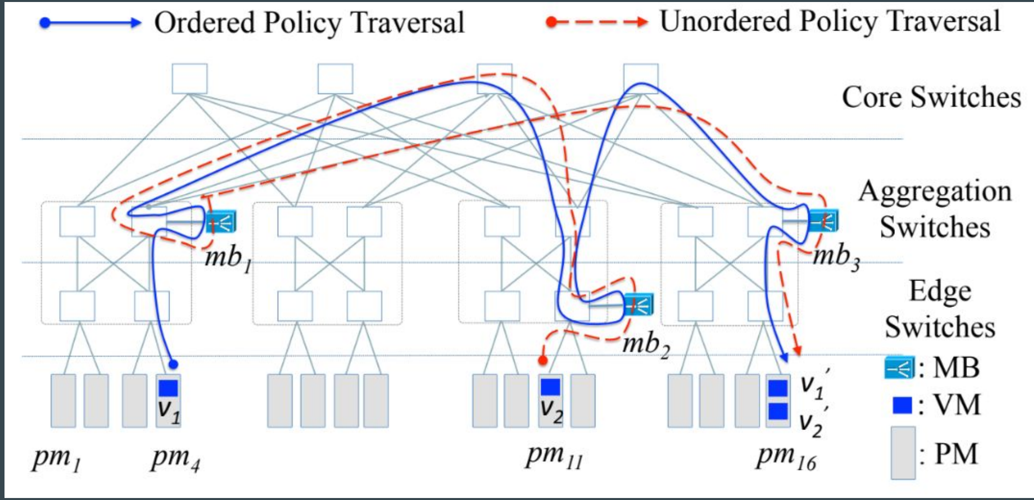


Figure 1: Sample fat tree topology.

Middleboxes

- Set of Middleboxes:
  - $M = \{ mb_1 , mb_2 , \dots , mb_m \}$
- MB Switch:
  - $mb_j \rightarrow sw(j) \in V_s$
- Bump Off the Wire Design

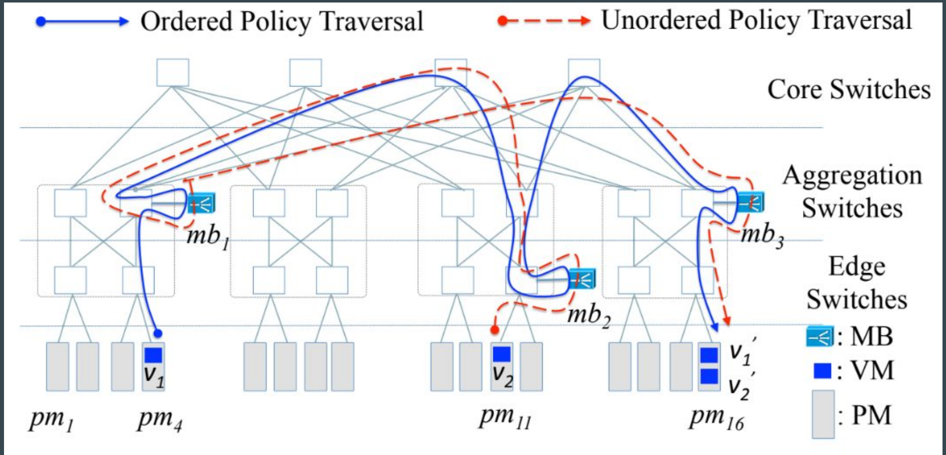


VM Pairs

- VM Pairs:
  - $P = \{ (v_1, v'_1), (v_2, v'_2), \dots, (v_L, v'_L) \}$
  - v_i = Source VM
  - v'_i = Destination VM
- Communication Frequency:
  - $\lambda = \langle \lambda_1, \lambda_2, \dots, \lambda_L \rangle$
  - Non-constant vector

Policies

- Ordered Policies
  - ($mb_1, mb_2, \ldots, mb_m$)
  - Ingress Switch = First MB visited
  - Egress Switch = Last MB visited
  - Sequential MB Dependencies
- Unordered Policies
  - { $mb_1, mb_2, \ldots, mb_m$ }
  - Independant MBs



Costs

- Distance Cost
  - $c(i, j)$
- VM Pair Communication Cost
  - (frequency) * (number of hops)
- VM Pair Migration Cost
  - $\mu * c(i, j)$

Virtual Machine Migration

Ordered Policy Goal

$$C_t(m) = \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big) + \sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i), m(v_i)\big) + \lambda_i \cdot c\big(m(v_i), sw(1)\big)\Big)$$

$$+ \sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i'), m(v_i')\big) + \lambda_i \cdot c\big(sw(m), m(v_i')\big)\Big)$$

Ordered Policy Goal

$$C_t(m) = \boxed{\sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big)} + \boxed{\sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i), m(v_i)\big) + \lambda_i \cdot c\big(m(v_i), sw(1)\big)\Big)}$$

$$\boxed{+ \sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i'), m(v_i')\big) + \lambda_i \cdot c\big(sw(m), m(v_i')\big)\Big)}$$

MB Traversal Cost

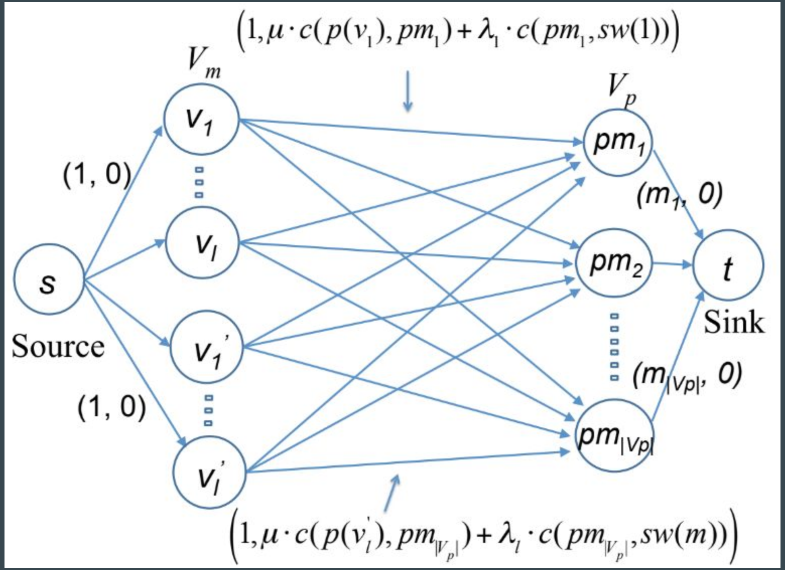
Migration and Ingress Cost

Migration and Egress Cost

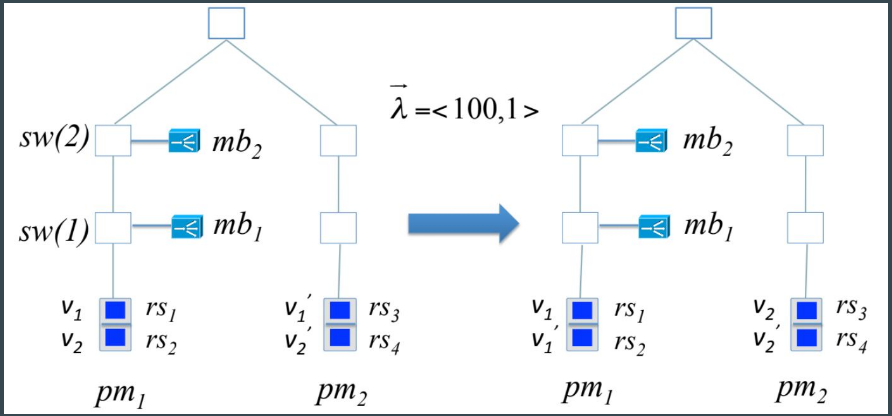
Ordered Policy Solution - MCF Algorithm

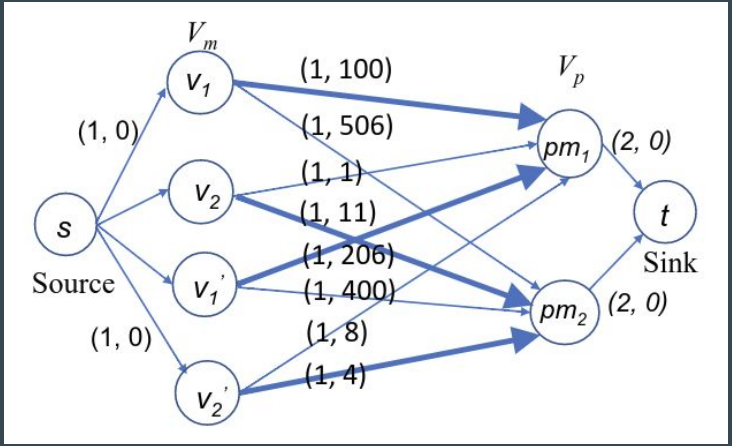
1. Add Source & Sink Node: $V' = \{s\} \cup \{t\} \cup V_m \cup V_p$
2. Connect Source/Sink to VMs/PMs:
   $E' = \{(s,v) : v \in V_m\} \cup \{(v, pm_j : v \in V_m, pm_j \in V_p\} \cup \{(pm_j, t) : pm_j \in V_p\}$
3. Source to VM: capacity 1, cost 0 & PM to Sink: capacity m_j, cost 0
4. Source VM to PM edges: capacity 1, cost:
   $\mu \cdot c\big(p(v_i), pm_j\big) + \lambda_i \cdot c\big(pm_j, sw(1)\big)$
   Destination VM to PM edges: capacity 1, cost:
   $\mu \cdot c\big(p(v_i'), pm_j\big) + \lambda_i \cdot c\big(pm_j, sw(m)\big)$
5. Supply = 2L, Demand = 2L

Ordered Policy Solution - MCF Algorithm



Ordered Policy Solution - MCF Algorithm





Unordered Policy Goal

$$C_t(m, \vec{\pi}) = \sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i), m(v_i)\big) + \mu \cdot c\big(p(v_i'), m(v_i')\big) \Big)$$

$$\sum_{i=1}^{l} \lambda_i \cdot \Big(\sum_{j=1}^{m-1} c\big(sw(\pi^i(j)), sw(\pi^i(j+1))\big) + c\big(m(v_i), sw(\pi^i(1))\big)$$

$$+c\big(sw(\pi^i(m)), m(v_i')\big) \Big)$$

Unordered Policy Goal

$$C_t(m, \vec{\pi}) = \boxed{\sum_{i=1}^{l} \Big(\mu \cdot c\big(p(v_i), m(v_i)\big) + \mu \cdot c\big(p(v_i'), m(v_i')\big) \Big)}$$

$$\sum_{i=1}^{l} \lambda_i \cdot \Big(\boxed{\sum_{j=1}^{m-1} c\big(sw(\pi^i(j)), sw(\pi^i(j+1))\big)} + \boxed{c\big(m(v_i), sw(\pi^i(1))\big)}$$

$$\boxed{+c\big(sw(\pi^i(m)), m(v_i')\big)} \Big)$$

Migration Cost

Cost to First MB

Variable MB Cost

Cost to Last MB

Unordered Policy Solution - Approximation

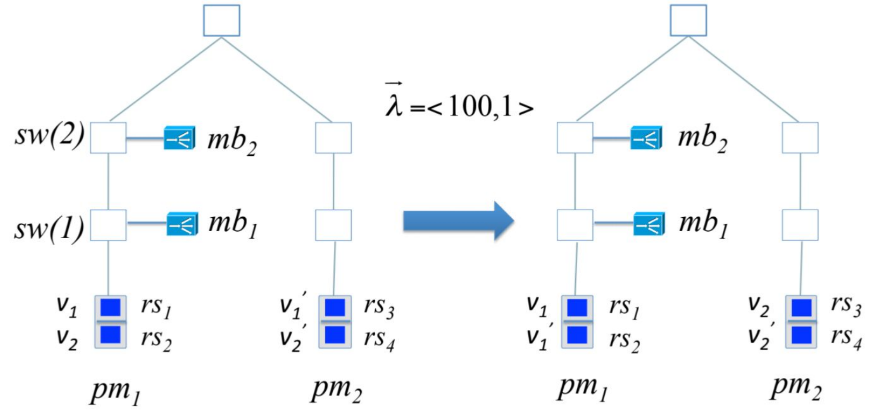
Algorithm 1: VM^{2}P Algorithm for Unordered Policy.

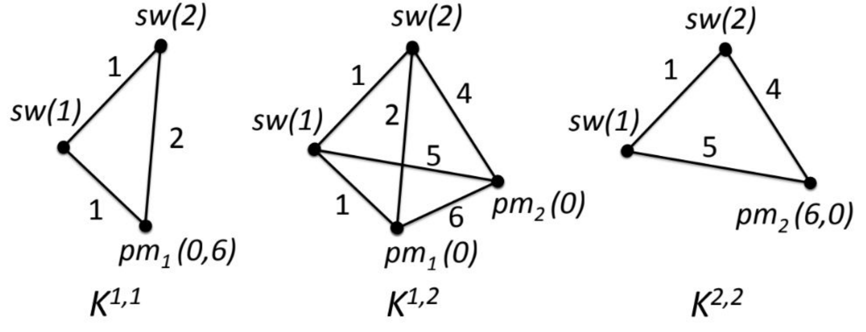
Input: A PDDC with unordered policy $\{mb_1, mb_2, ..., mb_m\}$, VM pairs P with placement p, $V_p = \{pm_i\}$, $m(i)$.

Output: A migration m and its total energy cost $C_t(m, \overrightarrow{\pi})$.

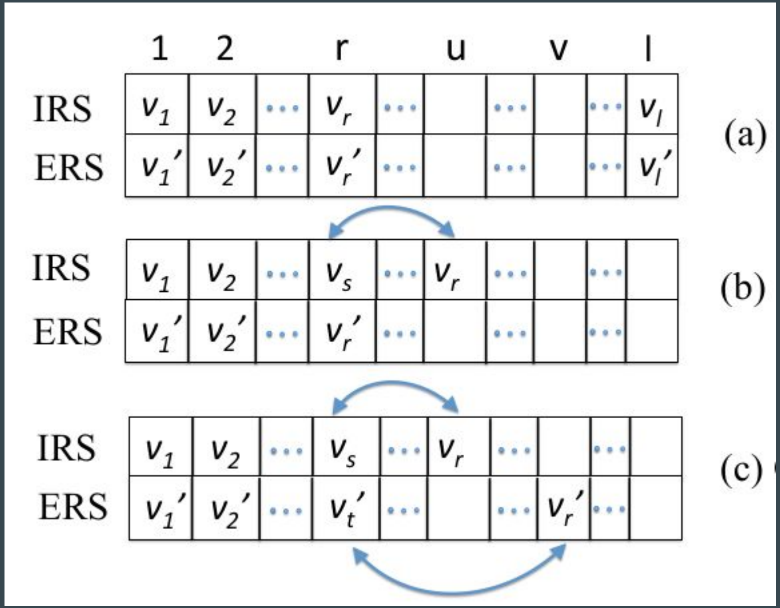
0. $m = \phi$, $C_t(m, \overrightarrow{\pi}) = 0$, $k = 0$
1. **while** $(k \leq l)$ //not all VM pairs are migrated yet
2. $a = 1$, $b = 1$, $c_{min} = \infty$;
3. **for** $(i = 1; i \leq |V_p|; i++)$
4. **if** $(m(pm_i) == 0)$ **break**;
5. **for** $(j = i; j \leq |V_p|; j++)$
6. **if** $(m(pm_j) == 0)$ **break**;
7. **if** $(i == j \wedge m(s_j) \leq 1)$ **break**;
8. $V_K^{i,j} = \{pm_i, pm_j, sw(1), sw(2), ...sw(m)\}$;
9. Construct complete graph $K^{i,j} = (V_K^{i,j}, E_K^{i,j})$;
10. Compute a minimum spanning tree MST for $K^{i,j}$;
11. Compute a walk W from pm_i to pm_j on MST by visiting all vertices using each edge *at most twice*, and calculate the cost $c_{i,j}$ of W;
12. $c(pm_i) = \mu \cdot c(p(v_i), pm_i)$, $c(pm_j) = \mu \cdot c(p(v_i^{'}), pm_j)$;
13. $c_{i,j} = \lambda_k \cdot c_{i,j} + c(pm_i) + c(pm_j)$;
14. **if** $(c_{i,j} < c_{min})$ $a = i$, $b = j$, $c_{min} = c_{i,j}$;
15. **end for;**
16. **end for;**
17. $m = m \cup \{(pm_a, pm_b)\}$;
18. $C_t(m, \overrightarrow{\pi})$ += c_{min};
19. $m(pm_a)--$, $m(pm_b)--$;
20. $k++$; // the next VM pair
21. **end while;**
22. **RETURN** m and $C_t(m, \overrightarrow{\pi})$.

Unordered Policy Solution - Approximation





Sketch of Optimal Proof



Virtual Machine Placement

Ordered Policy Goal

$$C_t(m) = \sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big) +$$

$$\sum_{i=1}^{l} \Big(\lambda_i \cdot c\big(m(v_i), sw(1)\big) + \lambda_i \cdot c\big(sw(m), m(v_i')\big) \Big)$$

Ordered Policy Goal

$$C_t(m) = \boxed{\sum_{i=1}^{l} \lambda_i \cdot \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big)} +$$

$$\boxed{\sum_{i=1}^{l} \Big(\lambda_i \cdot c\big(m(v_i), sw(1)\big) + \lambda_i \cdot c\big(sw(m), m(v_i')\big) \Big)}$$

MB Traversal Cost

Ingress and Egress Cost

Ordered Policy Solution - Optimal

Algorithm 2: VMP^2 Algorithm for Ordered Policy.

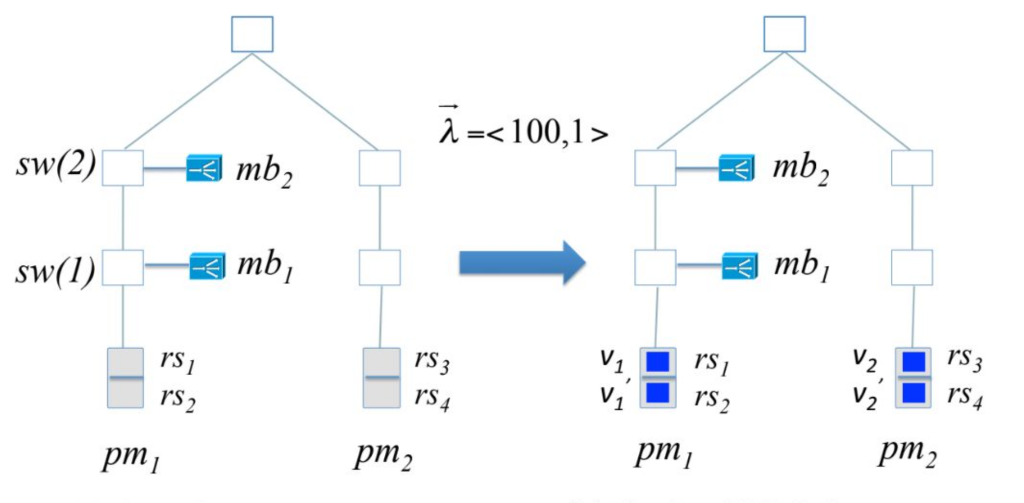
Input: A PDDC with ordered policy $(mb_1, mb_2, ..., mb_m)$, VM pairs P, $V_p = \{pm_i\}$, $m(i)$.

Output: A placement p and the total comm. cost $C_c(p)$.

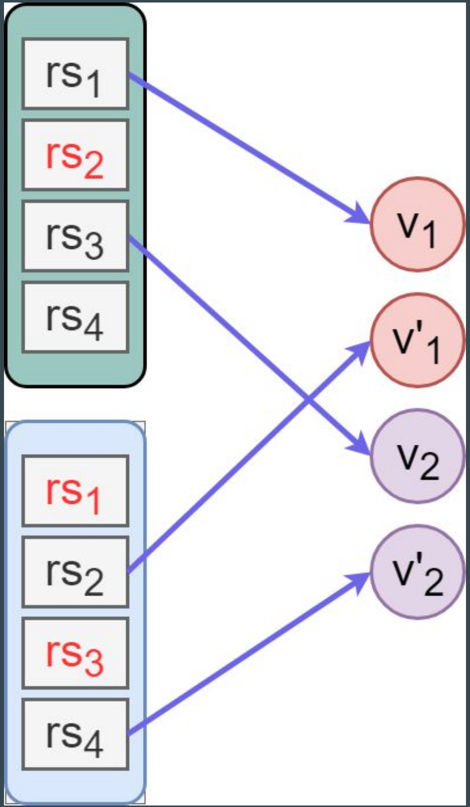
Notations: $\mathcal{I}[i].id$, $\mathcal{I}[i].dist$, $\mathcal{E}[i].id$, $E[i].dist$: ID and cost of the i^{th} resource slot in $\mathcal{I}$ and $\mathcal{E}$.

0. $i = 1$, $j = 1$, $k = 1$, $C_c(p) = 0$,
$\mathcal{I}= \phi$ (empty set), $\mathcal{E}= \phi$, $p = \phi$;
1. Assign all resource slots in the PDDC unique IDs;
2. Sort them in ascending order of their costs to ingress switch $sw(1)$ (and egress switch $sw(m)$), store the first $2l$ resource slots and their costs in an array A (and B);
3. **while** $(k \leq l)$
4.   **while** $(A[i].id \in \mathcal{E})$ $i++$;
5.   **while** $(B[j].id \in \mathcal{I})$ $j++$;
6.   **if** $(A[i].id \neq B[j].id)$
7. $\mathcal{I}[k].id = A[i].id, \mathcal{I}[k].dist = A[i].dist$;
8. $\mathcal{E}[k].id = B[j].id, \mathcal{E}[k].dist = B[j].dist$;
9. i++; j++;
10.   **else**
11.   **while** $(A[i+1].id \in \mathcal{E})$ i++;
12.   **while** $(B[j+1].id \in \mathcal{I})$ j++;
13.   **if** $(A[i+1].dist \leq B[j+1].dist)$
14. $\mathcal{I}[k].id = A[i+1].id, \mathcal{I}[k].dist = A[i+1].dist$;
15. $\mathcal{E}[k].id = B[j].id, \mathcal{E}[k].dist = B[j].dist$;
16. i += 2; j++;
17.   **else**
18. $\mathcal{I}[k].id = A[i].id, \mathcal{I}[k].dist = A[i].dist$;
19. $\mathcal{E}[k].id = B[j+1].id, \mathcal{E}[k].dist = B[j+1].dist$;
20. i++; j += 2;
21.   **end if;**
22.   **end if;**
23. k++;
24. **end while;**
25. $a = \sum_{j=1}^{m-1} c\big(sw(j), sw(j+1)\big)$;
26. **for** $(1 \leq i \leq l)$
27.   Place v_i at resource slot $\mathcal{I}[i].id$;
28.   Place $v_i^{'}$ at resource slot $\mathcal{E}[i].id$;
29. $p = p \cup \{(\mathcal{I}[i].id, \mathcal{E}[i].id)\}$;
30. $C_c(p)$ += $\lambda_i * (\mathcal{I}[i].dist + a + \mathcal{E}[i].dist)$;
31. **end for;**
32. **RETURN** p and $C_c(p)$.

Ordered Policy Solution - Optimal



(a) k =2 fat-tree PDDC (b) Optimal VM placement



Unordered Policy Goal

$$c^{p,\pi^i} = \sum_{i=1}^{l} \left(\boxed{\lambda_i \cdot c\big(p(v_i), sw(\pi^i(1))\big)} + \lambda_i \cdot \sum_{j=1}^{m-1} \left(\boxed{c\big(sw(\pi^i(j), sw(\pi^i(j+1))\big)} \right) + \boxed{\lambda_i \cdot c\big(sw(\pi^i(m)), p(v_i')\big)} \right)$$

Cost to First MB

Variable MB Traversal Cost

Cost to Last MB

Unordered Policy Solution - Approximation

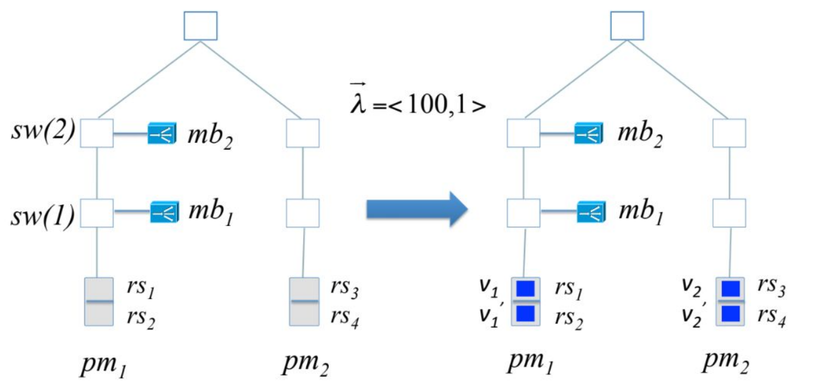
Algorithm 3: VMP^2 Algorithm for Unordered Policy.

Input: A PDDC with unordered policy $\{mb_1, mb_2, ..., mb_m\}$,
VM pairs P, $V_p = \{pm_i\}$, $m(i)$.

Output: A placement p and the total comm. cost $C_c(p, \overrightarrow{\pi})$.

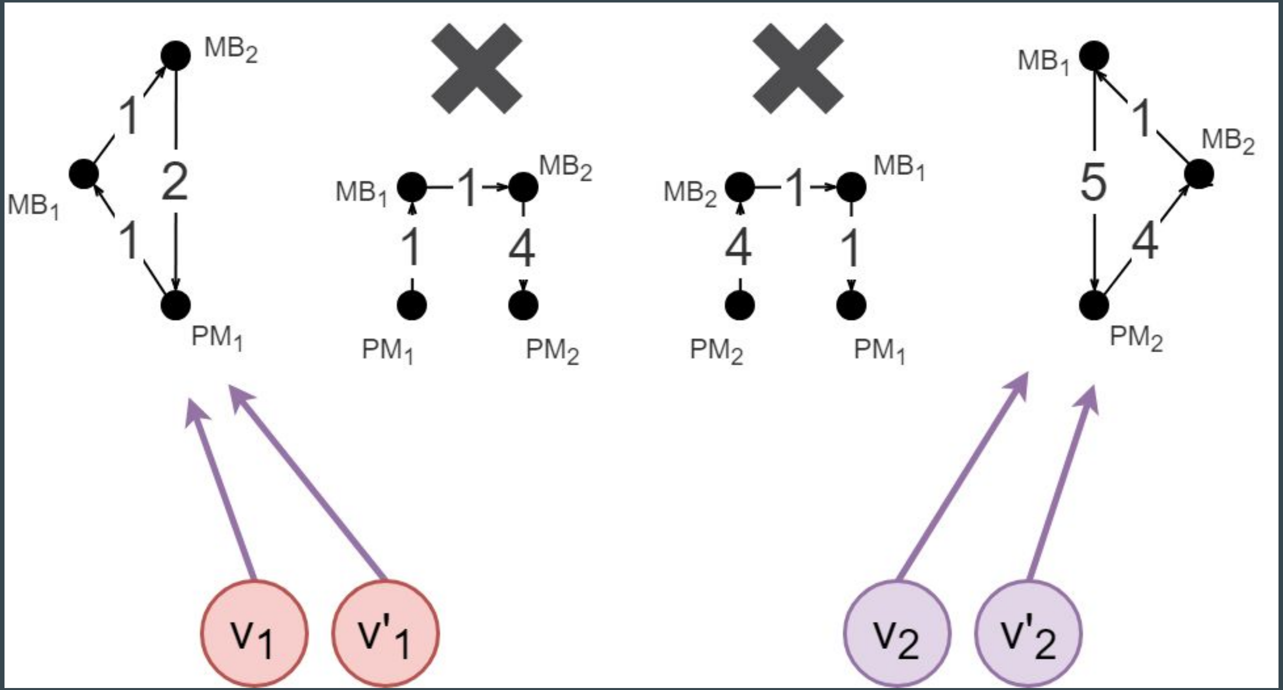
0. $X = \phi$; // stores each PM pair and the cost of the walk
1. **for** $(i = 1; i \leq |V_p|; i++)$
2. **for** $(j = i; j \leq |V_p|; j++)$
3. $V_K^{i,j} = \{pm_i, pm_j, sw(1), sw(2), ...sw(m)\}$;
4. Construct complete graph $K^{i,j} = (V_K^{i,j}, E_K^{i,j})$;
5. Compute a minimum spanning tree MST for $K^{i,j}$;
6. Compute a walk W from pm_i to pm_j on MST by visiting all vertices using each edge *at most twice*, and calculate the cost $c(i,j)$ of W.
7. $X = X \cup \{(i, j, c(i,j))\}$;
8. **end for;**
9. **end for;**
10. Sort X in non-descending order of $c(i,j)$.
Let $X = \{(s_1, t_1, c(s_1, t_1)), (s_2, t_2, c(s_2, t_2)), ...\}$;
11. $i = 1$, $j = 1$; // the i^{th} VM pair $(v_i, v_i^{'})$ is placed at the j^{th} PM pair (s_i, t_j) in X;
$p = \phi$, $C_c(p, \overrightarrow{\pi}) = 0$;
12. **while** $(i \leq l)$ //not all VM pairs are placed yet
13. **while** $\left(m(s_j) \geq 1 \wedge m(t_j) \geq 1\right)$
14. Place v_i at PM s_j, place $v_i^{'}$ at PM t_j;
15. $p = p \cup \{(s_j, t_j)\}$;
16. $C_c(p, \overrightarrow{\pi}) += \lambda_i * c(s_j, t_j)$;
17. $m(s_j)--$, $m(t_j)--$;
18. $i++$;
19. **if** $(i > l)$ **break**;
20. **end while;**
21. $j++$; // the next available PM pairs
22. **end while;**
23. **RETURN** p and $C_c(p, \overrightarrow{\pi})$.

Unordered Policy Solution - Approximation



(a) k =2 fat-tree PDDC (b) Optimal VM placement

Unordered Policy Solution - Approximation

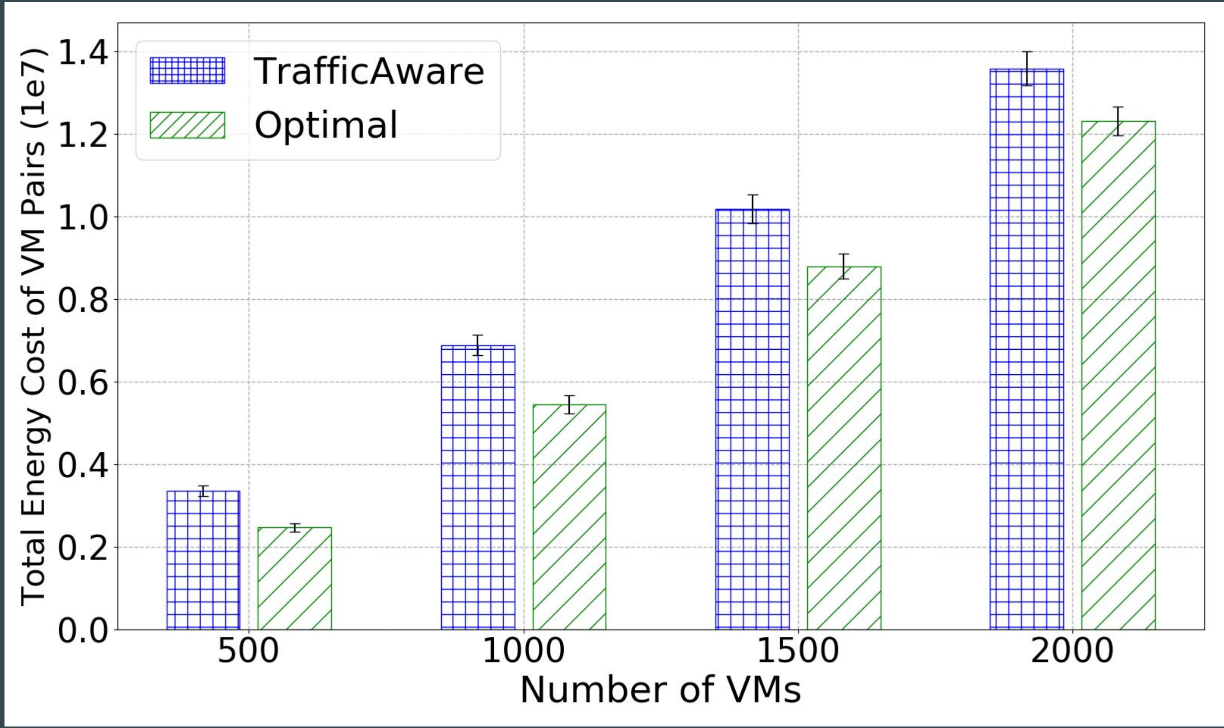


Performance Evaluation

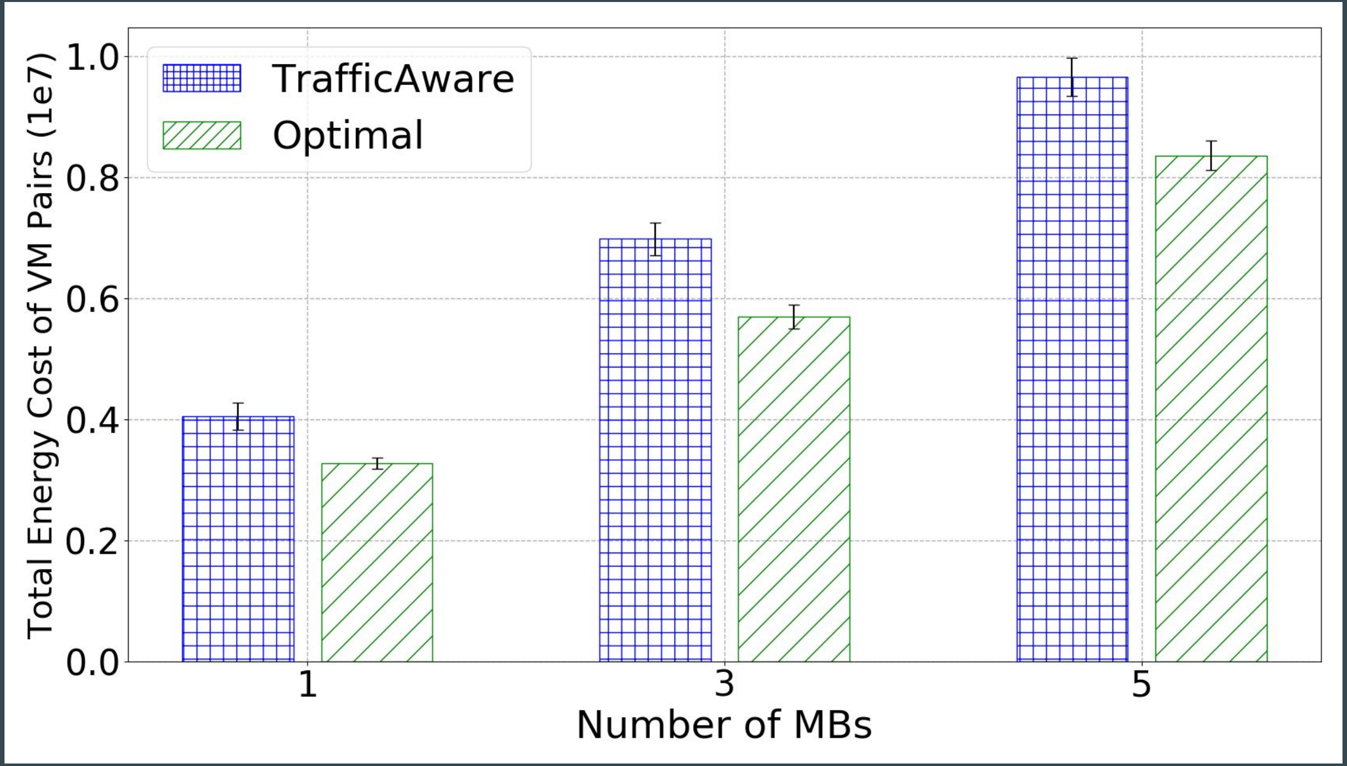
Common Simulation Parameters

- Fat Tree Topology (k = 8)
  - 128 Physical Machines
  - Frequency Range [1, 1000]
- Varying One of the Following (Placement):
  - Number of VM Pairs
  - Number of MBs
  - Number of Resource Slots
- Varying Mu Parameter (Migration)

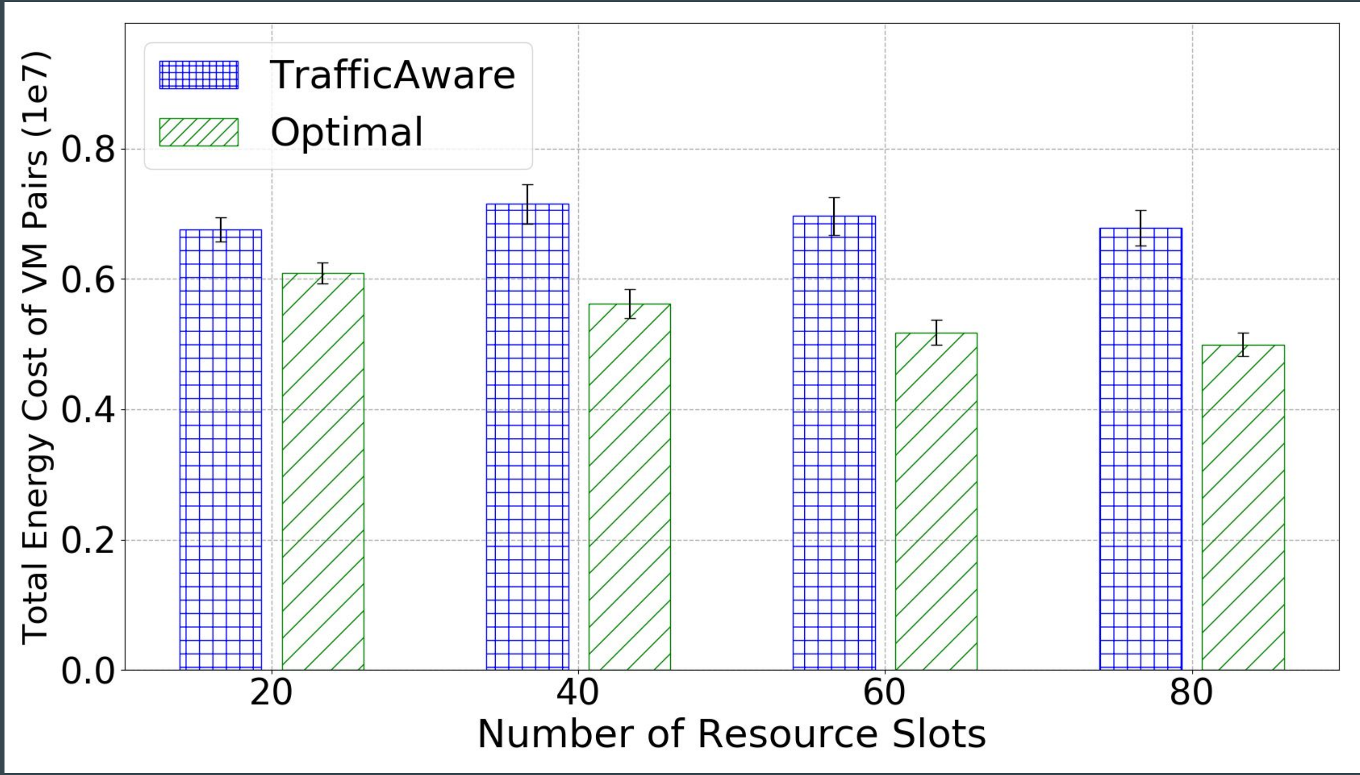
Ordered Placement - VM Simulation (rc = 40, mb = 3)



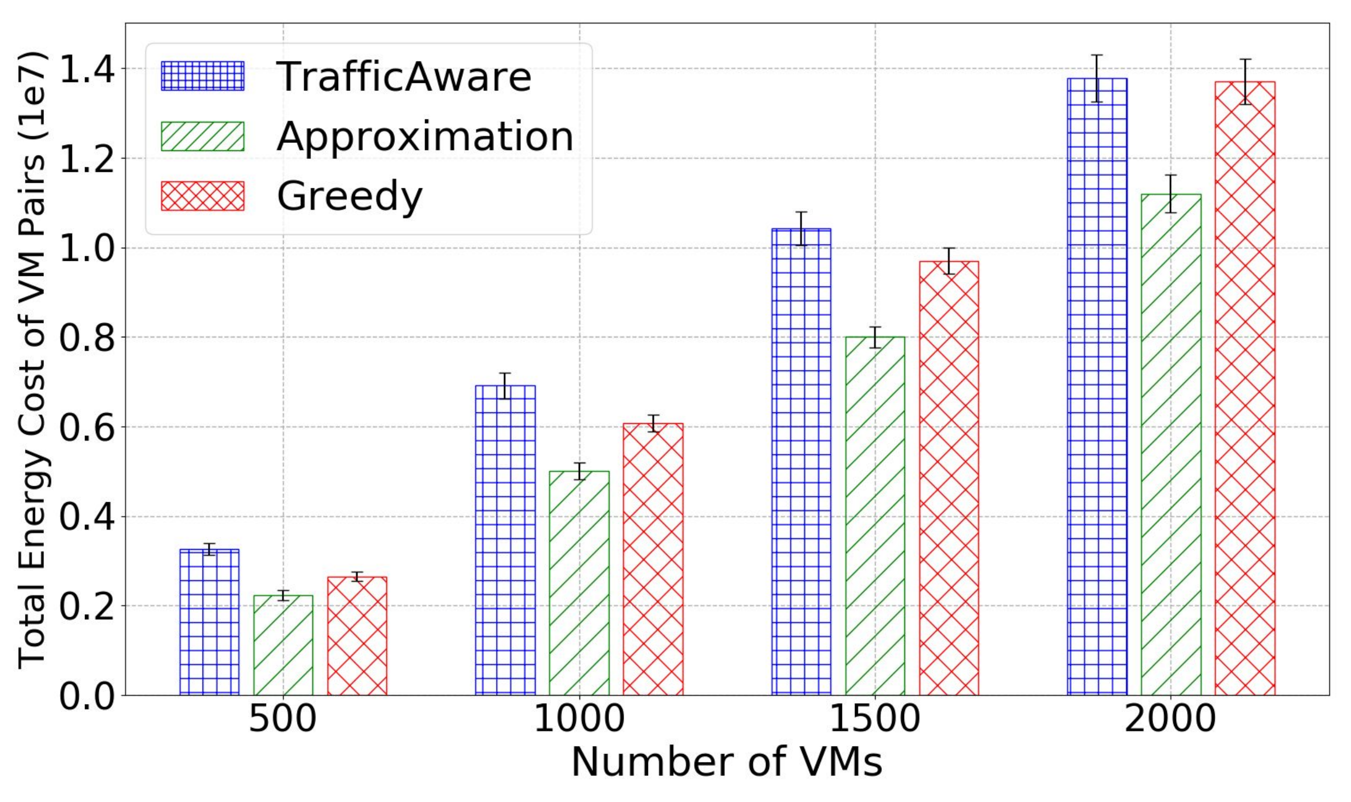
Ordered Placement - MB Simulation (rc = 40, l = 1000)



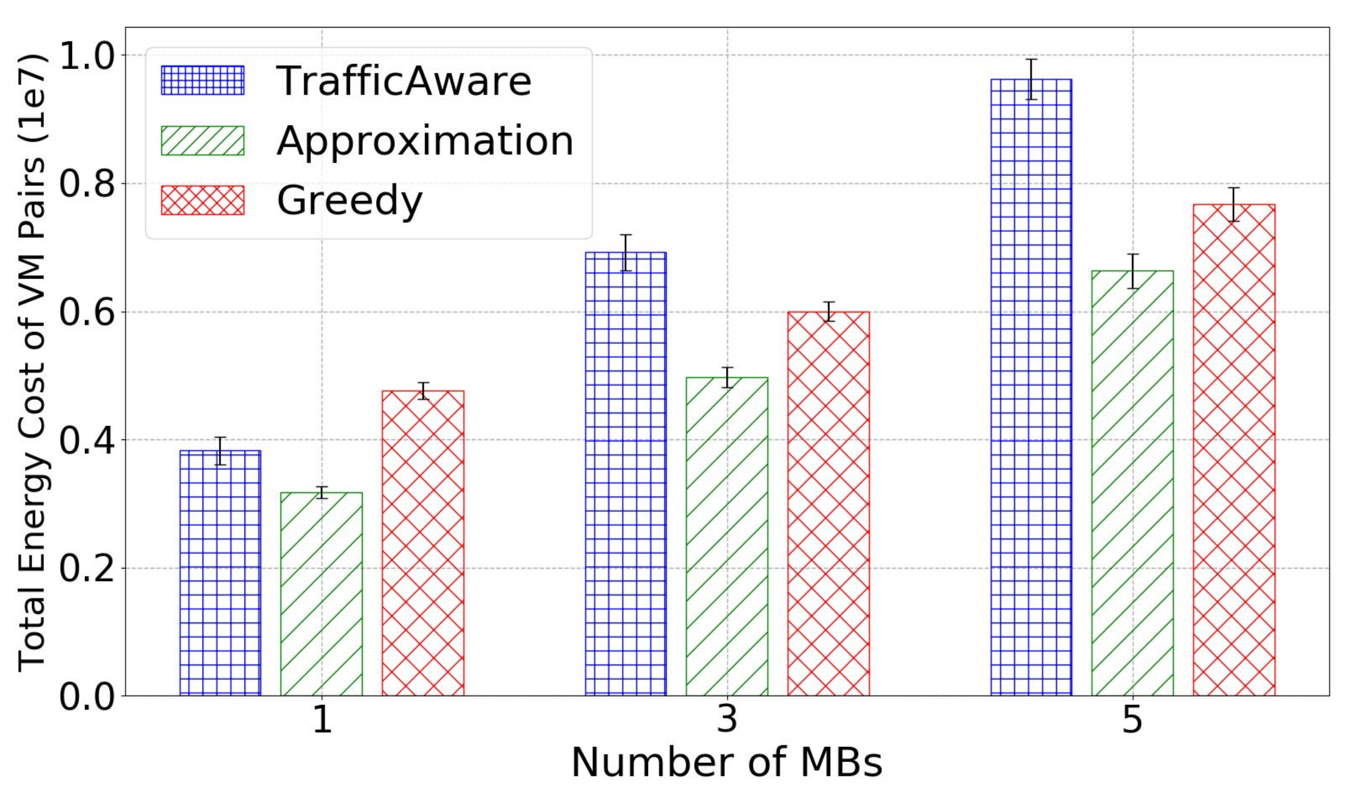
Ordered Placement - RC Simulation (I = 1000, mb = 3)



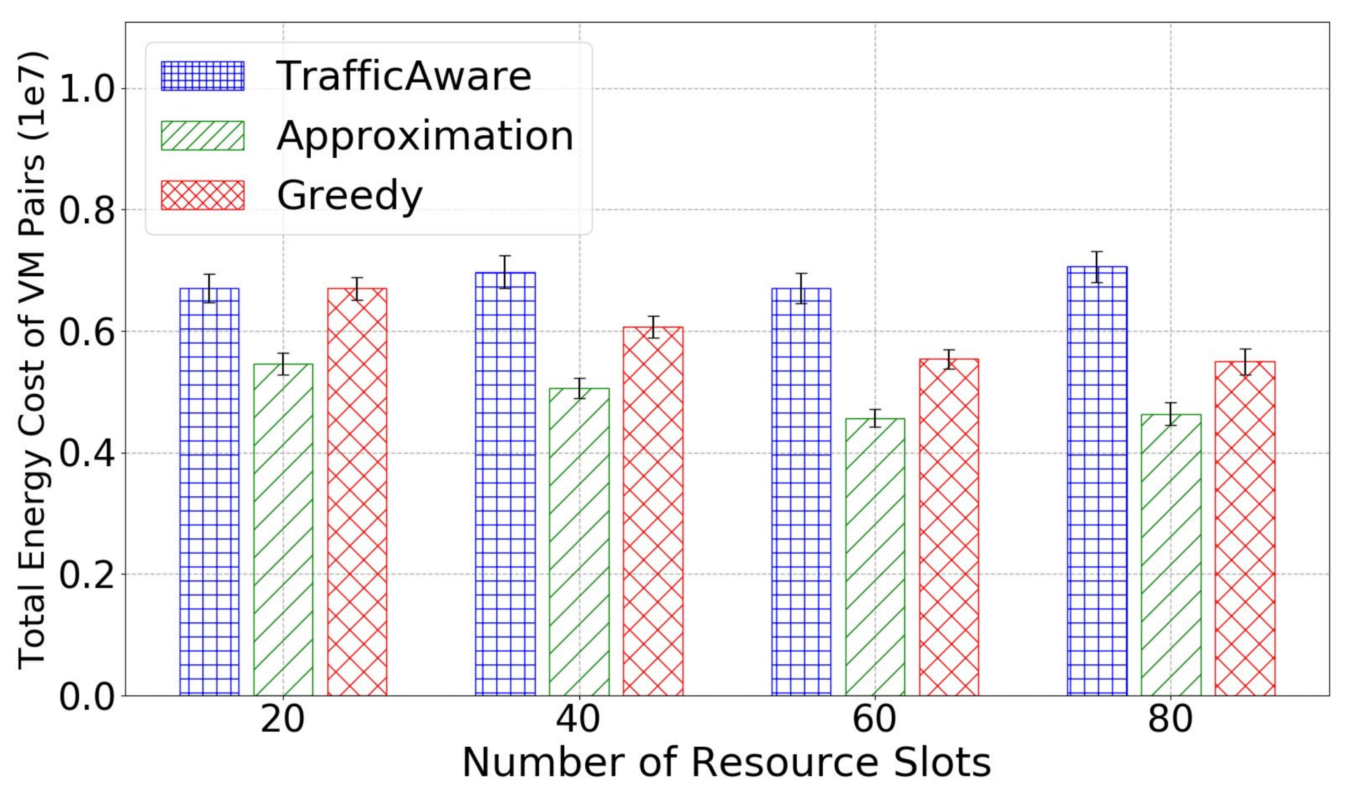
Unordered Placement - VM Simulation (rc = 40, mb = 3)



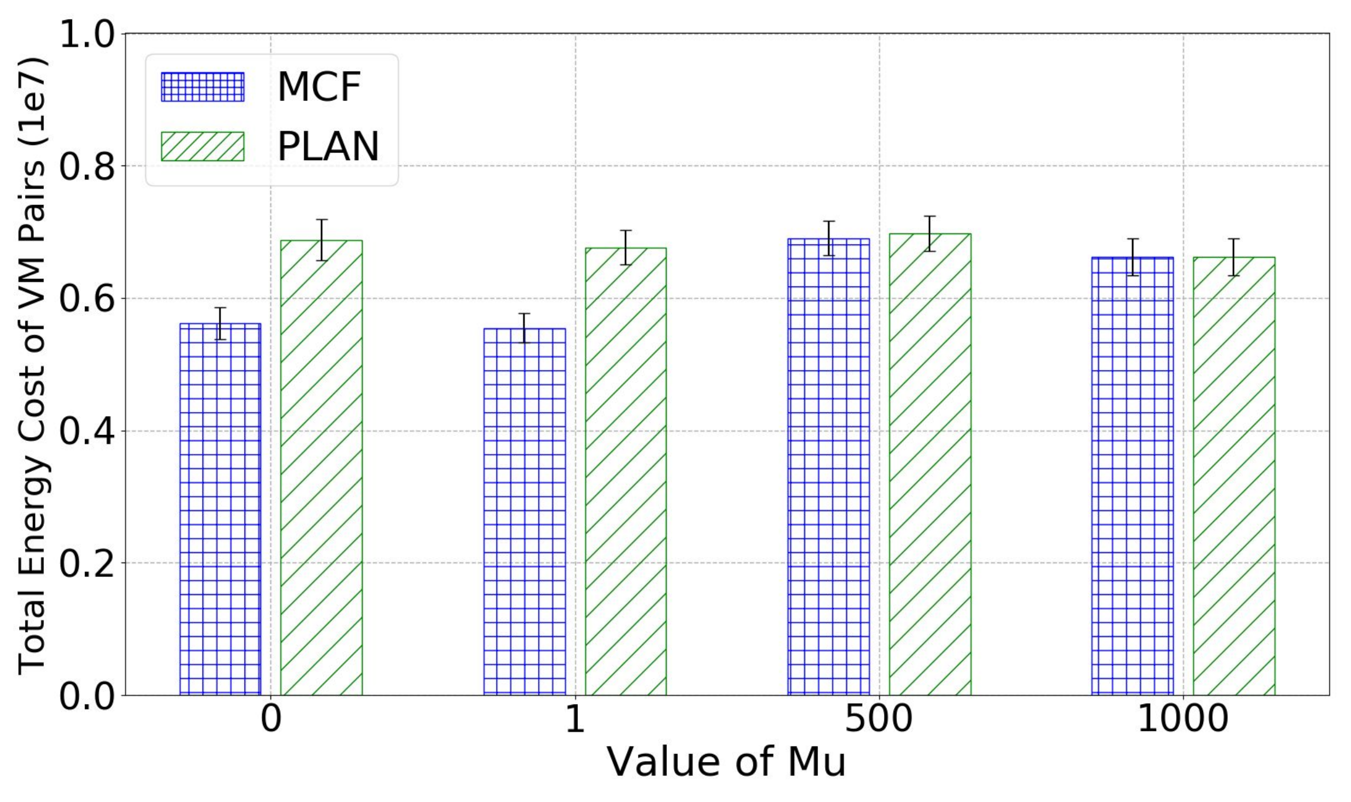
Unordered Placement - MB Simulation (rc = 40, l = 1000)



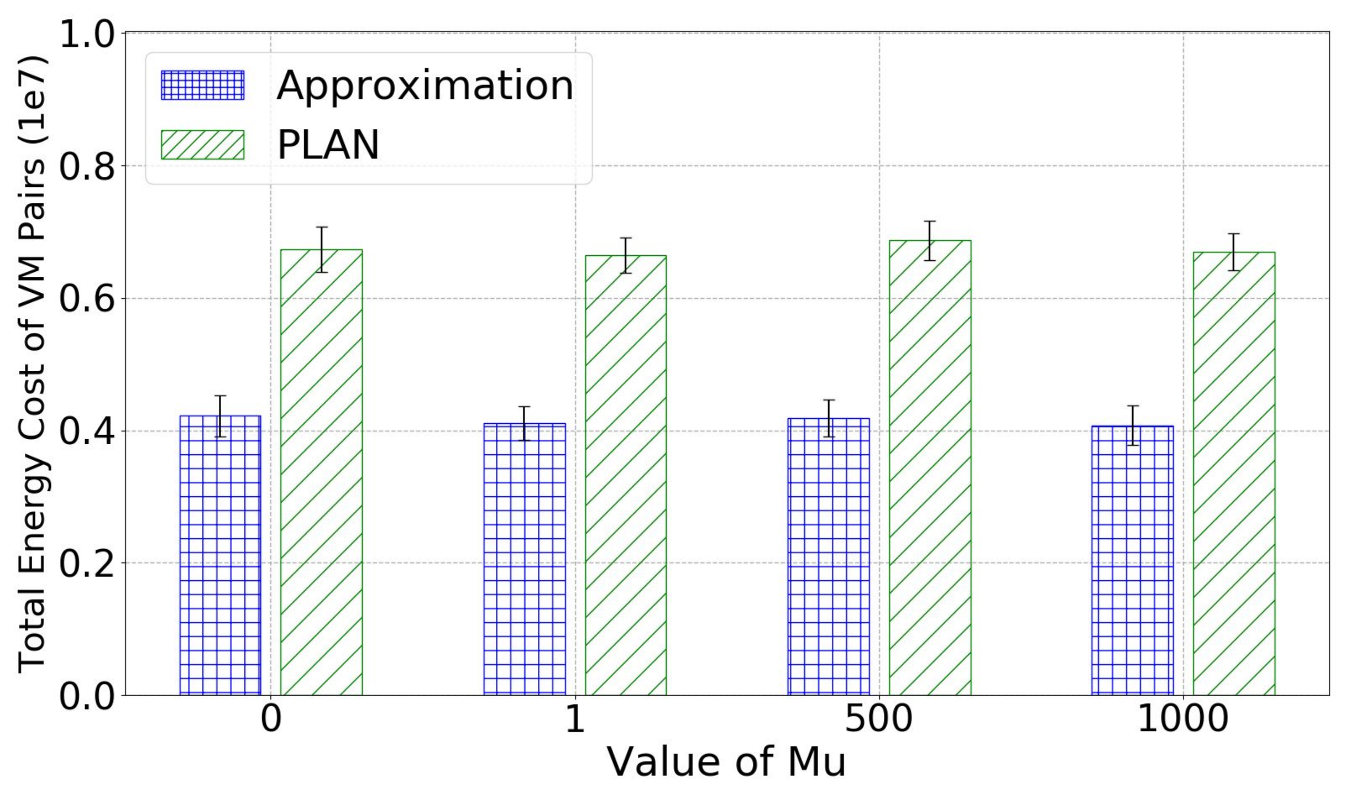
Unordered Placement - RC Simulation (l = 1000, mb = 3)



Ordered Migration - (I = 1000, mb = 3, rc=40)



Unordered Migration - (I = 1000, mb = 3, rc=40)



Conclusion

Conclusion

- Placement Special Case of Migration
- Ignoring PDDC constraints leads to Inefficiencies
- Future Work:
    - Testing in Real Networks
    - Variable ‘sized’ VMs
    - Network Function Virtualization (NFVs)