# Subgraph Matching for Resource Allocation in the Federated Cloud Environment

Atakan Aral, Tolga Ovatman

Istanbul Technical University – Department of Computer Engineering

June 27, 2015

# Outline

# Outline

# Geo-Distributed Clusters

- Opportunities:
  - Available mechanisms and policies such as **Federated Cloud**;
  - Very high speed inter-DC communication technologies such as **optical fiber**;
  - Programming models that minimize size of data flow between nodes such as **MapReduce**
- Advantages:
  - fault tolerance
  - vendor independence
  - closer proximity to user base
  - cost benefits

# Geo-Distributed Clusters

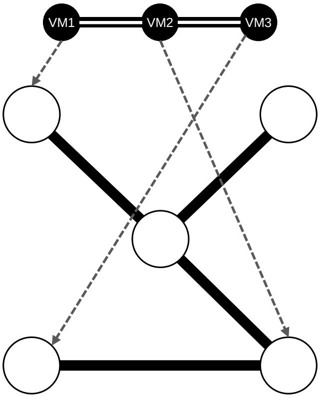
- Risks (regarding VM placement):
  - Cooperating VMs on distant DCs;
  - Clusters far away from their user base;
  - VMs placed without considering different pricing strategies of vendors
- Our Objectives:
  - To decrease **communication delay** (by placing connected VMs to the neighbour data centers)
  - To decrease **deployment delay** (by placing VMs close to the broker)
  - To reduce **resource costs** (by balancing load and avoiding overload in any DC)

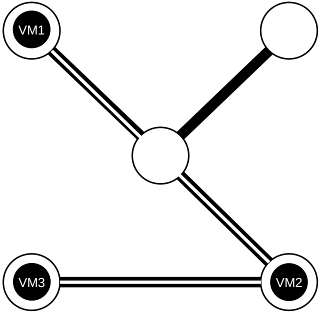
# Outline

# Topology Modeling

- Weighted, undirected, simple graphs
- Vertice represent cloud data centers / requested VMs.
  - CPU, Memory, Storage
- Edges represent the network connections between them.
  - Bandwidth, Latency
- Brokers represent the user base at each node

# Bandwidth Modeling



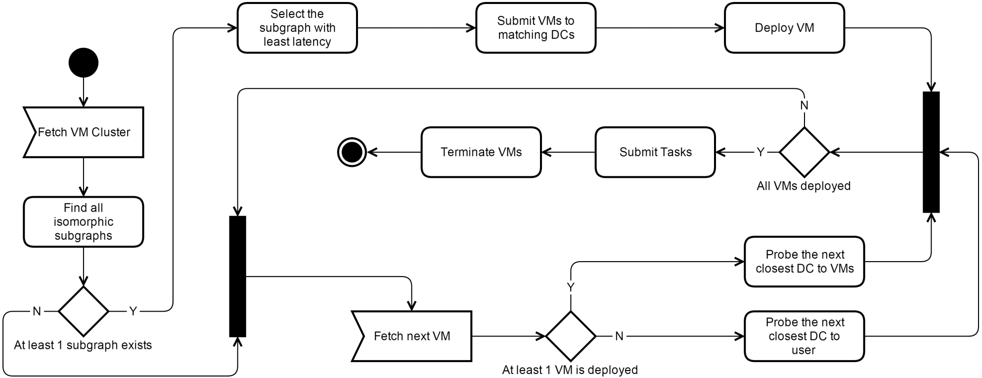



# Cost Modeling

1. Fixed pricing based on memory, bandwidth and duration.
2. Dynamic pricing via Yield management
   - Increase the price of the resource that is running low in a DC
   - $Cost = minCost + (maxCost - minCost) * Util$

# Outline

# Topology Based Matching

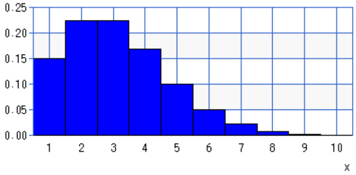


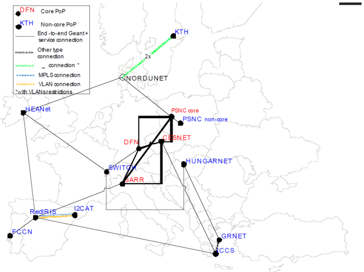
# Outline

# Experimental Setup

**Number of Clusters** Based on the population density around each location.

**Number of VMs** Based on Poisson distribution: $\lambda = 3$

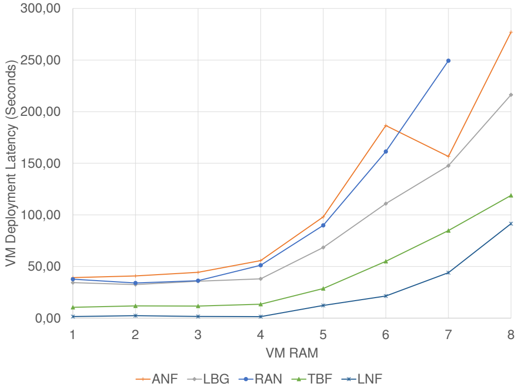


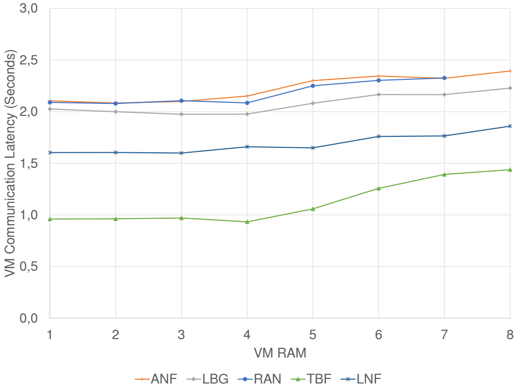




**Cluster Topologies** Either linear or complete

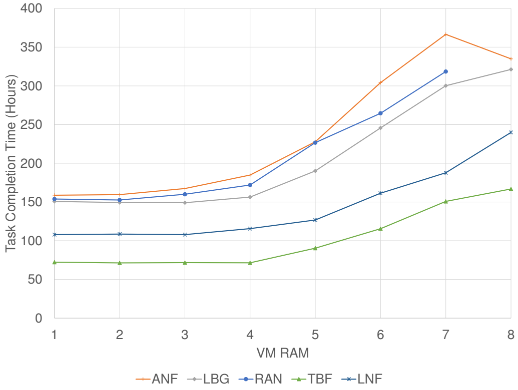
**Arrival Times** Uniform random in the range $[0, 50)$

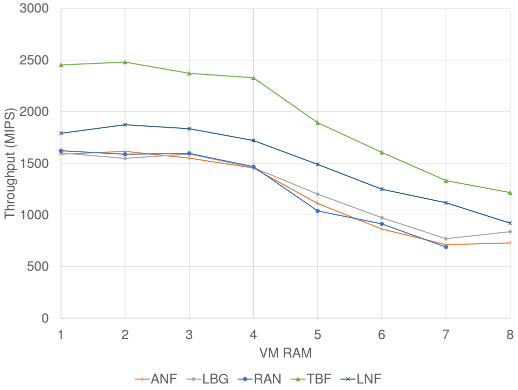
# Latencies


300,00
250,00
200,00
150,00
100,00
50,00
0,00
VM Deployment Latency (Seconds)
1 2 3 4 5 6 7 8
VM RAM
ANF LBG RAN TBF LNF

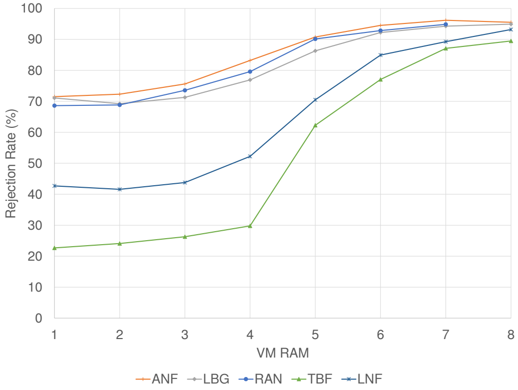

3,0
2,5
2,0
1,5
1,0
0,5
0,0
VM Communication Latency (Seconds)
1 2 3 4 5 6 7 8
VM RAM
ANF LBG RAN TBF LNF

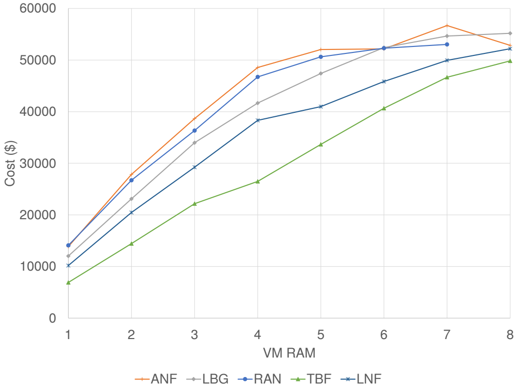
# Duration and Throughput



# Rejection Rate and Cost



# Thank you for your attention.

Atakan Aral
Istanbul Technical University
Department of Computer Engineering
*aralat@itu.edu.tr*

# Outline

# Subgraph Matching

- Search space is all possible injective matchings from the set of pattern nodes to the set of target nodes.
- Systematically explore the search space:
  - Start from an empty matching
  - Extend the partial matching by matching a non matched pattern node to a non matched target node
  - Backtrack if some edges are not matched
  - Repeat until all pattern nodes are matched (success) or all matchings are already explored (fail).
- Filters are necessary to reduce the search space by pruning branches that do not contain solutions.

# LAD Filtering

**Algorithm 1.** LAD-filtering

**Input**: A set $S$ of couples of pattern/target nodes to be filtered

**Output**: failure (if an inconsistency is detected) or success

In case of success, domains are filtered so that $\forall u \in N_p, \forall v \in D_u$, there exists a matching of $G_{(u,v)}$ that covers $adj(u)$.

**while** $S \neq \emptyset$ **do**

  Remove a couple of pattern/target nodes $(u, v)$ from $S$

  **if** *there does not exist a matching of* $G_{(u,v)}$ *that covers* $adj(u)$ **then**

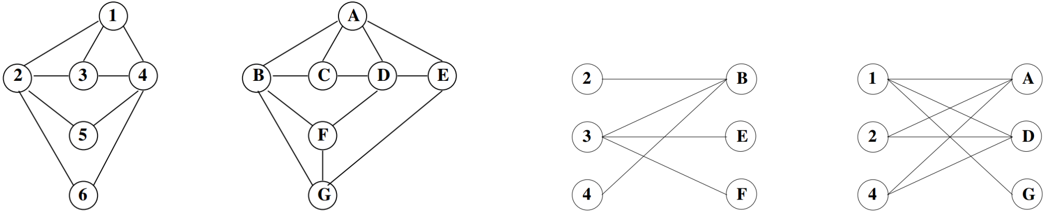
    Remove $v$ from $D_u$

    **if** $D_u = \emptyset$ **then return** failure

    $S \leftarrow S \cup \{(u', v') \mid u' \in adj(u), v' \in adj(v) \cap D_{u'}\}$

**return** success

# LAD Filtering




Pattern graph $G_p$ — Target graph $G_t$ — $G_{(1,G)}$ — $G_{(3,E)}$

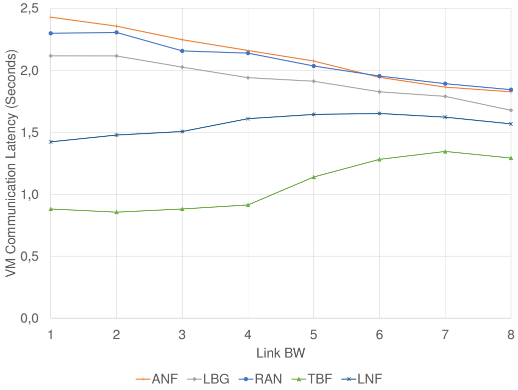
$$D_1 = D_3 = D_5 = D_6 = A, B, C, D, E, F, G$$

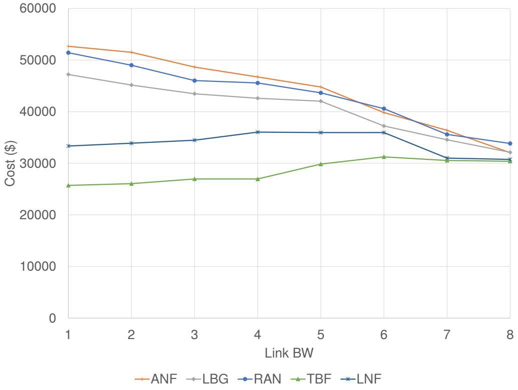
$$D_2 = D_4 = A, B, D$$

# Future Work

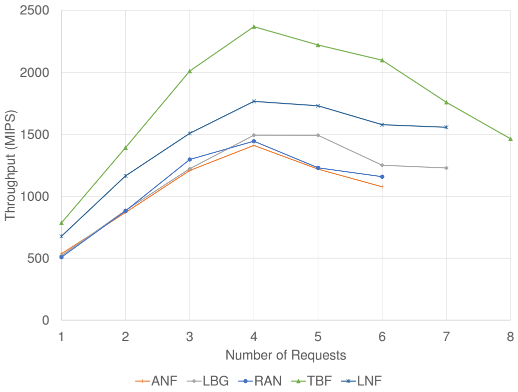
- Algorithm
  - Additional constraints (jurisdiction, partially known topology)
  - Vertical scaling support
  - Hybrid cloud support
  - Homeomorphism
  - Connected components
- Evaluation
  - Significance study
  - Evaluation with topology improvements
  - Multi-objective optimization
  - Dynamic heuristic selection, meta-heuristics

# Link bandwidth request



# Minimum number of requests



# VM network intensity