Resource Mapping Optimization for Distributed Cloud Services

PhD Thesis Defense

Atakan Aral

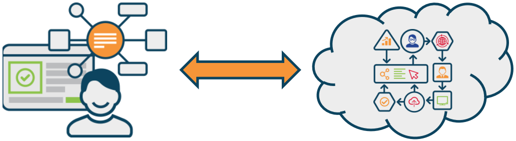
Thesis Advisor: Asst. Prof. Dr. Tolga Ovatman

Istanbul Technical University – Department of Computer Engineering

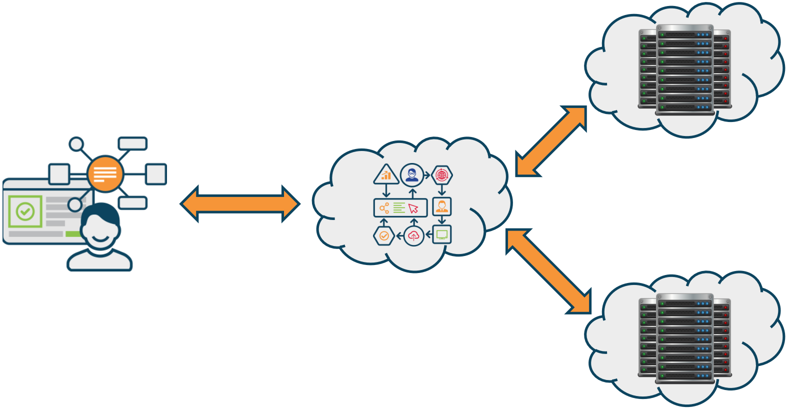
November 3, 2016

Outline

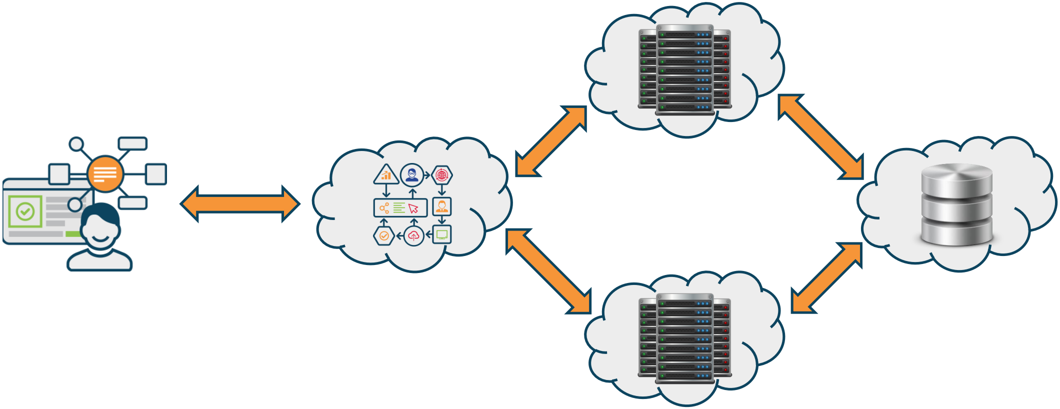
Motivation



Motivation



Motivation



Motivation

- Scientific computing
- Internet of Things
- Finance services
- Health care systems
- e-Learning
- (Mobile) image processing, VR
- Social networking
- On-demand or live video streaming
- Online gaming
- . . .

- **Real time, distributed, data- and computation-intensive cloud services** need low latency and low-cost communication.

Outline

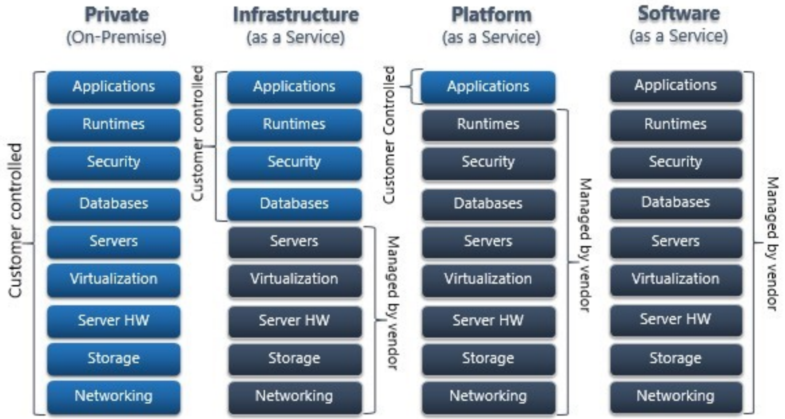
Cloud Computing

Definition

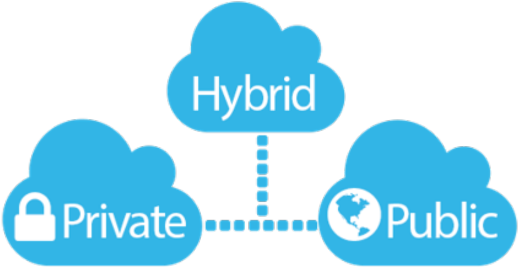
Applications and services that run on a distributed network using *virtualized* resources and accessed by common Internet protocols and networking standards.

- **Broad network access:** Platform-independent, via standard methods
- **Measured service:** Pay-per-use, e.g. amount of storage/processing power, number of transactions, bandwidth etc.
- **On-demand self-service:** No need to contact provider to provision resources
- **Rapid elasticity:** Automatic scale up/out, illusion of infinite resources
- **Resource pooling:** Abstraction, virtualization, multi-tenancy

Cloud Computing – Service Models



Cloud Computing – Deployment Models



Inter-Cloud

Definition

A cloud model that allows on-demand reassignment of resources and transfer of workload through an interworking of cloud systems of different cloud providers.

- Depending on the initiator of the Inter-Cloud endeavour:
  - **Cloud Federation:** A voluntary collaboration between a group of cloud providers to exchange their resources
  - **Multi-Cloud:** An uninformed combination of multiple clouds by a service provider to host the components of the service

Edge Computing

Definition

Pushing the frontier of computing applications, data, and services away from centralized nodes to the logical extremes of a network (e.g. mobile devices, sensors, micro data centers, cloudlets, routers, modems, . . .

- Low latency access to powerful computing resources
- **Cyber-foraging:** Computation or data offloading from mobile devices to servers for extending computation power, storage capacity and/or battery life.

Outline

General Problem

- Deployment on multiple clouds can benefit cloud services
  - Allows seemingly infinite scalability,
  - Provides better geographical coverage,
  - Avoids vendor lock-in and eases hybridization,
  - Increases fault tolerance and availability,
  - Allows exploiting differences in pricing schemes, . . .
- Business solutions that allow basic inter-cloud deployment are already here, e.g. *Nuvla, EGI, RightScale, Equinix, . . .*
- However, such solutions leave cloud data center selection to user.
- There is no **automatic mapping** between provider resources and user requirements.

General Problem *(continued)*

- Resource mapping in distributed cloud is a complex problem due to factors such as:
  - Multiple objectives and perspectives (QoS, **access latency**, throughput, availability, **cost**, profit, . . .),
  - Constraints (SLAs, **processing capacity**, **network bandwidth**, energy consumption, . . .),
  - Geographical distribution and nonuniform network conditions,
  - Heterogeneity and dynamicity of both entities (i.e. resources and requests),
  - The large number of the entities (especially in the case of Edge Computing),
  - Inter-dependencies among the components that constitute a cloud service (e.g. Virtual Machines, Databases).

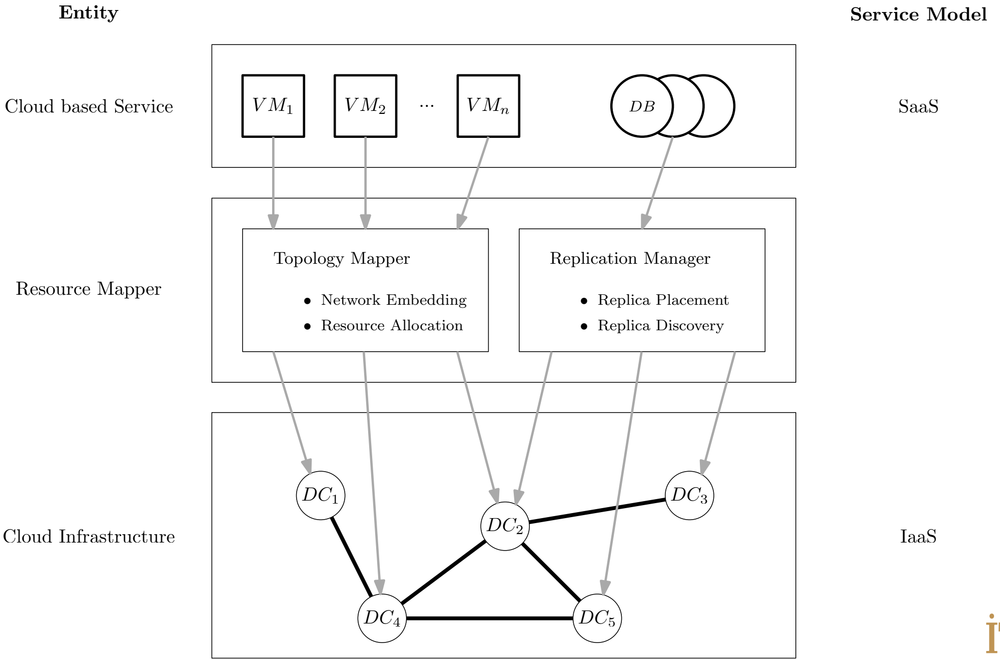
Outline

Proposed Solution

Hypothesis

Employing **resource mapping algorithms** that consider and utilize **structural characteristics** of the distributed cloud services would increase the **QoS** experienced by service users in a **cost-efficient** way.

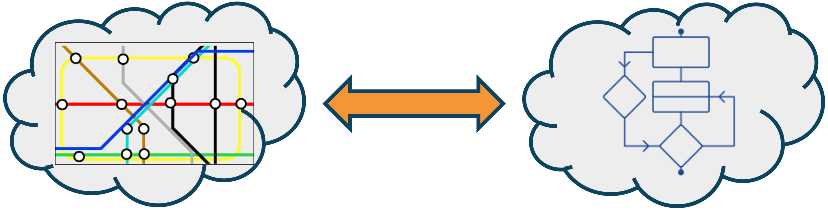
- QoS indicators are: (i) access latency to the service, (ii) access latency between dependent components, and (iii) access latency to the data.
- Cost indicators are: (i) VM provisioning cost, (ii) data storage cost, and (iii) data transfer (bandwidth) cost.


Entity
Service Model
Cloud based Service
VM_1
VM_2
$\cdots$
VM_n
DB
SaaS
Resource Mapper
Topology Mapper
• Network Embedding
• Resource Allocation
Replication Manager
• Replica Placement
• Replica Discovery
Cloud Infrastructure
DC_1
DC_2
DC_3
DC_4
DC_5
IaaS

Outline

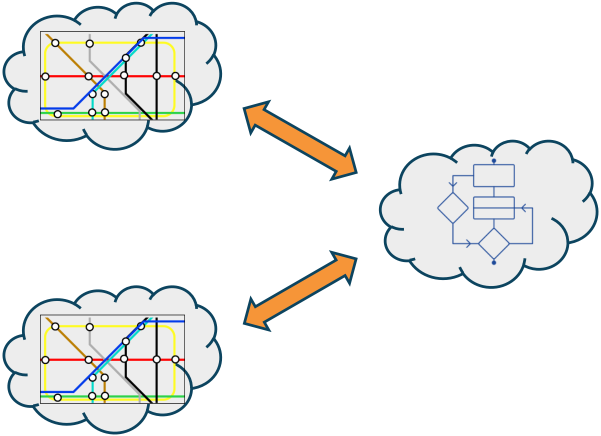
Motivating Example

- A small cloud-based navigation service for public transportation
- Two-tier architecture: User interface and route planning
  - VM 1: 8 CPU cores, 16 GB memory, 2 TB HDD storage
  - VM 2: 32 CPU cores, 60 GB memory, 80 GB SSD storage
- 500 Mbps of dedicated bandwidth between the two tiers is desired



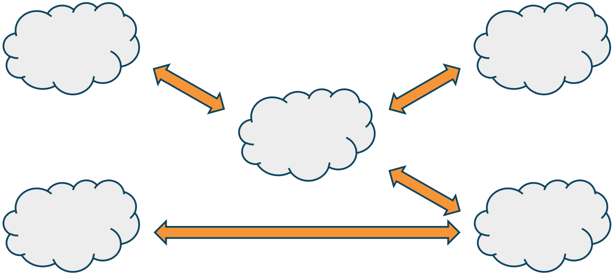
Motivating Example *(continued)*

- First tier is replicated in two separate DCs to improve:
  - Availability
  - Fault tolerance
  - Proximity
- Second tier is not replicated due to:
  - Economical constraints
  - Uncriticality to the service
- But it is still deployed on a third DC because of:
  - Pricing differences
  - Fairness



Motivating Example *(continued)*

- There are 5 (federated) cloud providers in the area.
- Not all have dedicated network connections between them.
- Heterogeneous bandwidth capacities and latencies
- Heterogeneous resource capacities and loads



- **How to map user virtual machines to cloud data centers?**

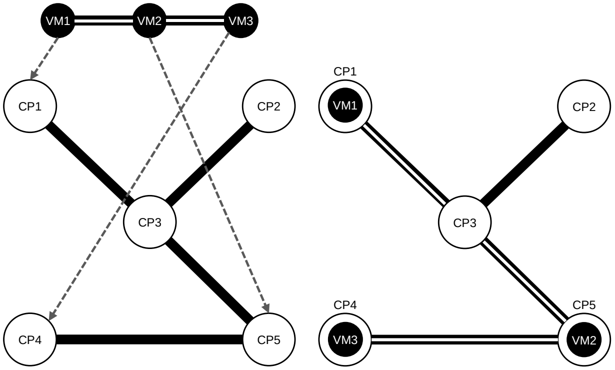
Outline

Virtual Machine Cluster Embedding

- Mapping VM clusters across inter-cloud infrastructure (node & edge mapping)

- Reduce **inter-cloud latency**
- Reduce makespan
- Reduce resource costs

- Optimize **bandwidth utilization**
- Increase system throughput
- Increase acceptance rate
- Increase revenue



Virtual Machine Cluster Embedding *(continued)*

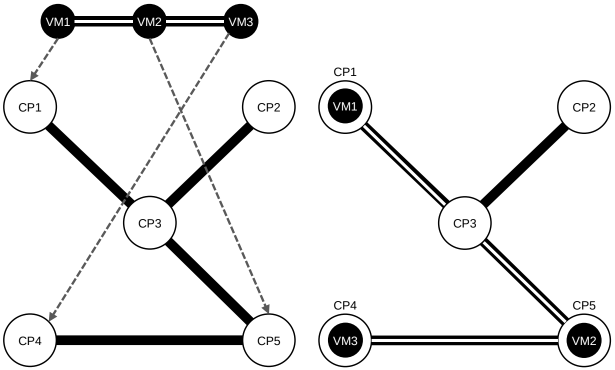
$$G_C = \left(V_C, E_C, A_C^V, A_C^E\right)$$

$$G_F = \left(V_F, E_F, A_F^V, A_F^E\right)$$

$$\forall (v \in V_C) \; \exists (v' \in V_F) \mid v \mapsto v'$$

$$\forall (e \in E_C) \; \exists (E' \subseteq E_F) \mid e \mapsto E'$$

- Validity conditions:
  1. Capacity sufficiency
  2. Path creation
  3. No cycle forming



$$f = (VM_1 \mapsto CP_3, VM_2 \mapsto CP_5, VM_3 \mapsto CP_4)$$

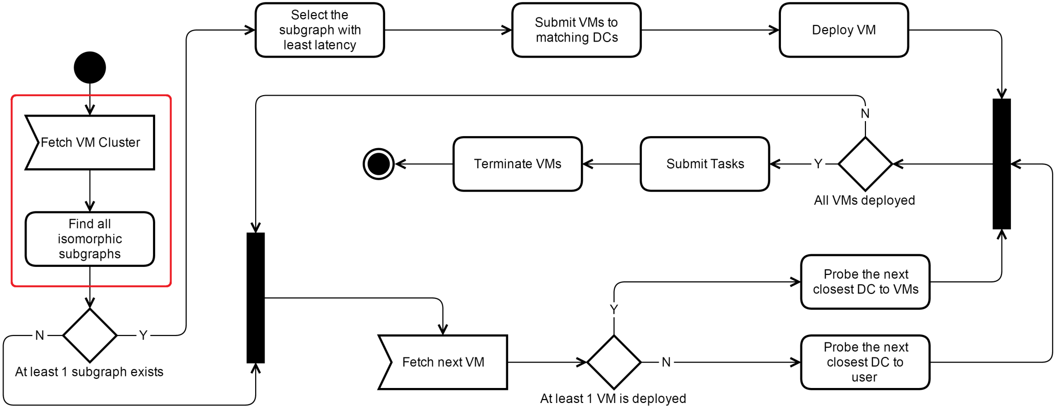
$$g = \left(D_{1,2} \mapsto \{C_{1,3}, C_{3,5}\}, D_{2,3} \mapsto \{C_{4,5}\}\right)$$

Outline

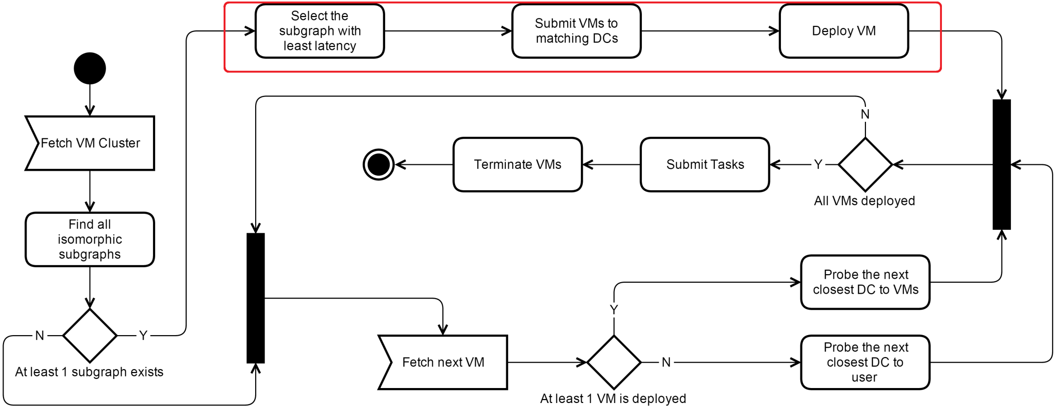
Topology based Mapping (TBM) Algorithm

- Map VM clusters to the subgraphs of the federation topology that are **isomorphic** to their topology.
  - Mapping function *f* is injective and $\forall (e \in E_C)\ \exists (E' \subseteq E_F) \mid e \mapsto E' \wedge |E'| = 1$
- In case of multiple alternatives, choose by average latency to the user.
- Fall back to a greedy heuristic in case of failure.
  - Instead, map to a homeomorphic subgraph where $|E'| \geq 1$

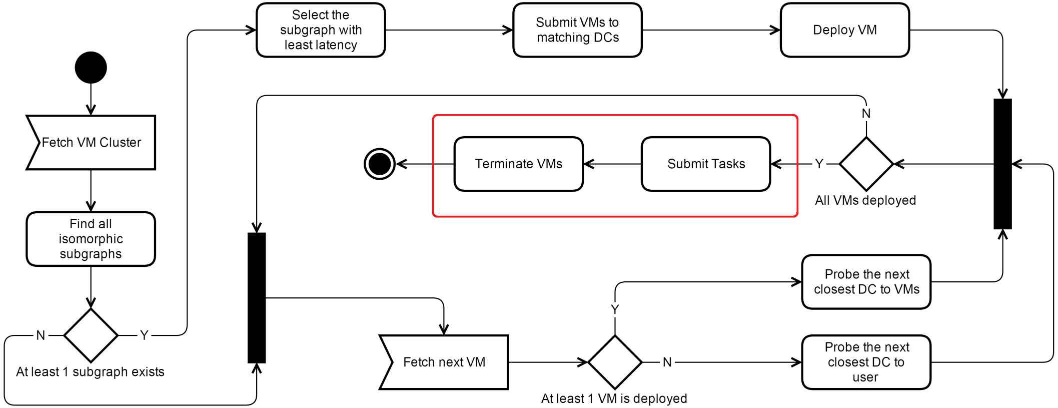
Topology based Mapping (TBM) Algorithm *(continued)*



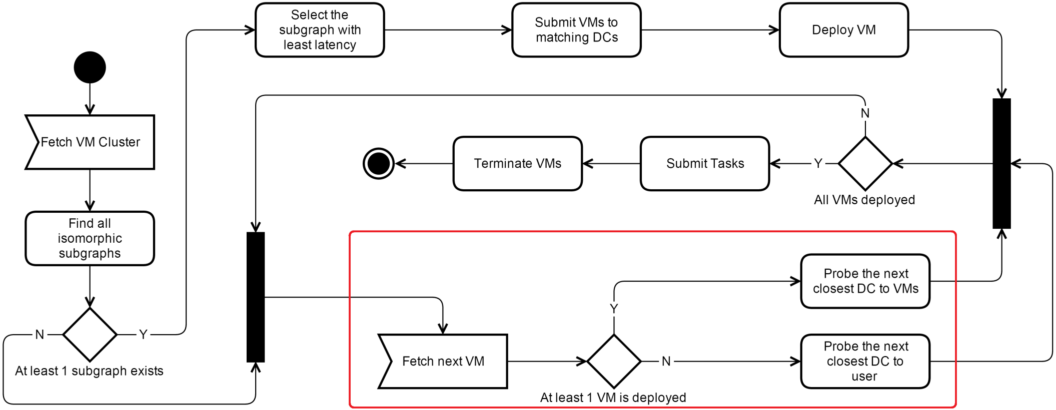
Topology based Mapping (TBM) Algorithm *(continued)*



Topology based Mapping (TBM) Algorithm *(continued)*



Topology based Mapping (TBM) Algorithm *(continued)*

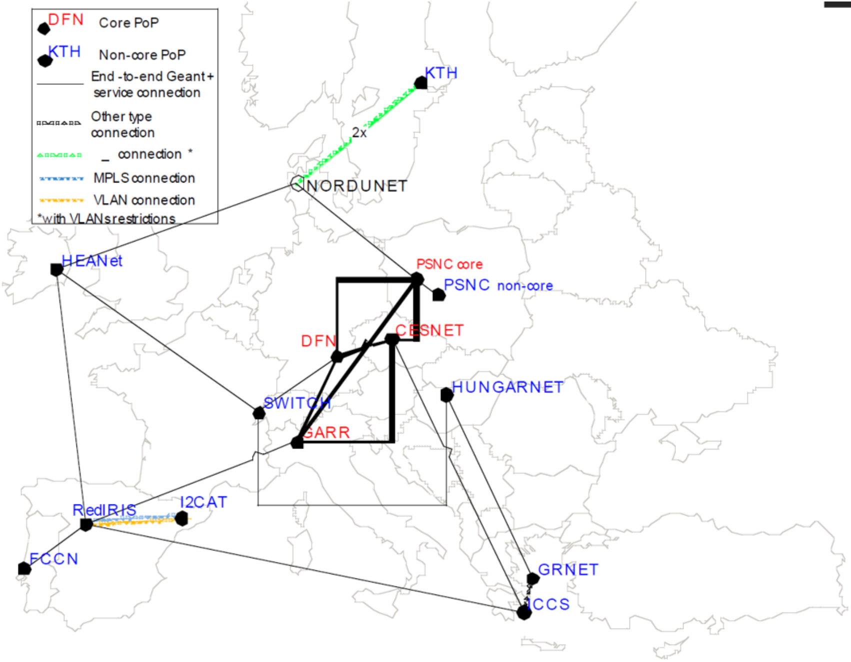


Outline

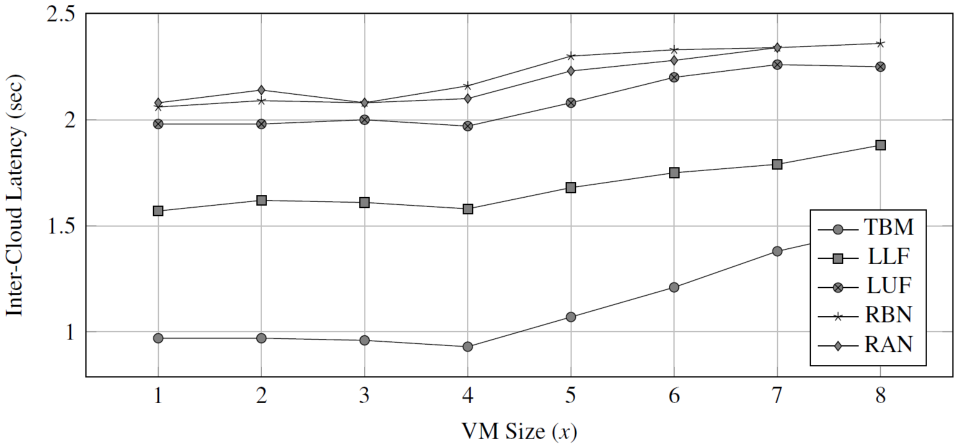
İTÜ

Experimental Setup

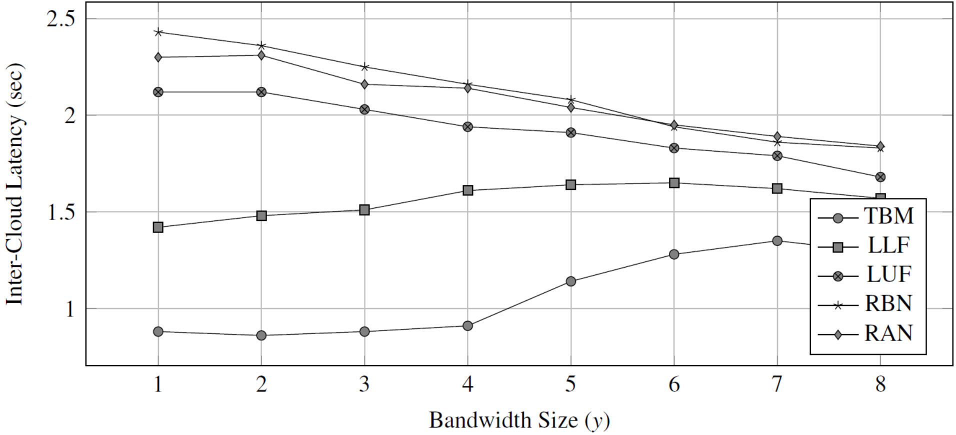
- Simulated on the RalloCloud framework which is based on CloudSim.
- Federation topology is taken from the FEDERICA project and contains 14 clouds across Europe.
- Virtual machine clusters are randomly generated based on population density.
- Simulation period is 50 hours.


DFN Core PoP
KTH Non-core PoP
End-to-end Geant+ service connection
Other type connection
_ connection *
MPLS connection
VLAN connection
*with VLANs restrictions
KTH
2x
NORDUNET
HEANet
PSNC core
PSNC non-core
DFN
CESNET
HUNGARNET
SWITCH
GARR
I2CAT
RedIRIS
FCCN
GRNET
ICCS

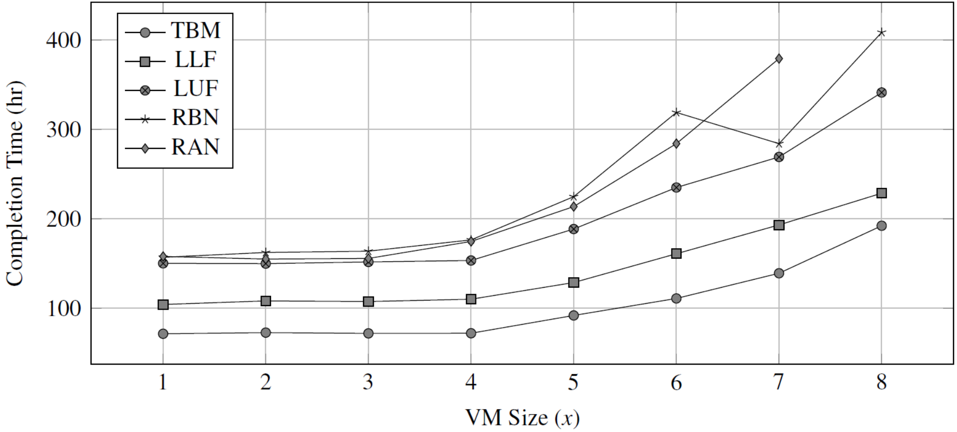
Results (1/4)


Inter-Cloud Latency (sec)
2.5
2
1.5
1
1
2
3
4
5
6
7
8
VM Size (x)
TBM
LLF
LUF
RBN
RAN

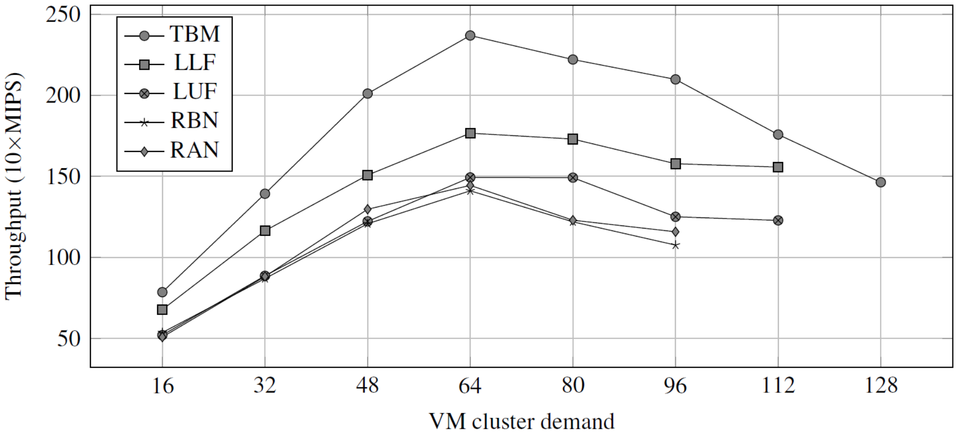
Results (2/4)


2.5
2
1.5
1
Inter-Cloud Latency (sec)
1
2
3
4
5
6
7
8
Bandwidth Size (y)
TBM
LLF
LUF
RBN
RAN

Results (3/4)


TBM
LLF
LUF
RBN
RAN
400
300
200
100
Completion Time (hr)
1
2
3
4
5
6
7
8
VM Size (x)

Results (4/4)


TBM
LLF
LUF
RBN
RAN
Throughput (10×MIPS)
250
200
150
100
50
16
32
48
64
80
96
112
128
VM cluster demand

Outline

Sport Event


Reed Messe Wien
WU (Wirtschaftsuniversität Wien)
Messe Wien
Danube
Trabrennpark Krieau
Hilton Vienna Danube Waterfront
U-Bahn Stadion
INTERSPORT Winninger
Ernst Happel Stadium

Sport Event


Reed Messe Wien
Messe Wien
Trabrennpark Krieau
Danube
Hilton Vienna Danube Waterfront
U-Bahn Stadion
Ernst Happel Stadium

Sport Event


Reed Messe Wien
Messe Wien
Trabrennpark Krieau
Danube
Hilton Vienna
Danube Waterfront
U-Bahn Stadion
Ernst Happel Stadium

Traffic

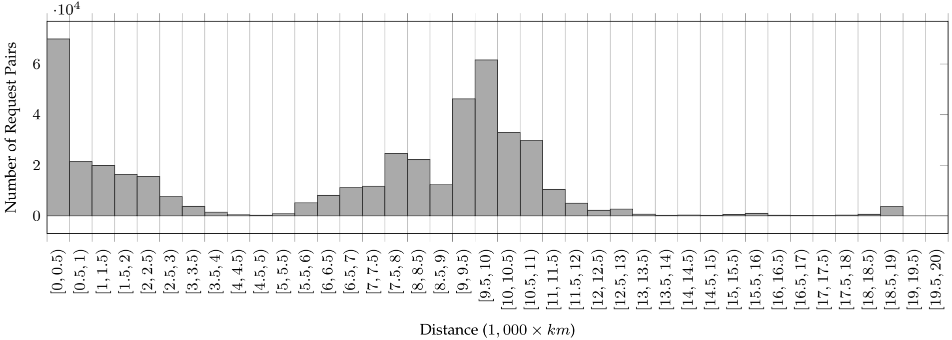
Traffic

Motivating Examples

- Edge Computing provides low-latency access to computing resources for mobile code offloading.
- However, many services need to access data that is stored centrally due to:
  - Limited storage capacity of the edge entities
  - Economic constraints
  - Availability for offline analysis
  - Simpler maintenance and concurrency control
- High latency to central storage harms QoS.
- How to decide the number and location of data replicas so that the **data access latency** is decreased in a **cost** efficient way?

Locality of Reference

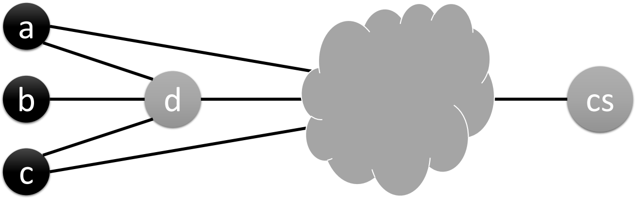
- The patterns that the user data accesses exhibit: Temporal Locality, Spatial Locality, and **Geographical Locality**



The CAIDA Anonymized Internet Traces 2015 Dataset (2.3 Billion IPv4 packets)

Outline

Edge Replica Placement



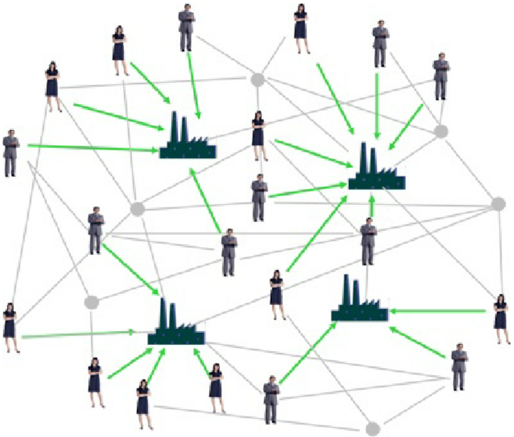
1. Which data objects to replicate?
2. When to create/destroy a replica?
3. How many replicas for each object?
4. Where to store each replica?
5. How to redirect requests to the closest replica?

- In order to minimize **average replica-to-client distance** in a **bandwidth- and cost-effective** way.

Facility Location Problem

$$minimize : \sum_j f_j \cdot Y_j + \sum_i \sum_j h_i \cdot d_{ij} \cdot X_{ij}$$

- $f_j = unit\_price_j \cdot replica\_size \cdot epoch$
- $h_i = num\_requests_i \cdot replica\_size$
- $d_{ij} = latency_{ij} \cdot \lambda$



Requirements

- Centralized solutions are not feasible due to the large number of entities.
  - The solution must be distributed with minimal input and communication.
- Edge users continuously enter and leave the network topology through connections with nonuniform latencies.
  - The solution must be dynamic and online.
- Edge entities should be aware of the closest replica when they need a certain data object.
  - A Replica Discovery technique is necessary.

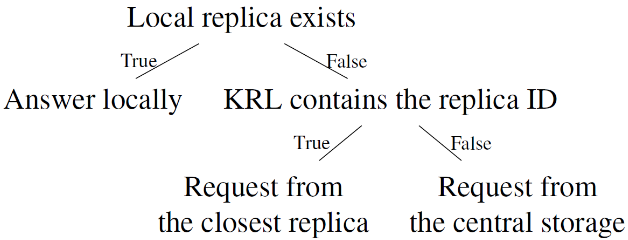
Outline

Decentralized Replica Placement (D-ReP) Algorithm

- Storage nodes that host replicas act as local optimizers.
- They evaluate experienced demand, storage cost as well as expected latency improvement to carry out either:
  - Duplication $num\_requests_{knh} \cdot latency_{nh} \cdot \lambda > unit\_price_n \cdot epoch$
  - Migration $(num\_reqs_{knh} - \sum_{\substack{i \in N \\ i \neq n}} num\_reqs_{kih}) \cdot latency_{nh} \cdot \lambda > (unit\_price_n - unit\_price_h) \cdot epoch$
  - Removal $\sum_{i \in N} num\_requests_{kih} < original\_num\_requests_h \cdot \alpha$
- The replicas are incrementally pushed from the central storage to the edge.
- λ allows user to control the trade-off between cost- and latency-optimization.

Replica Discovery

- Only the most relevant nodes are notified of the replica creations or removals.
  - Temporal locality: requesters of the source during the n most recent epochs
  - Geographical locality: m-hop sphere of the destination
- Each active node keeps a Known Replica Locations (KRL) table.

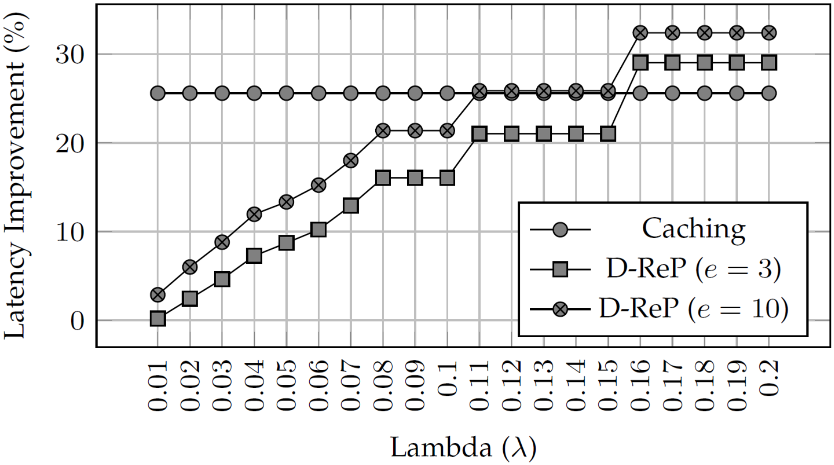


Outline

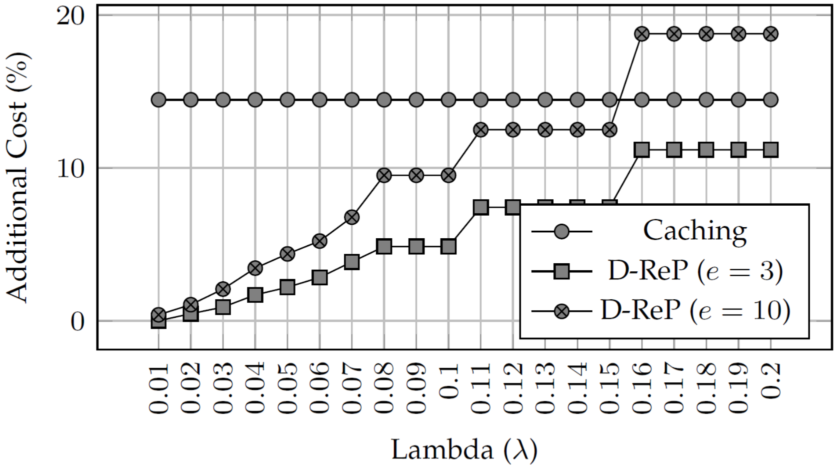
Experimental Setup

- Simulated on the RalloCloud framework which is based on CloudSim.
- The CAIDA Anonymized Internet Traces 2015 Dataset: IPv4 packets data from February 19, 2015 between 13:00-14:00 (UTC) which contains more than 2.3 Billion records
- GeoLite2 IP geolocation database
- Synthetic workloads based on uniform, exponential, normal, Chi-squared, and Pareto distributions of request locations
- Barabási–Albert scale-free network generation model: 1000 nodes, 2994 edges, and a heavy-tailed distribution of bandwidth in [10, 1024] mbps
- Amazon Web Services S3 prices

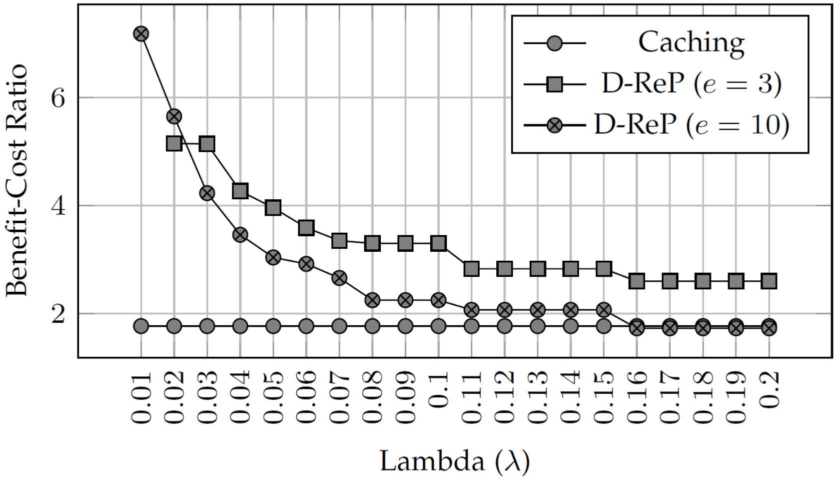
Results (1/4)

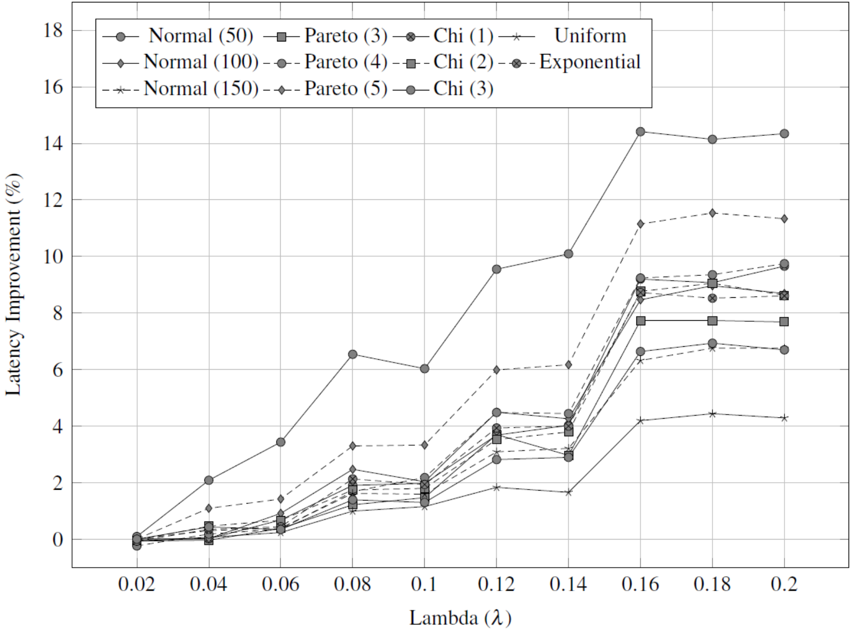

Latency Improvement (%)
30
20
10
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
0.11
0.12
0.13
0.14
0.15
0.16
0.17
0.18
0.19
0.2
Lambda (λ)
Caching
D-ReP (e = 3)
D-ReP (e = 10)

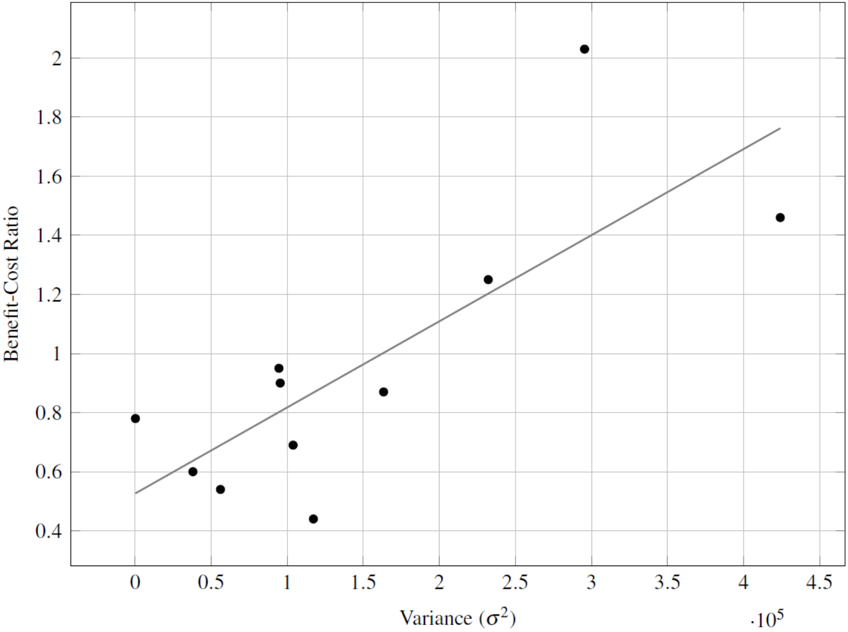
Results (2/4)


20
10
0
Additional Cost (%)
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
0.09
0.1
0.11
0.12
0.13
0.14
0.15
0.16
0.17
0.18
0.19
0.2
Lambda (λ)
Caching
D-ReP (e = 3)
D-ReP (e = 10)

Results (3/4)


Caching
D-ReP ($e = 3$)
D-ReP ($e = 10$)
Benefit-Cost Ratio
6
4
2
0.01 0.02 0.03 0.04 0.05 0.06 0.07 0.08 0.09 0.1 0.11 0.12 0.13 0.14 0.15 0.16 0.17 0.18 0.19 0.2
Lambda (λ)


Normal (50)
Pareto (3)
Chi (1)
Uniform
Normal (100)
Pareto (4)
Chi (2)
Exponential
Normal (150)
Pareto (5)
Chi (3)
Latency Improvement (%)
Lambda (λ)



Benefit-Cost Ratio
2
1.8
1.6
1.4
1.2
1
0.8
0.6
0.4
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
Variance (σ^2)
$\cdot 10^5$

Outline

Contribution

- Topology Mapping Algorithm[1 4 5]
  - First attempt to employ subgraph isomorphism to find a injective match between virtual and physical cloud/grid topologies
  - First to explicitly evaluate network latency for the VMNE
- Minimum Span Heuristic[3 5]
  - Novel heuristic algorithm to defer MIP
- Decentralized Replica Placement Algorithm[2 5]
  - First completely decentralized, partial-knowledge replica placement algorithm
- KRL based Discovery Technique[2]
  - Novel replica discovery technique with low overhead
- RalloCloud: A simulation environment for Inter-Cloud resource mapping[1]
  - First simulator for inter-cloud systems

Publications

1. Aral, A., and Ovatman, T. 2016. Network-Aware Embedding of Virtual Machine Clusters onto Federated Cloud Infrastructure. ***The Journal of Systems and Software***, vol. 120, pp. 89-104. DOI: 10.1016/j.jss.2016.07.007
2. Aral, A., and Ovatman, T. 2017?. A Decentralized Replica Placement Algorithm for Edge Computing. Under review in ***The IEEE Transactions on Parallel and Distributed Systems***.

---

3. Aral, A., and Ovatman, T. 2014. Improving Resource Utilization in Cloud Environments using Application Placement Heuristics. In *4th Int'l. Conf. on Cloud Comp. and Services Science (**CLOSER 2014**)*, pp. 527-534.
4. Aral, A., and Ovatman, T. 2015. Subgraph Matching for Resource Allocation in the Federated Cloud Environment. In *IEEE 8th International Conference on Cloud Computing (**IEEE CLOUD 2015**)*, pp. 1033-1036.
5. Aral, A. 2016. Network-Aware Resource Allocation in Distributed Clouds. *IEEE International Conference on Cloud Engineering (**IC2E 2016**)*, Doctoral Symposium.

Future Work

- Standardization of cloud APIs
- Real-time performance and cost guarantees
- Service self-awareness
- Load balancing and fairness between cloud providers
- Fault tolerance and availability

Thank you for your time.

Appendices

Virtual Network Embedding

	Cloud	Federation	Embedding	Entity	Simultaneous	NW-Aware
TBM	✓	✓	✓	VMs	✓	✓
[30]	✓	✓	✓	VMs		
[31]	✓	✓	✓	VMs	✓	✓
[32]	✓	✓	✓	Serv.	✓	
[33]	✓	✓	✓	Tasks		
[34]	✓	✓	✓	VMs	✓	✓
[35]	✓	✓	✓	VNs		✓
[22]	✓	✓	✓	VMs	✓	
[36]	✓	✓	✓	VMs		✓
[37]	✓	✓	✓	VNs		✓
[25]	✓	✓	✓	VMs	✓	✓
[38]	✓	✓	✓	VNs	✓	✓

Virtual Network Embedding

	Cloud	Federation	Embedding	Entity	Simultaneous	NW-Aware
TBM	✓	✓	✓	VMs	✓	✓
[39]			✓	VNs		
[40]			✓	VNs	✓	✓
[41]			✓	VMs	✓	✓
[42]			✓	VNs		✓
[43]		✓	✓	VNs	✓	✓
[44]		✓	✓	VMs	✓	✓
[45]		✓	✓	VNs	✓	✓
[46]	✓		✓	VMs	✓	✓
[47]	✓		✓	VNs		✓
[48]	✓		✓	Tasks	✓	✓
[49]	✓		✓	VNs		✓

Virtual Network Embedding

	Cloud	Federation	Embedding	Entity	Simultaneous	NW-Aware
TBM	✓	✓	✓	VMs	✓	✓
[50]	✓		✓	VNs		✓
[51]	✓	✓		VMs	✓	✓
[52]	✓	✓		Jobs		
[53]	✓	✓		WEs	✓	✓

Virtual Network Embedding

[25] *Tordsson, J., Montero, R.S., Moreno-Vozmediano, R. and Llorente, I.M. (2012). Cloud brokering mechanisms for optimized placement of virtual machines across multiple providers, Future Generation Computer Systems, 28(2), 358–367.*

Exact solution with IP, cloud brokerage and uniform interface

[35] *Leivadeas, A., Papagianni, C. and Papavassiliou, S. (2013). Efficient resource mapping framework over networked clouds via iterated local search-based request partitioning, IEEE Transactions on Parallel and Distributed Systems, 24(6), 1077–1086.*

Graph partitioning, IP based embedding, no latency consideration

[38] *Xin, Y., Baldine, I., Mandal, A., Heermann, C., Chase, J. and Yumerefendi, A. (2011). Embedding virtual topologies in networked clouds, Proceedings of the 6th International Conference on Future Internet Technologies, ACM, pp.26–29.*

Connected components of the VM cluster topology are merged, isomorphic subgraph to the resulting graph is sought

Replica Placement (Centralized)

	P	D	Environment	Topology	Objectives
[68]			Web	Tree	Proximity
[69]			CDN	Unrestricted	Proximity
[70]			N/A	Unrestricted	Proximity
[71]			Data Grid	Tree	Proximity, Cost, Load Balance
[72]			N/A	Tree	Proximity
[73]			N/A	Unrestricted	Proximity
[65]			Cloud	Unrestricted	Proximity
[74]			Cloud	Unrestricted	Bandwidth, Load Balance
[67]			Cloud	Unrestricted	Proximity, Cost
[75]		✓	N/A	Unrestricted	Availability
[76]		✓	Data Grid	Multi-Tier	Proximity, Cost, Bandwidth
[77]		✓	CDN	Unrestricted	Proximity, Cost

Replica Placement (Centralized)

	P	D	Environment	Topology	Objectives
[63]		✓	Cloud	Unrestricted	Prox., Bandwidth, Load Balance
[78]		✓	Data Grid	Multi-Tier	Proximity
[64]		✓	Data Grid	Unrestricted	Prox., Bandwidth, Availability
[66]		✓	Cloud-CDN	Unrestricted	Proximity, Cost
[79]		✓	Data Grid	Tree	Prox., Bandwidth, Availability
[80]		✓	Data Grid	Multi-Tier	Proximity, Bandwidth
[81]		✓	Cloud	Tree	Proximity, Availability
[82]		✓	Cloud	Unrestricted	Bandwidth, Load Balance
[83]		✓	Cloud-CDN	Unrestricted	Cost, Availability
[84]		✓	Cloud (Mobile)	Unrestricted	Proximity, Availability

Replica Placement (Decentralized)

	P	D	Environment	Topology	Objectives
[89]			Cloud-CDN	Unrestricted	Proximity
[90]		✓	Cloud	Complete	Prox., Cost, Bandwidth, Avail.
[91]		✓	Data Grid	Unrestricted	Prox., Cost, Bandwidth, Avail.
[92]		✓	P2P	Unrestricted	Cost, Bandwidth, Availability
[93]		✓	Web	Unrestricted	Proximity, Bandwidth
[94]	✓		N/A	Multi-Tier	Proximity, Bandwidth
[88]	✓	✓	Web	Unrestricted	Prox., Cost, Load B., Bandwidth
[95]	✓	✓	P2P	Unrestricted	Proximity, Cost
[87]	✓	✓	Web	Unrestricted	Proximity, Cost
[96]	✓	✓	Web	Unrestricted	Proximity, Cost
D-ReP	✓	✓	Cloud	Unrestricted	Proximity, Cost, Bandwidth

Replica Placement

[90] *Bonvin, N., Papaioannou, T.G. and Aberer, K. (2010). A self-organized, fault-tolerant and scalable replication scheme for cloud storage, Proceedings of the 1st ACM symposium on Cloud computing, pp. 205–216.*
Game theoretical, replicate/migrate autonomously, each node is aware of other replicas, prices, demand, bandwidth, . . .

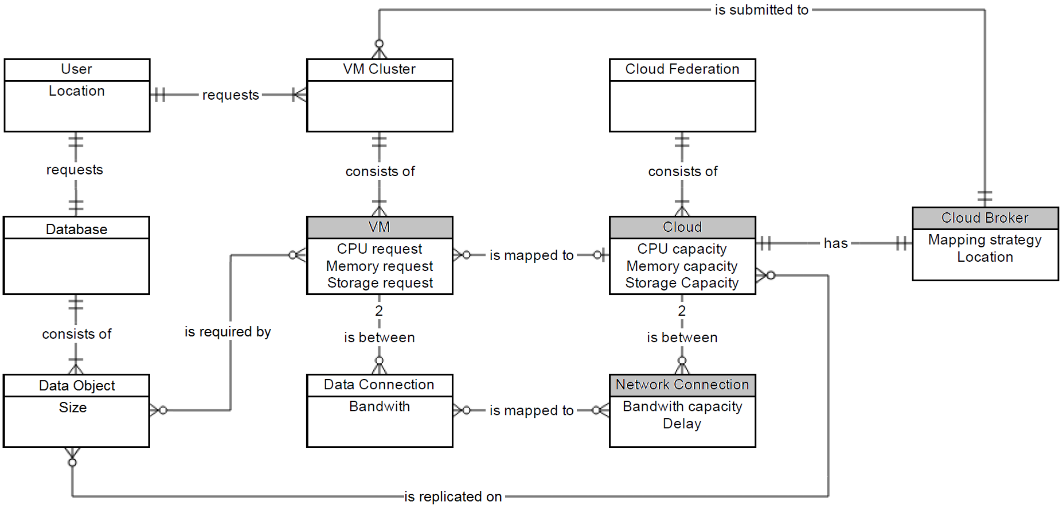
[95] *Shen, H. (2010). An efficient and adaptive decentralized file replication algorithm in P2P file sharing systems, IEEE Transactions on Parallel and Distributed Systems, 21(6), 827–840.*
Replicas on data access path intersections (traffic hubs), traffic analysis

[87] *Pantazopoulos, P., Karaliopoulos, M. and Stavrakakis, I. (2014). Distributed placement of autonomic internet services, IEEE Transactions on Parallel and Distributed Systems, 25(7), 1702–1712.*

[96] *Smaragdakis, G., Laoutaris, N., Oikonomou, K., Stavrakakis, I. and Bestavros, A. (2014). Distributed server migration for scalable Internet service deployment, IEEE/ACM Trans. on NW, 22(3), 917–930.*
Solve FLP in an r-ball, centrality, demand and FLP decisions are broadcasted

Appendices

Entity Relationship Diagram



Network Modeling

- Requested **bandwidth** between two VMs is allocated from all edges on the shortest path between the clouds to which two VMs are deployed.
- Three types of **latency** are considered:

$$Deployment\ Latency = M + \sum_{i \in P_1} L_i + \frac{S}{B}$$

$$Communication\ Latency = \sum_{i \in P_2} L_i + \frac{D}{B}$$

$$Data\ Access\ Latency = \sum_{i \in P_3} L_i + \frac{D}{B}$$

Cost Modeling

- Static Pricing and Trough Filling strategies

$$\begin{aligned} \textit{Total Service Cost} = & \sum_{i \in V_C} \sum_{j \in A_C^V} \textit{resource\_size}_{ij} \cdot \textit{unit\_price}_{ij} \cdot \textit{duration}_{ij} \\ & + \sum_{i \in E_C} \textit{resource\_size}_i \cdot \textit{unit\_price}_i \cdot \textit{duration}_i \\ & + \sum_{i} \textit{replica\_size}_i \cdot \textit{unit\_price}_i \cdot \textit{duration}_i \end{aligned}$$

Performance Criteria

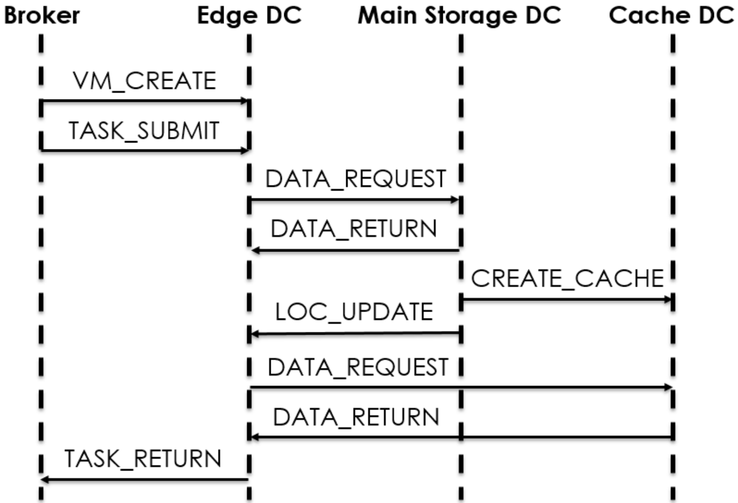
- Criteria
  - Temporal
    - Latency
      - **Cloud-to-User Latency**
      - Hop Count
      - **Inter-Cloud Latency**
    - Duration
      - **Completion Time**
      - **Execution Time**
      - Makespan
      - SLA Violations
  - Volumetrical
    - **Throughput**
    - Utilization Rate
  - Economical
    - **Benefit-Cost Ratio**
    - **Resource Cost**
    - Revenue
  - Distributional
    - **Distribution Rate**
    - Load Balance
  - Other
    - **Acceptance/Rejection Rate**
    - Fairness

Appendices

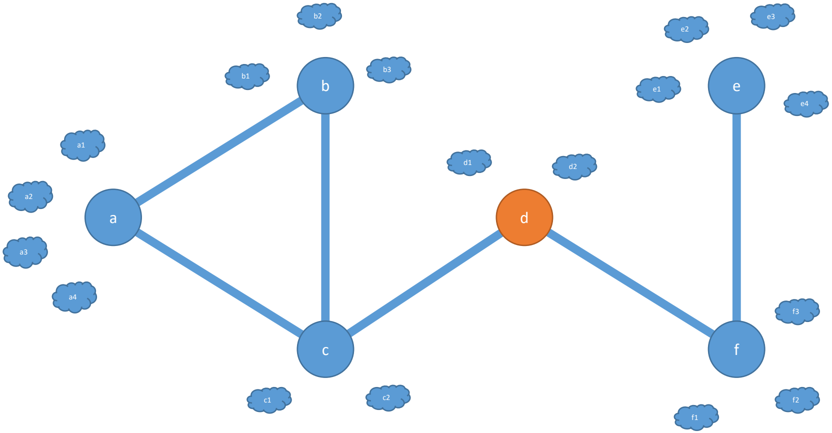
TBM

```
foreach cluster request G_C in queue do
    subgraphs[ ] ← SearchIsomorphicSubgraph(G_F, G_C)
    if size(subgraphs[ ])>0 then
        chosenSubgraph ← argmin_x(AvgLatency(subgraph[x], user))
        map each VM in G_C to the corresponding node in chosenSubgraph
    else
        foreach virtual machine VM in G_C do
            deployedVMs[ ] ← deployed(G_C)
            if size(deployed[ ])>0 then
                chosenNode ← argmin_x(AvgLatency(V_F[x], deployed[ ]))
            else
                chosenNode ← argmin_x(AvgLatency(V_F[x], user))
            map VM to chosenNode
    Try to deploy VMs at mapped nodes and allocate data connections
```

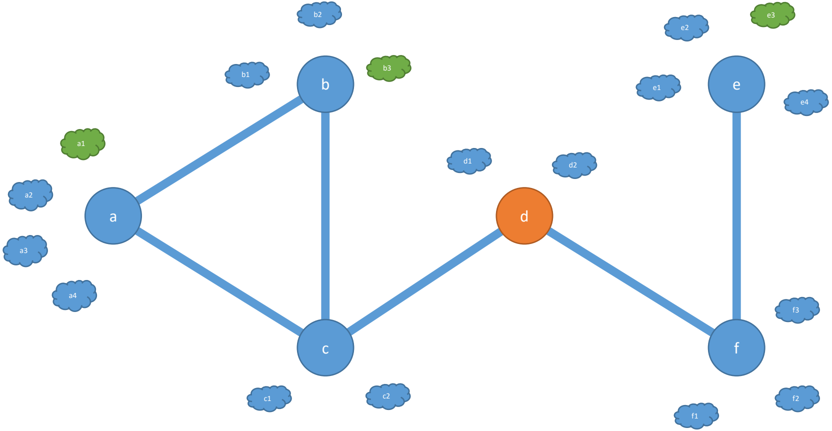
Replica Discovery



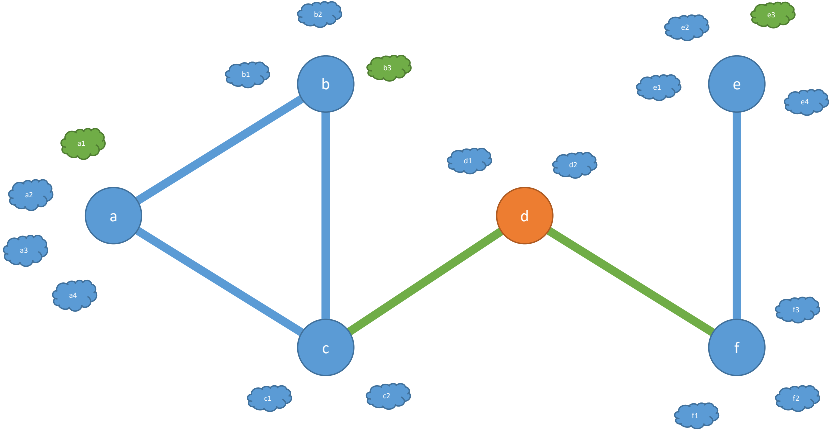
D-ReP



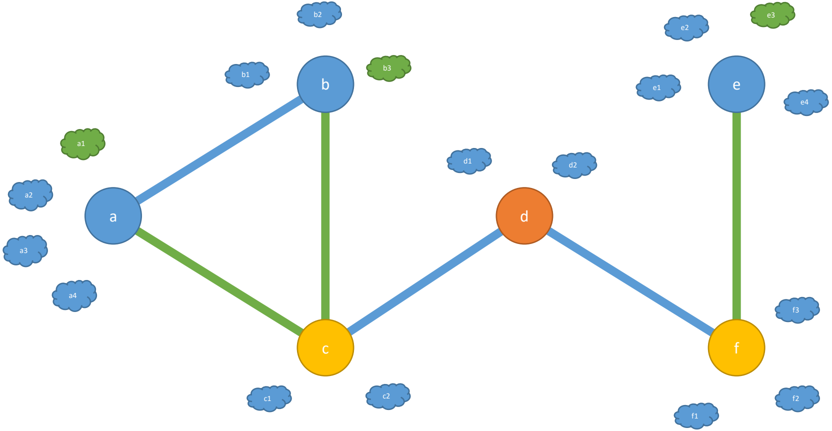
User demand locations



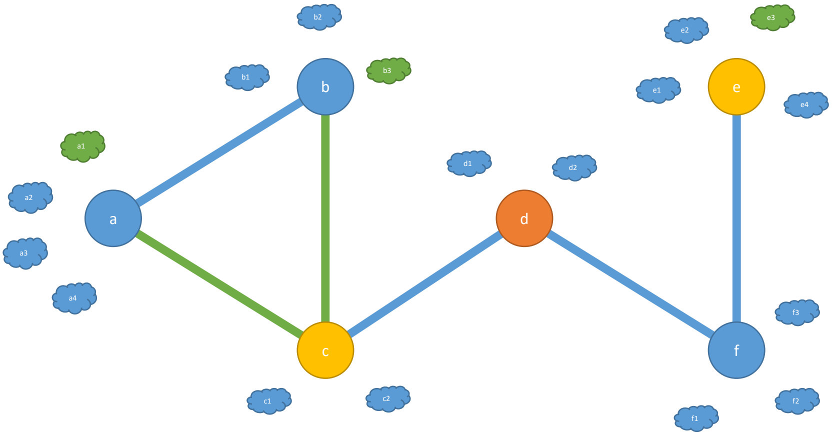
ITERATION 1d: User demand received from c and f



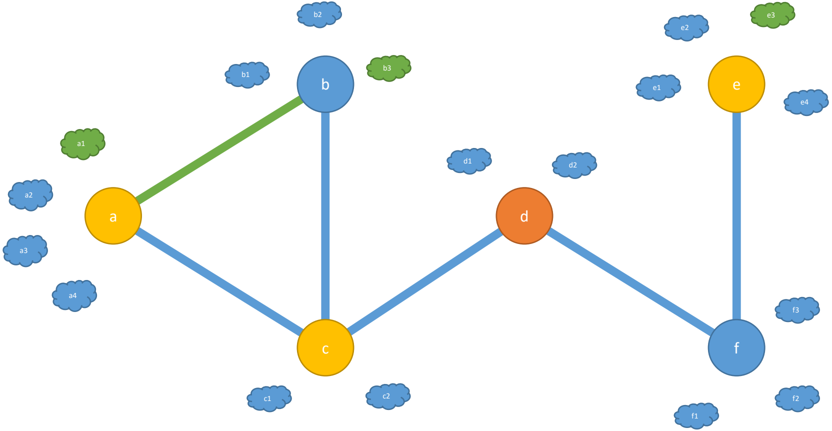
ITERATION 1d: Cache creation decision



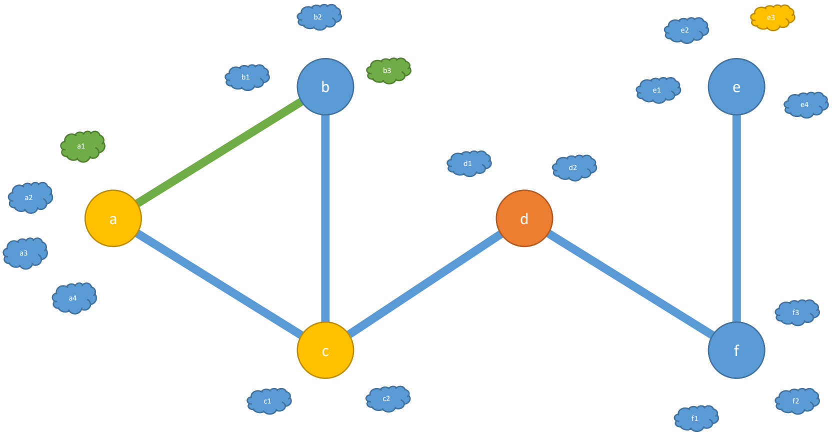
ITERATION 2f: Migration decision



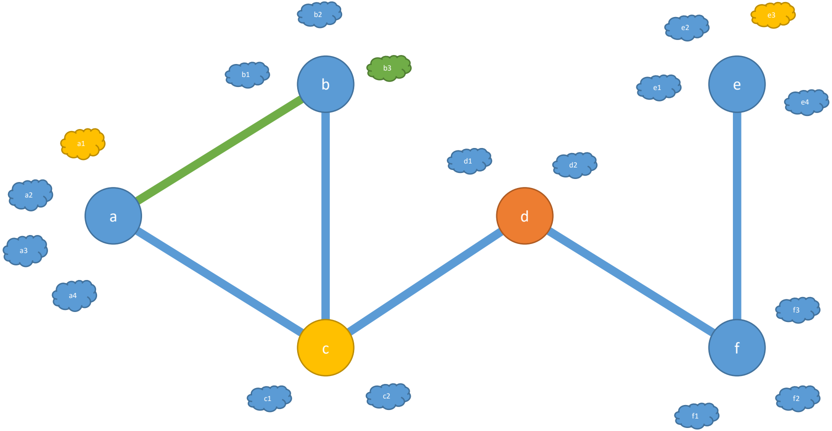
ITERATION 2c: Duplication decision



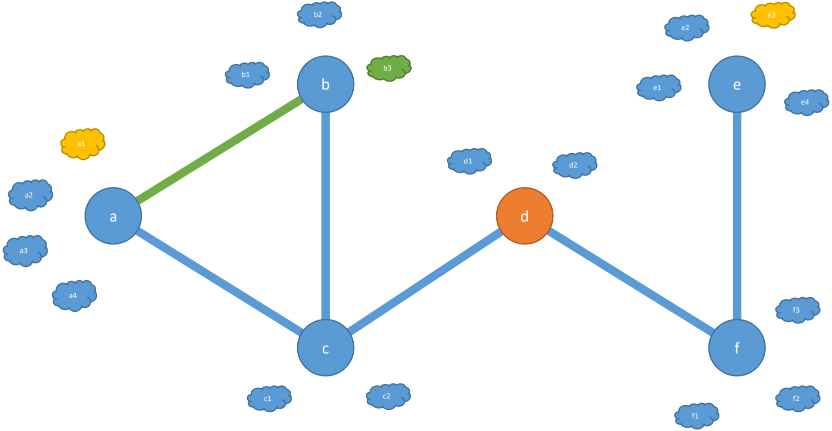
ITERATION 3e: Migration decision



ITERATION 3a: Migration decision



ITERATION 3c: Removal decision



Appendices

Virtual Machine Cluster Embedding

- Allocation of PMs to VMs in a single cloud data center.

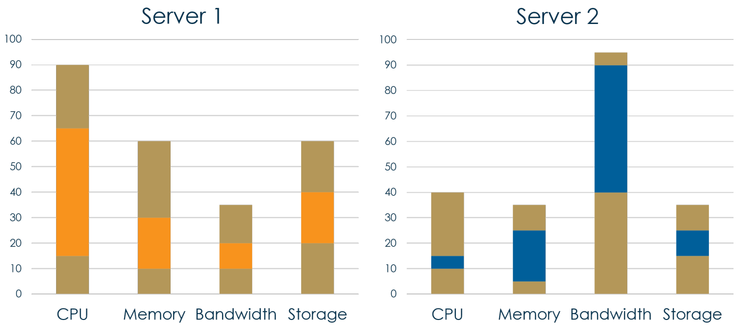
- Increase **resource utilization**
- Increase data center throughput
- Increase acceptance rate
- Increase revenue

$$\bigwedge_{i=0}^{\#PMs}\left(\sum_{j=0}^{\#VMs} A[i][j]=1\right)$$

$$\bigwedge_{i=0}^{\#PMs}\bigwedge_{j=0}^{\#Res}\left(\left(\sum_{k=0}^{\#VMs} A[i][k]\times RR[j][k]\right)\leq AR[i][j]\right)$$

Minimum Span Heuristic

- Map a VM to the PM with the maximum **resource utilization evenness.**
- $Span(p) = \max_{i \in [1,m]} (r_i) - \min_{i \in [1,m]} (r_i)$
- Fall back to a Mixed Integer Programming (MIP) solution in case of failure.



Pseudo Code

```
rejected ← false
while rejected = false do
    receive VM v
    assignable ← false
    foreach PM p do
        if p has enough capacity for v then
            assignable ← true
            assign v to p
            calculate unevenness of p
            remove v from p
    if assignable = true then
        assign v to the p with the minimum unevenness
    else
        run optimization algorithm
        if optimization succeeds then
            assign v and migrate others
        else rejected ← true
```

Other Heuristics

$$SD(v) = \sqrt{\sum_{i=1}^{m} (r_i - \bar{r})^2}$$

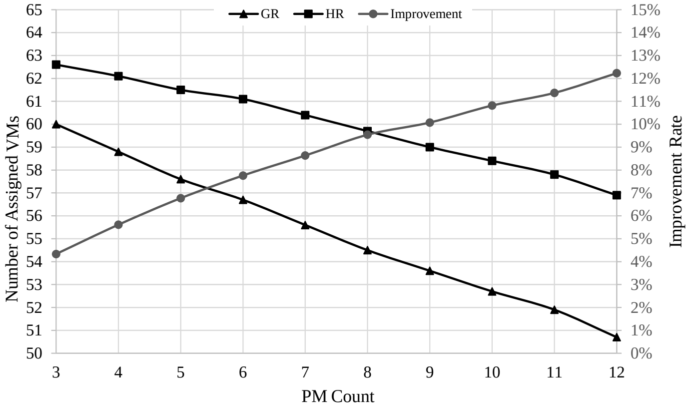
$$DM(v) = \sum_{i=1}^{n} \left(r_i - \min_{j \in [1,m]} (r_j) \right)$$

$$CD(v) = \frac{\sum_{i=1}^{m} \sum_{j=1}^{m} |r_i - r_j|}{2}$$

$$SK(v) = \sqrt{\sum_{i=1}^{m} \left(\frac{r_i}{\bar{r}} - 1 \right)^2}$$

ITÜ

Results (1/2)



Strategy	Avg. Migr. Count	Perfect Count
RR	8,4	26
SD_{min}	5,5	108
SP_{min}	5,6	100
CD_{min}	5,8	86
SK_{min}	7,2	88

- 10.8% perfect placement
- 12.1% more VMs
- 34.5% less migrations

Results (2/2)

VM Capacity	100	150	200	250	300	200	200	200	200	200
VM Count	**8**	**8**	**8**	**8**	**8**	**4**	**6**	**8**	**10**	**12**
RR	42,0	67,2	92,7	118,4	144,3	46,0	69,3	92,7	116,2	139,7
SK_{min}	45,8	72,3	98,8	125,3	151,9	48,1	73,4	98,8	124,3	149,9
SK_{dec}	45,5	72,0	98,6	125,2	151,9	48,1	73,2	98,6	124,1	149,7
SP_{min}	**46,2**	**73,2**	**100,2**	**127,2**	**154,3**	**48,7**	**74,4**	**100,2**	**126,2**	**152,3**
SP_{dec}	46,0	72,6	99,3	126,0	153,2	48,4	73,8	99,3	124,9	150,7
SD_{min}	**46,2**	**73,2**	**100,2**	**127,3**	**154,3**	**48,7**	**74,4**	**100,2**	**126,2**	**152,3**
SD_{dec}	45,8	72,3	98,9	125,5	152,1	48,2	73,4	98,9	124,4	150,1
CD_{min}	**46,2**	**73,2**	**100,2**	**127,3**	**154,3**	**48,8**	**74,4**	**100,2**	**126,2**	**152,3**
CD_{dec}	45,9	72,5	99,1	125,9	152,7	48,3	73,7	99,1	124,8	150,5
DM_{min}	45,7	72,6	99,6	126,5	153,6	48,3	73,8	99,6	125,4	151,5
DM_{dec}	45,8	72,5	99,2	126,2	153,1	48,3	73,6	99,2	125,0	150,8
Expected	53,3	80,0	106,7	133,3	160,0	53,3	80,0	106,7	133,3	160,0
Improvement	**10,0%**	**8,9%**	**8,1%**	**7,5%**	**6,9%**	**6,1%**	**7,4%**	**8,1%**	**8,6%**	**9,0%**

Appendices

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

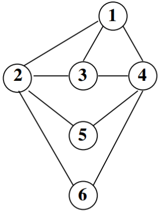
Output: failure (if an inconsistency is detected) or success

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 $adj(u)$.

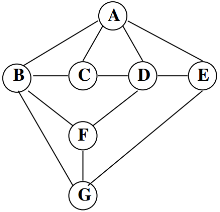
while $S \neq \emptyset$ **do**
- Remove a couple of pattern/target nodes (u, v) from S
- **if** *there does not exist a matching of* $G_{(u,v)}$ *that covers* $adj(u)$ **then**
  - Remove v from D_u
  - **if** $D_u = \emptyset$ **then return** failure
  - $S \leftarrow S \cup \{(u', v') \mid u' \in adj(u), v' \in adj(v) \cap D_{u'}\}$

return success

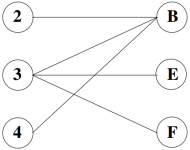
LAD Filtering



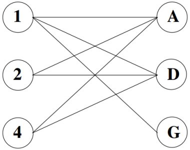
Pattern graph G_p



Target graph G_t



$G_{(1,G)}$



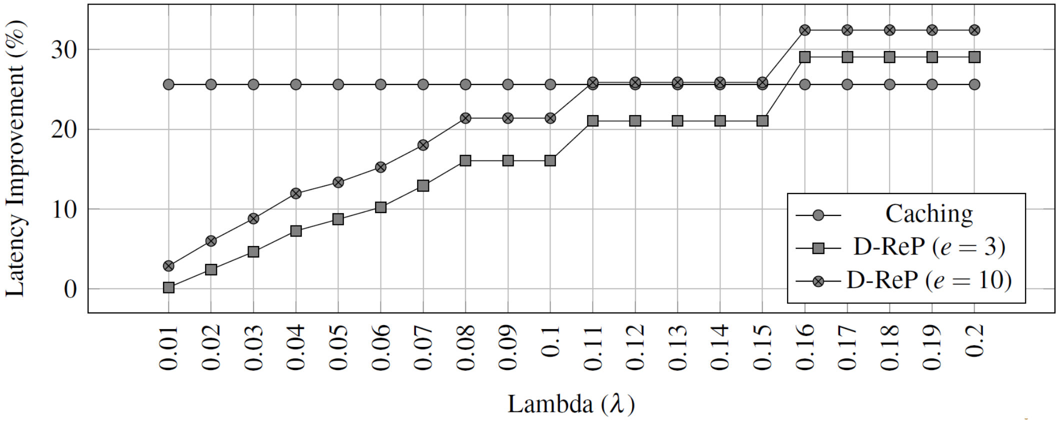
$G_{(3,E)}$

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

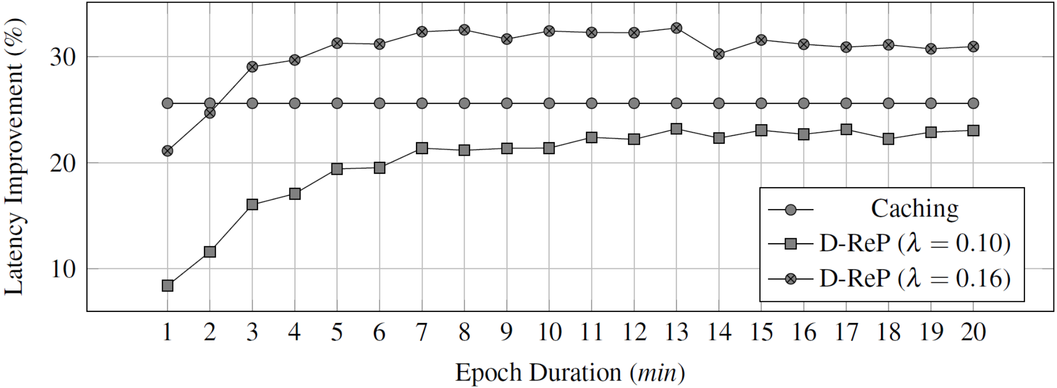
$$D_2 = D_4 = A, B, D$$

Appendices

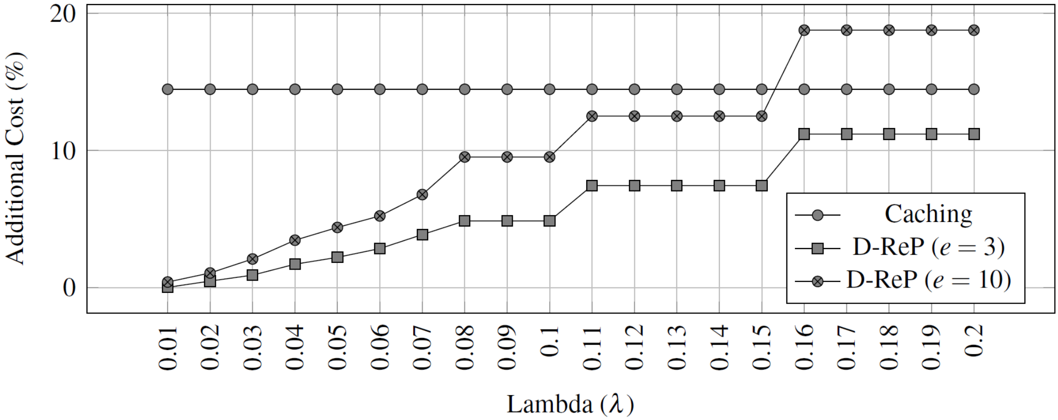
Complete Results



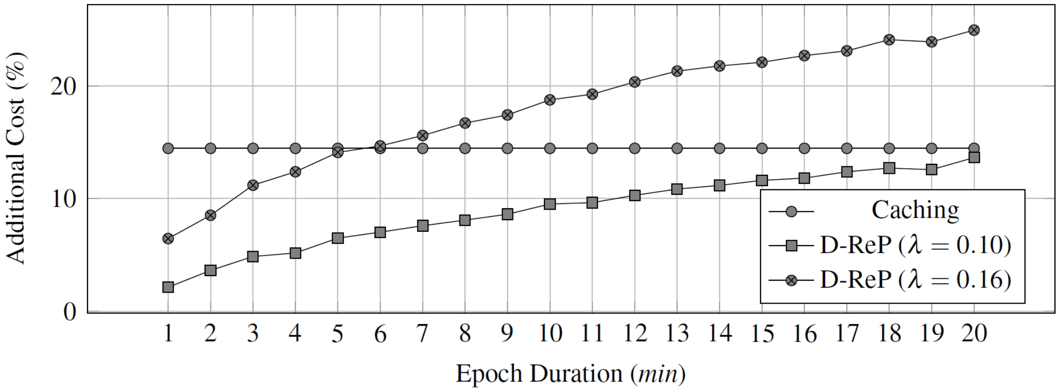
Complete Results



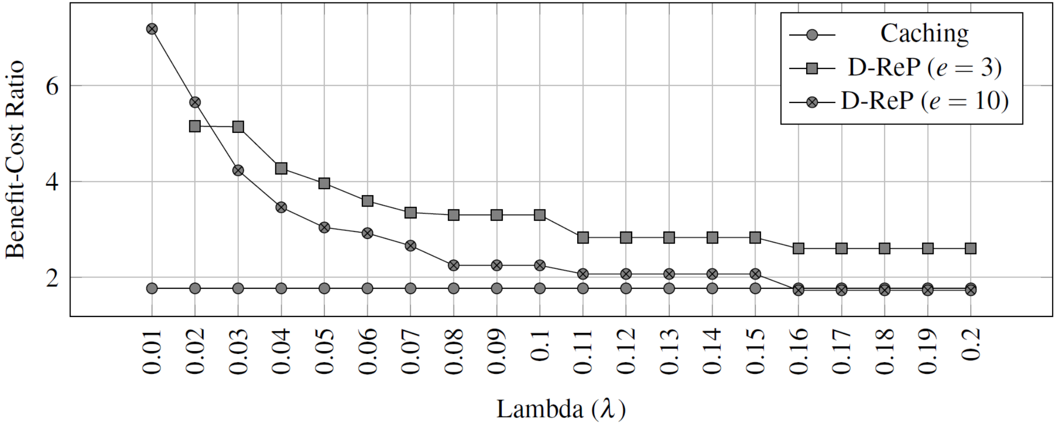
Complete Results



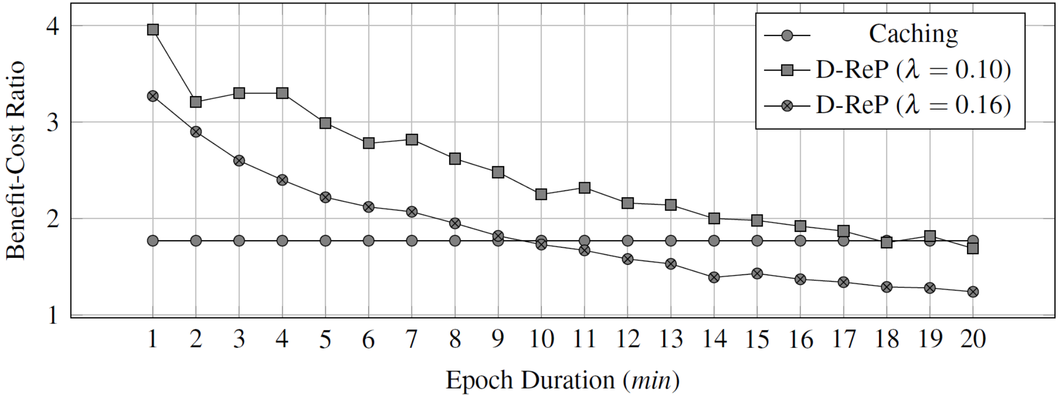
Complete Results



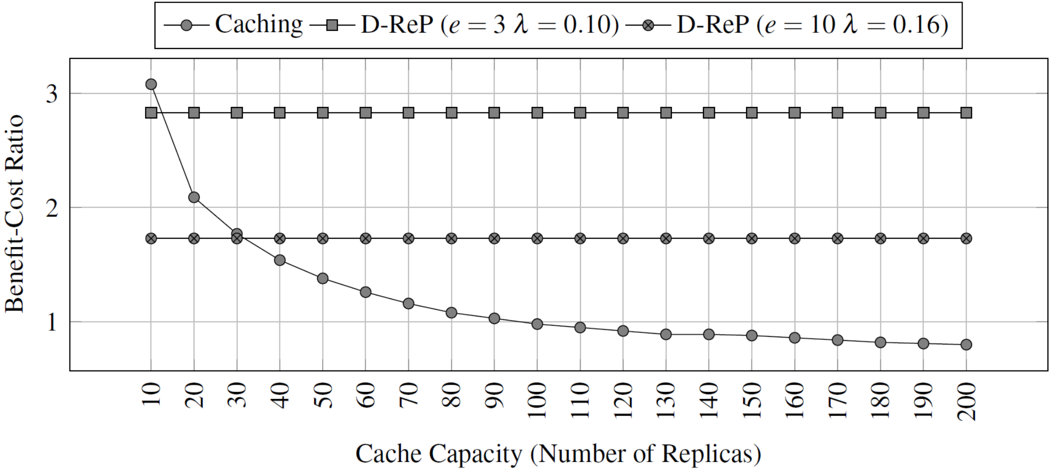
Complete Results



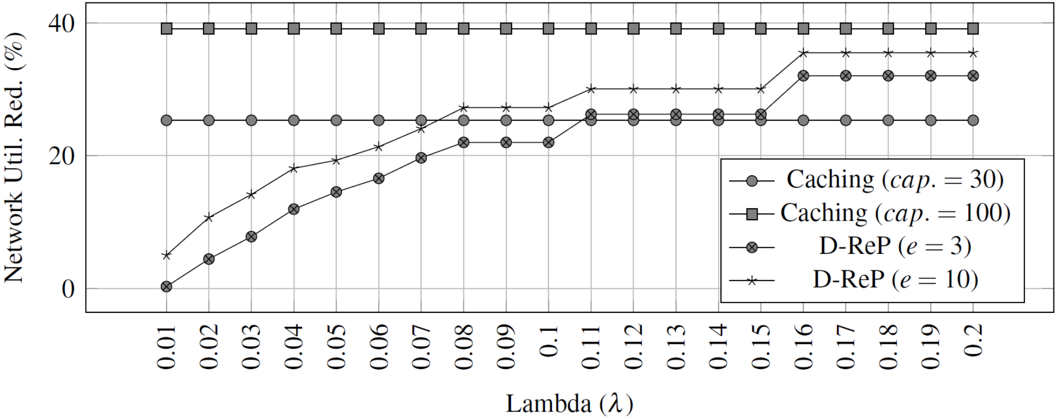
Complete Results



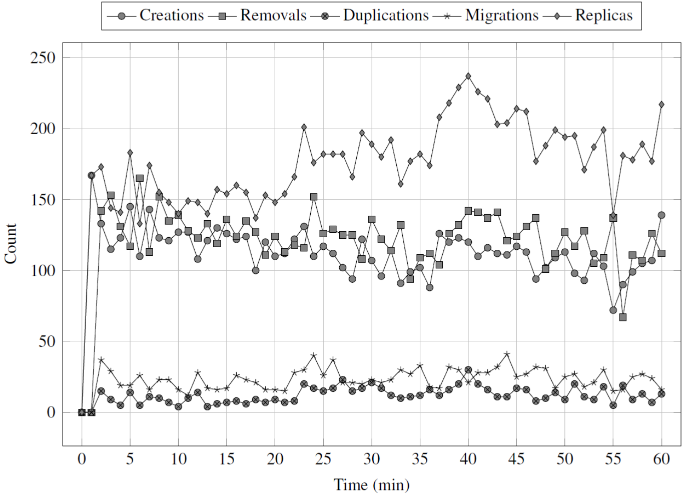
Complete Results



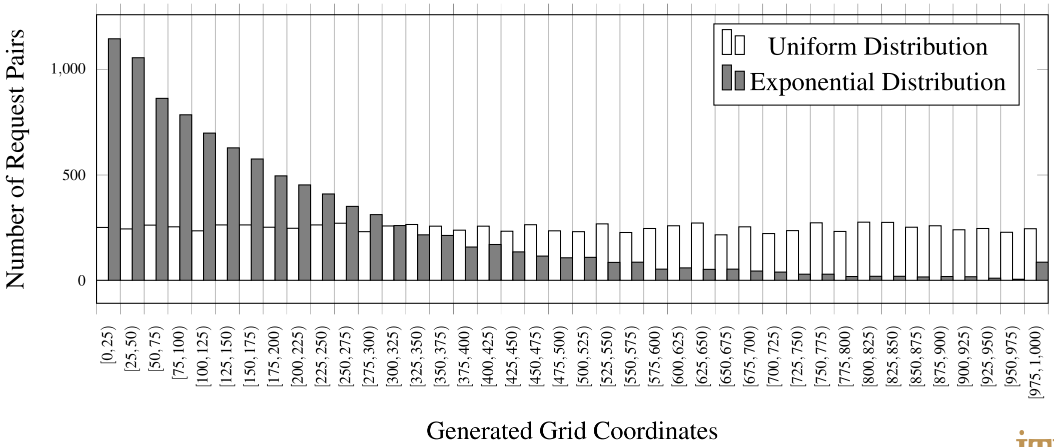
Complete Results



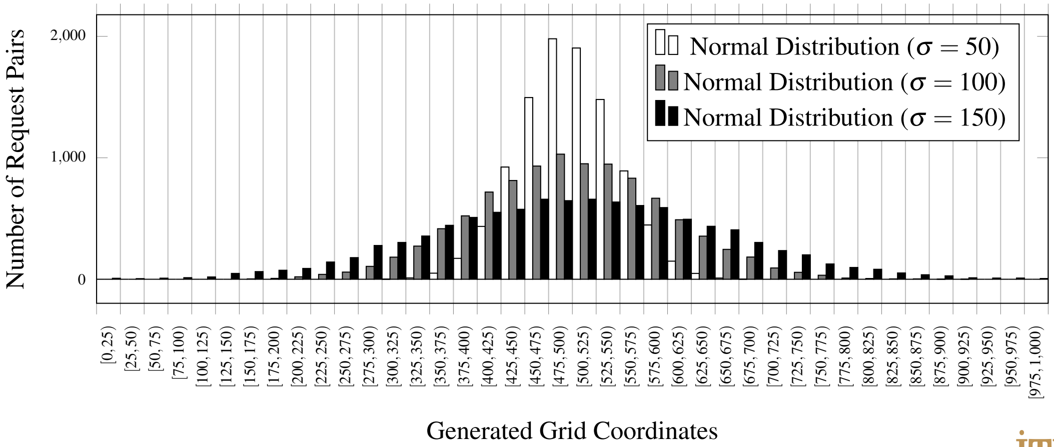
Complete Results



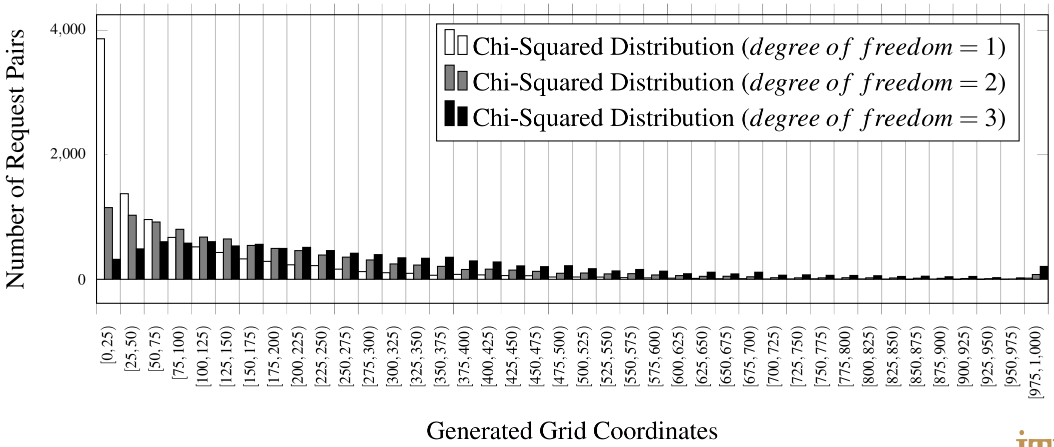
Complete Results



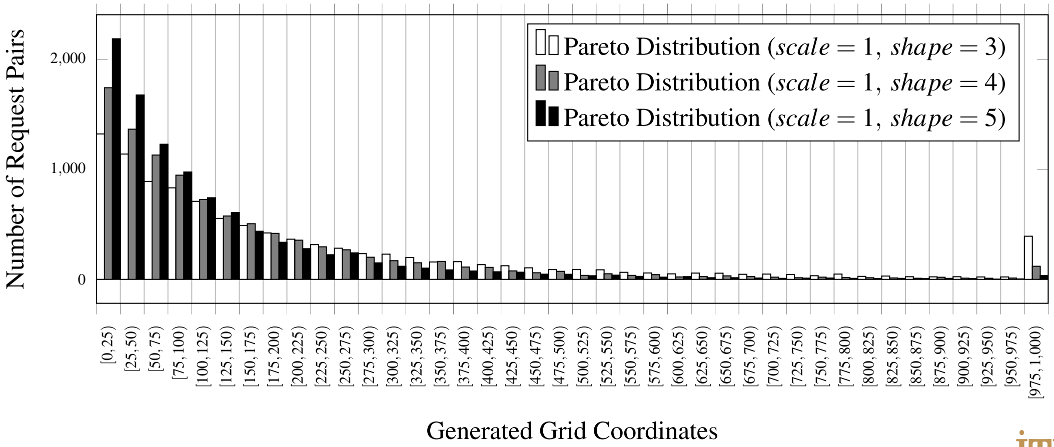
Complete Results



Complete Results



Complete Results



Complete Results