Container Image Placement for Service Provisioning in the Edge

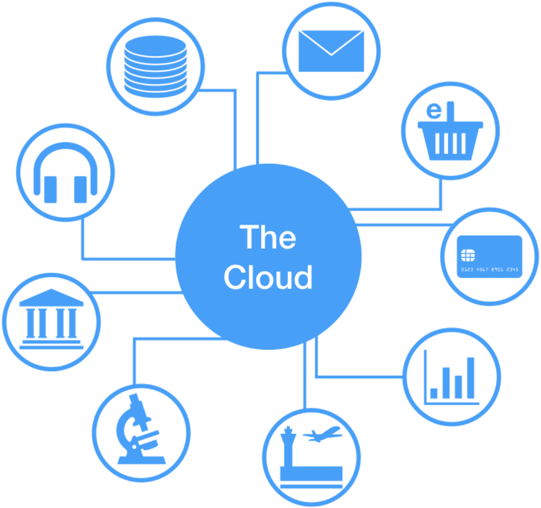
J.Darrous, T.Lambert, S.Ibrahim

Per3S

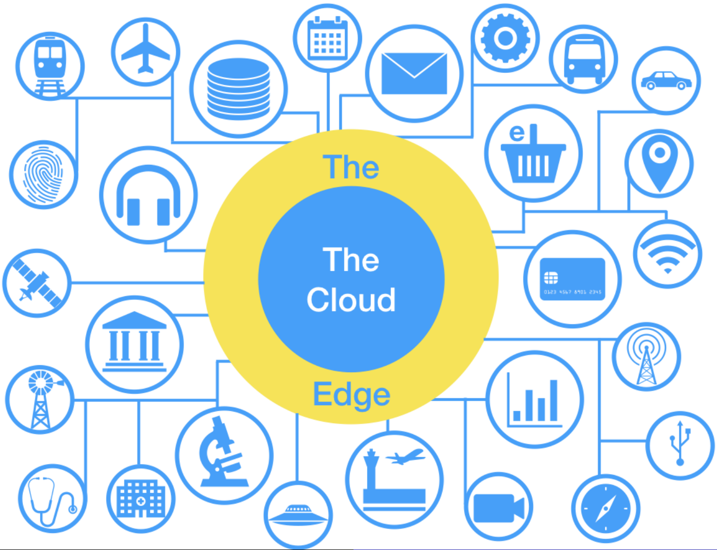
May 30 2023

Table of Contents

Context



Context



Containers in Edge

- Containers are extensively used in cloud data centers
  - Google launches more than 2 billion containers a week[1]

[1] www.theregister.co.uk/2014/05/23/google_containerization_two_billion

Containers in Edge

- Containers are extensively used in cloud data centers
  – Google launches more than 2 billion containers a week[1]
- Containers are widely accepted as the virtualization technology for Edge, due to their lightweight overhead

[1] www.theregister.co.uk/2014/05/23/google_containerization_two_billion

Containers in Edge

- Containers are extensively used in cloud data centers
  – Google launches more than 2 billion containers a week[1]
- Containers are widely accepted as the virtualization technology for Edge, due to their lightweight overhead
- Retrieving images from a central (remote) repository is time consuming
  – Downloading a 500 MB image over 5 MB/s link takes 100s

[1] www.theregister.co.uk/2014/05/23/google_containerization_two_billion

Containers in Edge

- Containers are extensively used in cloud data centers
  – Google launches more than 2 billion containers a week[1]
- Containers are widely accepted as the virtualization technology for Edge, due to their lightweight overhead
- Retrieving images from a central (remote) repository is time consuming
  – Downloading a 500 MB image over 5 MB/s link takes 100s

What we propose

Placing container images across Edge servers!

[1] www.theregister.co.uk/2014/05/23/google_containerization_two_billion

Service provisioning

- **Goal**: providing *fast* and *predictable* retrieving times for a set of images on the entire network

- Challenges:
  - **Heterogeneity of the network (bandwidth)**
  - **Ensure data availability**
  - **Limited storage capacities**

Service provisioning

- **Goal**: providing *fast* and *predictable* retrieving times for a set of images on the entire network
  ↪ Reduce the maximum time to retrieve an image to any Edge-server
- **Challenges**:
  - **Heterogeneity of the network (bandwidth)**
  - **Ensure data availability**
  - **Limited storage capacities**

Service provisioning

- **Goal**: providing *fast* and *predictable* retrieving times for a set of images on the entire network
  ↪ Reduce the maximum time to retrieve an image to any Edge-server
- **Challenges**:
  - **Heterogeneity of the network (bandwidth)**
    ↪ Network awareness during placement and retrieval
  - **Ensure data availability**
  - **Limited storage capacities**

Service provisioning

- **Goal**: providing *fast* and *predictable* retrieving times for a set of images on the entire network
  $\hookrightarrow$ Reduce the maximum time to retrieve an image to any Edge-server
- **Challenges**:
  - **Heterogeneity of the network (bandwidth)**
    $\hookrightarrow$ Network awareness during placement and retrieval
  - **Ensure data availability**
    $\hookrightarrow$ Replication of images
  - **Limited storage capacities**

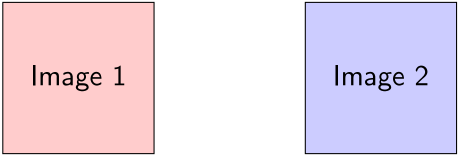
Service provisioning

- **Goal**: providing *fast* and *predictable* retrieving times for a set of images on the entire network
  ↪ Reduce the maximum time to retrieve an image to any Edge-server
- **Challenges**:
  - **Heterogeneity of the network (bandwidth)**
    ↪ Network awareness during placement and retrieval
  - **Ensure data availability**
    ↪ Replication of images
  - **Limited storage capacities**
    ↪ Not too much replications!

Table of Contents

Docker, Images and Layers

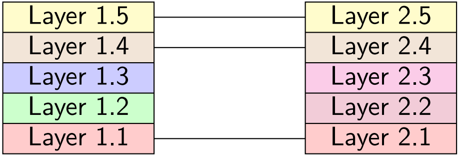
- We base our model on the Docker structure of containers.
- Each image is composed of several layers.
- A layer can be shared between several images.



- Layers are replicated, not images.
- Gain in term of storage cost.

Docker, Images and Layers

- We base our model on the Docker structure of containers.
- Each image is composed of several layers.
- A layer can be shared between several images.



- Layers are replicated, not images.
- Gain in term of storage cost.

Retrieving assumptions

- We focus on placement here but we need to define the retrieving policy.
- **Policy**: If an image is requested on one node, each layer is individually retrieve from the node that owns a replica that has the largest bandwidth.
- **The retrieving time of an image is determined by the longest retrieving time among the ones of its layers.**

MaxLayerRetrievalTime

Problem (*MaxLayerRetrievalTime*)

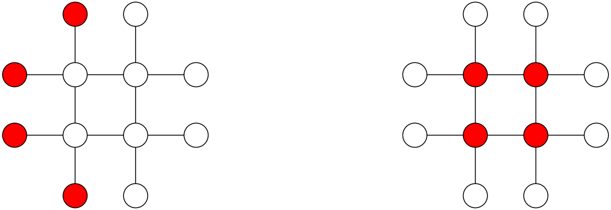
Let V be a set of nodes with storage capacity c and $\mathcal{L}$ be a set of layers. Return a valid placement that minimizes: $\max_{u \in V,\ l_i \in \mathcal{L}} T_i^u$.

- V: set of nodes of the network (seen as a complete graph).
- c: storage capacity of a node (equal for all nodes).
  $\hookrightarrow$ The sum of the sizes of layers stored on each node has to be lower than c.
- T_i^u: retrieving time of layer l_i on node u.
  $\hookrightarrow$ Depends on the size of l_i and on the bandwidth between u and the chosen node.

K-Center

Problem (*K-Center*)

Placing k facilities on a graph such that the maximum distance from any node to any facility is minimized.



- Popular model for Content Delivery Networks (CDNs).

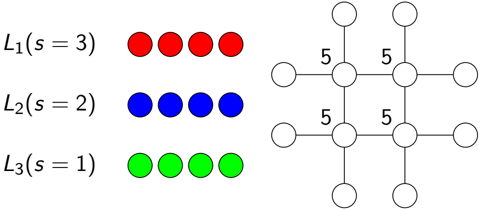
K-Center

- *K-Center* is NP-complete.
  - The best possible approximation ratio is 2.[2].
  - Some algorithm with good average ratio exist (1.058)[3].
- With only one layer, replicated k times, *MaxLayerRetrievalTime* is equivalent to *K-center*.
  $\Rightarrow$ *MaxLayerRetrievalTime* is NP-complete.
- Because of limited storage capacities, all layers cannot be placed on the k most central nodes.

[2] W.-L. Hsu and G. L. Nemhauser, 1979
[3] B. Robič and J. Mihelič, 2005

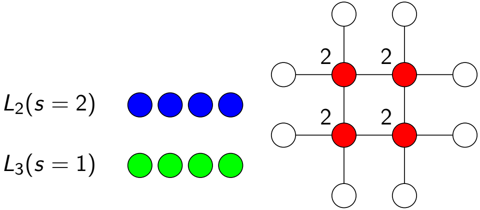
K-Center Based Placement

- Our solution: iterating a K-center approximation algorithm.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$



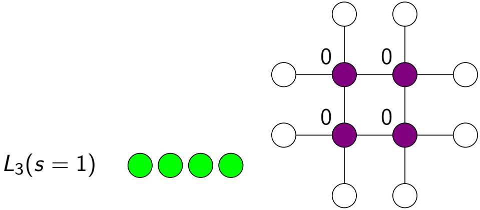
K-Center Based Placement

- Our solution: iterating a K-center approximation algorithm.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$



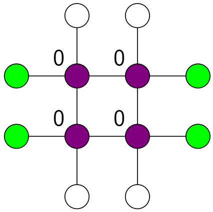
K-Center Based Placement

- Our solution: iterating a K-center approximation algorithm.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$



K-Center Based Placement

- Our solution: iterating a K-center approximation algorithm.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$



MaxImageRetrievalTime

- Several downloads from the same node may degrade the bandwidth.
- Layer level may be too optimistic.
- **New rule**: if several layers are retrieved from the same node, these downloads are done sequentially.

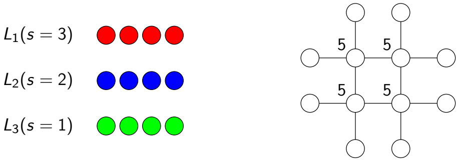
Problem (*MaxImageRetrievalTime*)

Let V be a set of nodes with storage capacity c and $\mathcal{I}$ be a set of images. Return a valid placement that minimizes: $\max_{u \in V, I_j \in \mathcal{I}} T^u_{I_j}$.

- An image is a set of layers.

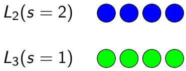
K-Center Based Placement-Without Conflict

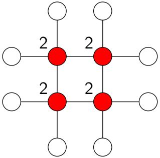
- KCBP tends to gather many layers on same nodes → higher chance to have two layers of an image on the same nodes.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$ **and that do not own layers that share an image with this layer**



K-Center Based Placement-Without Conflict

- KCBP tends to gather many layers on same nodes → higher chance to have two layers of an image on the same nodes.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$ **and that do not own layers that share an image with this layer**

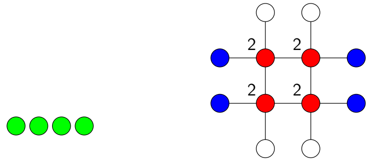




K-Center Based Placement-Without Conflict

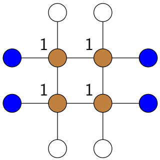
- KCBP tends to gather many layers on same nodes → higher chance to have two layers of an image on the same nodes.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$ **and that do not own layers that share an image with this layer**

$L_3(s = 1)$



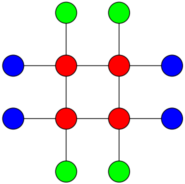
K-Center Based Placement-Without Conflict

- KCBP tends to gather many layers on same nodes → higher chance to have two layers of an image on the same nodes.
- Sort the layers by decreasing sizes
- For each layer:
  - s_i size of the layer
  - Use a K-Center solver (k number of replicas) on the subgraph with all nodes with remaining storage capacities $c_j \geq s_i$ **and that do not own layers that share an image with this layer**



K-Center Based Placement-Without Conflict

- We do not want to spread too much!



- What if another layer share an image with the three previous ones?
- We only apply the criterion "not sharing an image" on the α% largest layers ($\alpha = 10$ here).

Table of Contents

Simulation Methodology

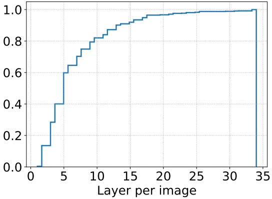
Simulation Methodology

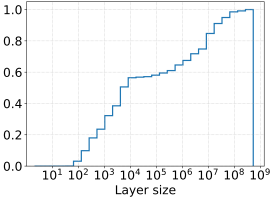
- **Simulator:** written in Python and publicly available at `gitlab.inria.fr/jdarrous/image-placement-edge`
- **Dataset:** cloud container images dataset
- **Networks:** synthetic and real network topologies

Container Images

- IBM cloud traces from Frankfort data centers.

Total #images	996
Total size of images	93.76 GB
Total #layers	5672
Total size of unique layers	74.25 GB





Synthetic Networks

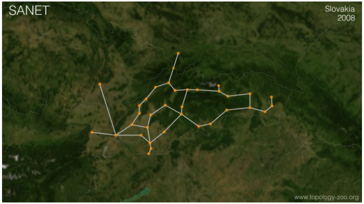
- Complete graphs with random bandwidths on edges.
- **Homogeneous**: same bandwidth for all.
- **Low**: most of the edges have low bandwidth.
- **High**: most of the edges have high bandwidth.
- **Uniform**: edges bandwidths follow a uniform distribution.

Network	**Number of nodes**	**Links bandwidths (bps)**				
		min	**25th**	**median**	**75th**	**max**
Homogeneous	50	4G	4G	4G	4G	4G
Low	50	8M	763M	1G	2G	8G
High	50	478M	5G	6G	7G	8G
Uniform	50	8M	2G	4G	6G	8G

Real Networks

- France and Slovakia national networks[4].

Network	Number of nodes	Links bandwidths (bps)				
		min	25th	median	75th	max
Renater	38	102M	126M	132M	139M	155M
Sanet	35	63M	6G	8G	8G	10G



[4]http://www.topology-zoo.org/

Strategies

- Our placement strategies:
  - KCBP
  - KCBP-WC
- Comparison strategies:
  - Best-Fit (round-robin distribution of layers)
  - Random
  - 50 runs for each.
- All layers are replicated 3 times.
- Storage capacity: $f \times \frac{\text{size of total dataset}}{\text{number of nodes}}$, $f \in \{1.1, 2, INF\}$.

Impact of Conflicts

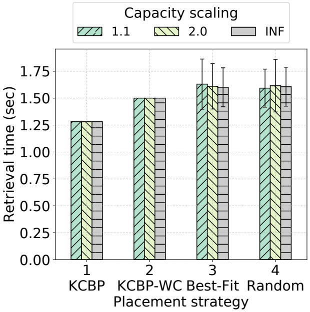


Figure: Layers Retrieval Times (High Network)

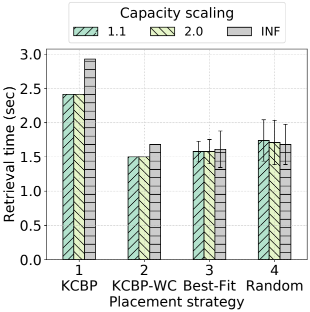


Figure: Images Retrieval Times (High Network)

- Conflicts have significant impact.
- "Extra space effect": having more storage capacity increase retrieving time.

Impact of Heterogeneity of Bandwidths

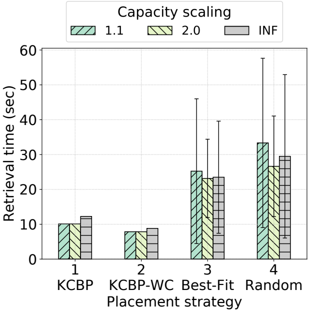


Figure: Low Network

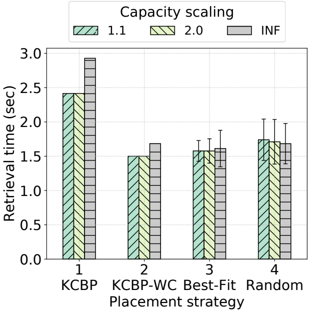


Figure: High Network

- Low Network: many "low connectivity nodes" $\rightarrow$ centrality of layers placement is important.
- High Network: few "low connectivity nodes".

Impact of Heterogeneity of Bandwidths

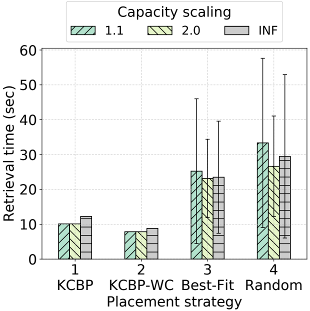


Figure: Low Network

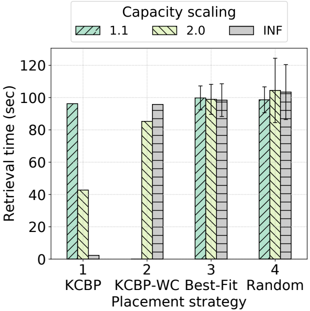


Figure: Sanet Network

- Low Network: many "low connectivity nodes" $\rightarrow$ centrality of layers placement is important.
- High Network: few "low connectivity nodes".

Distribution of Image Retrieving Times

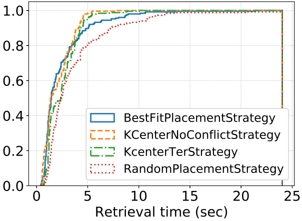


Figure: Low Network

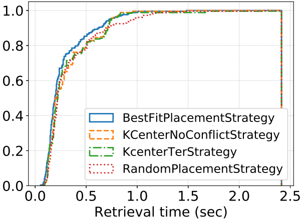


Figure: High Network

- Best-Fit has best retrieving time for 20% of the largest images on High Network.
- For Low Network, KCBP-WC has the lead on these images.

Table of Contents

Contributions and Perspectives

- Contributions:
  - A formal model for containers placement on Edge networks.
  - Two placement strategies.
  - An experimental evaluation with state-of-the-art techniques.
- Perspectives:
  - Improvement of placement strategies ("extra space effect", α).
  - Adding several levels of replication.
  - Location-aware placements.
  - Retrieving strategies.

Any Question?