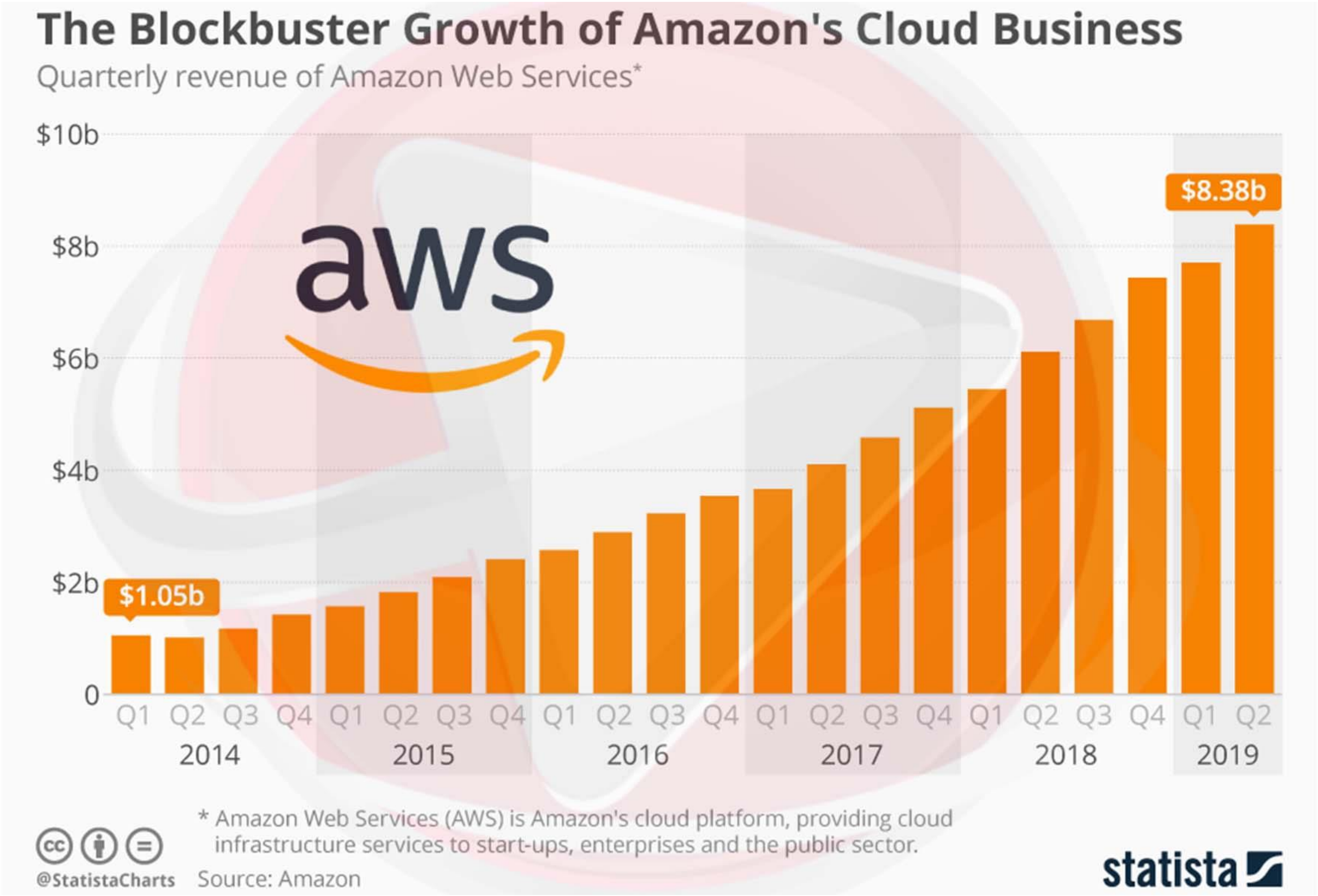
An algorithm for virtual machine consolidation

Ilya Pauzner
Higher School of Economics,
Faculty of Computer Science,
Applied math and informatics

Scientific advisor:
Andrey Tikhonov,
Lead engineer,
Russian Research Institute,
Huawei

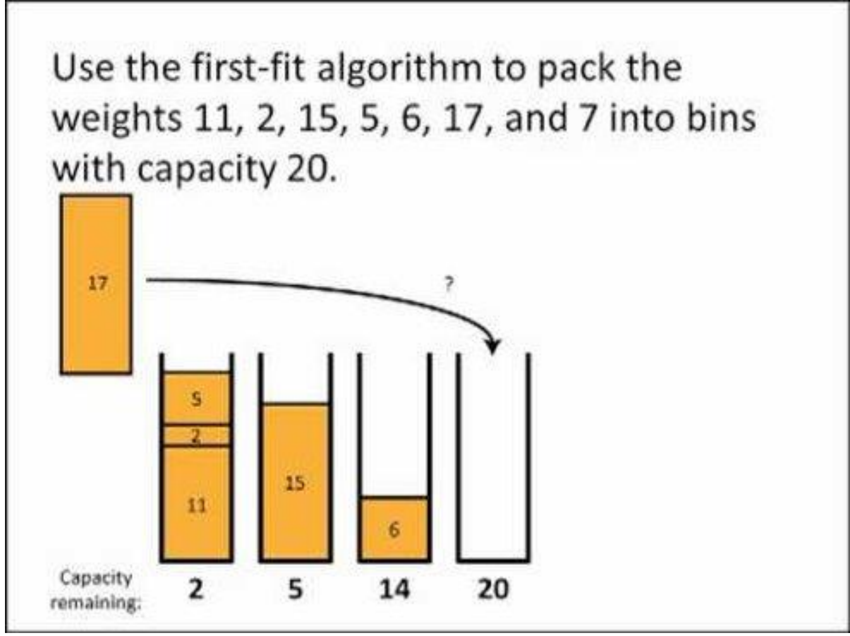
Relevance



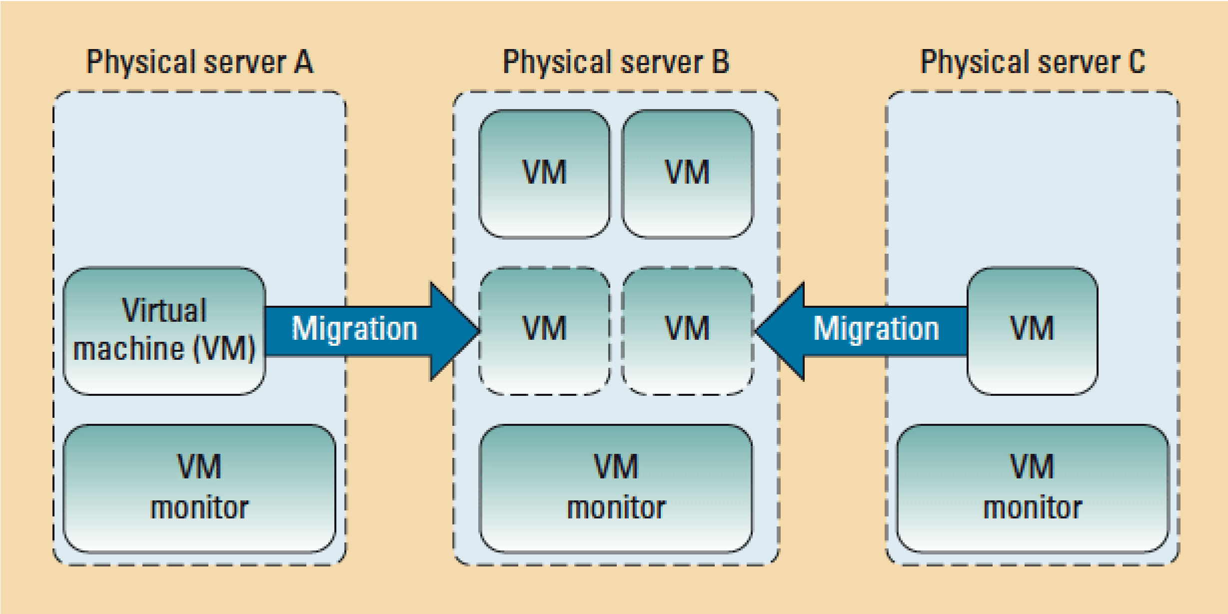
The Blockbuster Growth of Amazon's Cloud Business
Quarterly revenue of Amazon Web Services*
$10b
$8b
$6b
$4b
$2b
0
aws
$1.05b
$8.38b
Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2 Q3 Q4 Q1 Q2
2014 2015 2016 2017 2018 2019
* Amazon Web Services (AWS) is Amazon's cloud platform, providing cloud infrastructure services to start-ups, enterprises and the public sector.
@StatistaCharts
Source: Amazon
statista

Vector bin packing problem (well-known NP-hard)

- VMs (demands) $\mathrm{W}: \overrightarrow{\mathrm{w_i}} = (w_{1,i}, \dots, w_{d,i})$
- Hosts (capacities) $\mathrm{H}: \overrightarrow{c_j} = (c_{1,j}, \dots, c_{d,j})$
- Mapping $f: W \rightarrow H$
- Constraint $\forall j \in H, \forall i \in f^{-1}(j): \sum w_i \leq c_j$
- Active score $\#\{f^{-1}(j) = \emptyset\}$
- Minimize active score



Virtual machine consolidation (not so well-known)



- Items already are placed in bins (Init_Mapping)
- Migration score: total memory of moved VMs

Mathematical problem statement

Indexing VMs as v_i ($i = 1, \dots, n$) and PMs as p_j ($j = 1, \dots, m$), the following binary variables are introduced:

$$Alloc_{i,j} = \begin{cases} 1 & \text{if } v_i \text{ is allocated on } p_j \\ 0 & \text{otherwise} \end{cases}$$

$$Active_j = \begin{cases} 1 & \text{if } p_j \text{ is active} \\ 0 & \text{otherwise} \end{cases}$$

$$Migr_i = \begin{cases} 1 & \text{if } v_i \text{ is migrated} \\ 0 & \text{otherwise} \end{cases}$$

Using these variables, the integer program can be formulated as follows ($i = 1, \dots, n$ and $j = 1, \dots, m$):

$$\min \quad \alpha \cdot \sum_{j=1}^{m} Active_j + \mu \cdot \sum_{i=1}^{n} Migr_i \qquad (4)$$

Optimization objective

$$\text{s. t.} \quad \sum_{j=1}^{m} Alloc_{i,j} = 1 \qquad \forall i \qquad (5)$$

Virtual machine should be on exactly host

$$Alloc_{i,j} \le Active_j \qquad \forall i, j \qquad (6)$$

Host with any VMs should be active

$$\sum_{i=1}^{n} load(v_i) \cdot Alloc_{i,j} \le_d cap(p_j) \qquad \forall j \qquad (7)$$

Capacity constraint

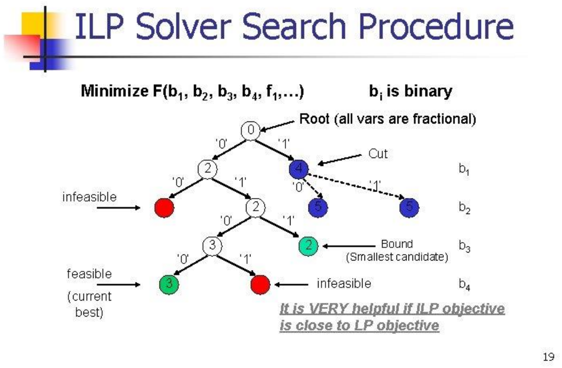
$$Migr_i = 1 - Alloc_{i,map_0(v_i)} \qquad \forall i \qquad (8)$$

Migration is when VM is moved from one host to another

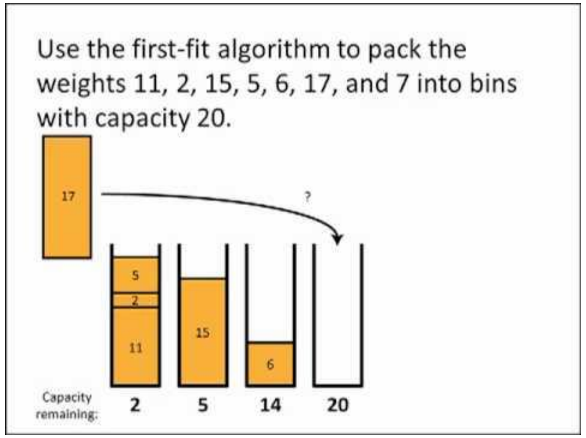
$$Alloc_{i,j}, Active_j, Migr_i \in \{0, 1\} \qquad \forall i, j \qquad (10)$$

Integrality constraint

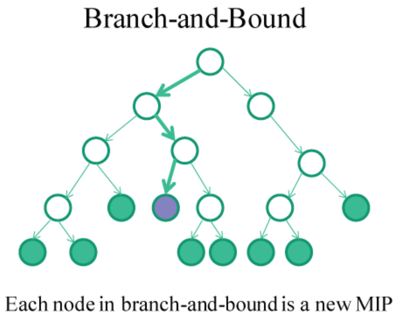
Existing solutions



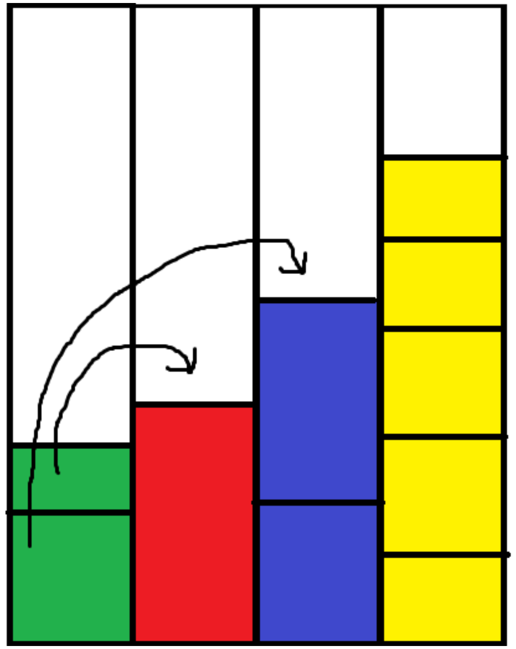
ILP solver
Well-developed industry
Slow



First-Fit-Decreasing heuristic
Simple
High migration cost



Branch and bound
Uses domain knowledge
Slow

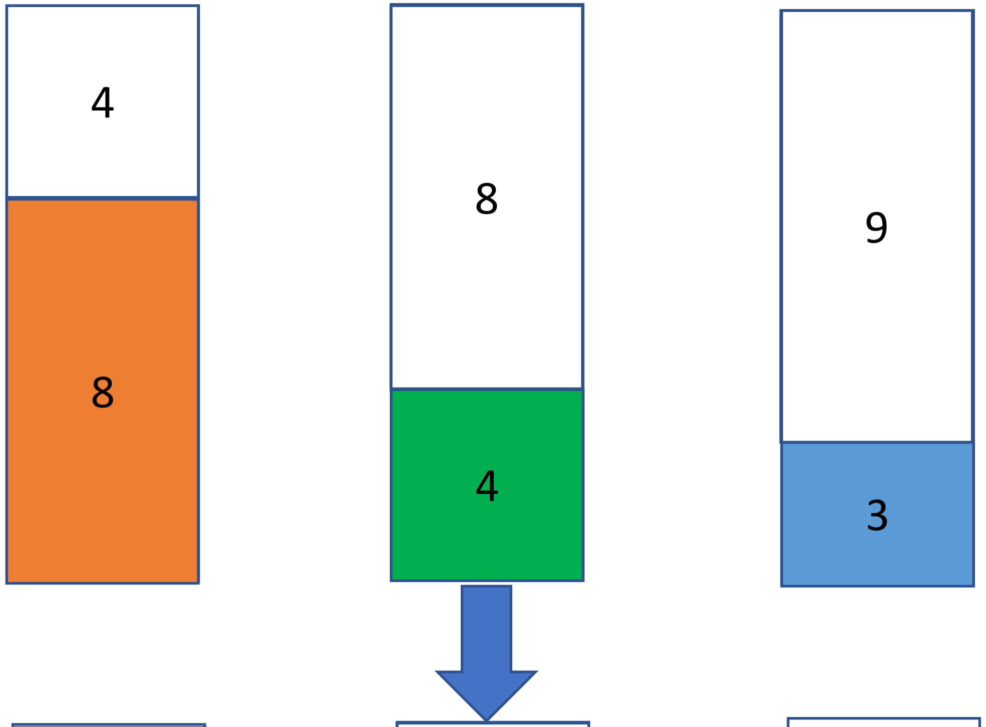


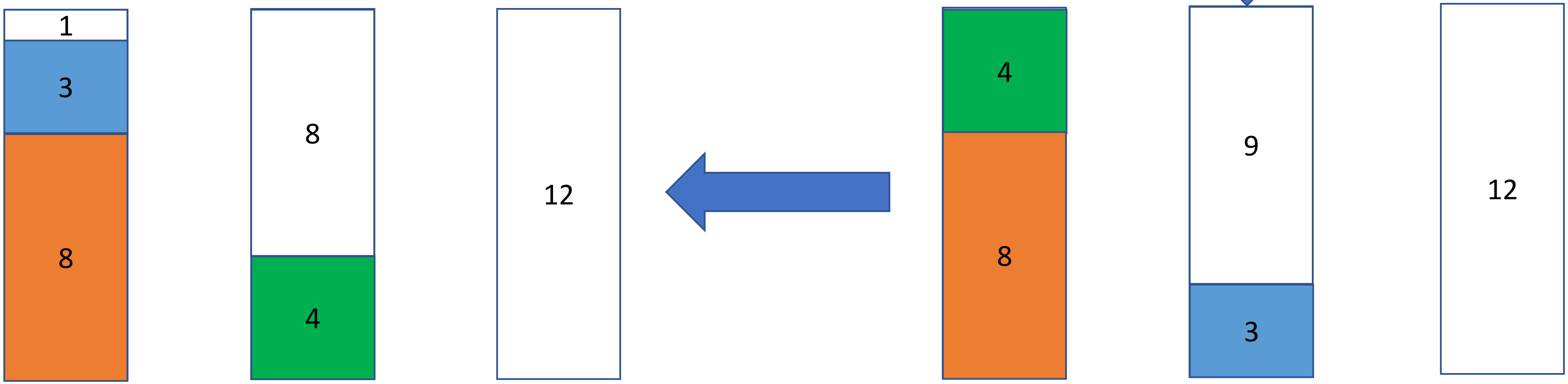
SerCon heuristic
Smallest migration cost
Ineffective packing

Solution in thesis

Two stages

- Bin packing via FFD
- Migration optimization

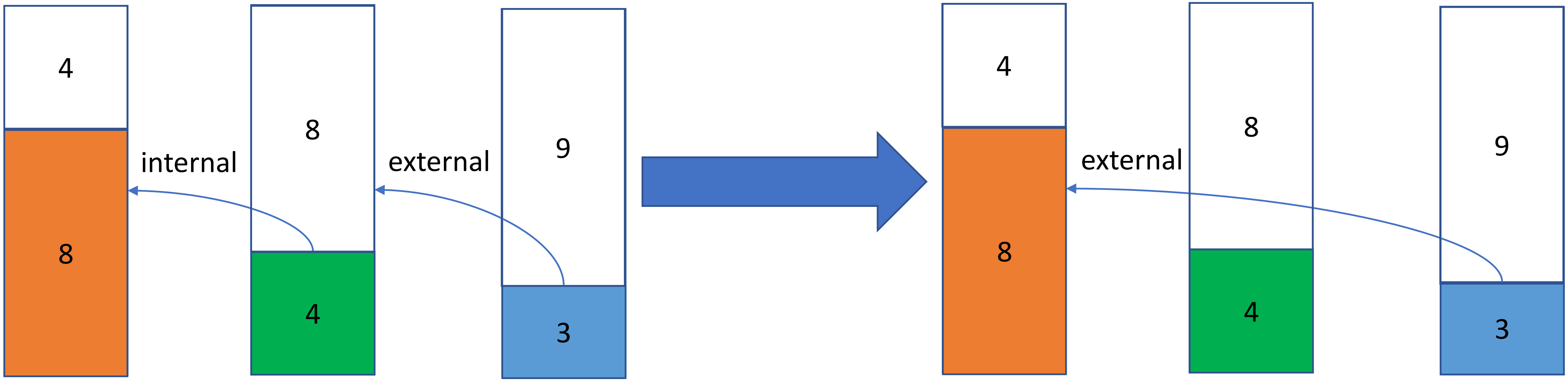




Migration types

Internal migration – source host would be active

External migration – source host would NOT be active



Algorithms in comparison

- Initial (do nothing)
- First-Fit-Decreasing
- First-Fit-Decreasing + Migration Optimizer
- VPSolver
- VPSolver + Migration Optimizer
- SerCon

Algorithms in comparison

- Initial (do nothing)
- First-Fit-Decreasing
- First-Fit-Decreasing + Migration Optimizer
- VPSolver
- VPSolver + Migration Optimizer
- SerCon

Q: Why no SerCon + Migration Optimizer?

A: Migration optimizer can reduce internal migrations, and SerCon produces only external migrations.

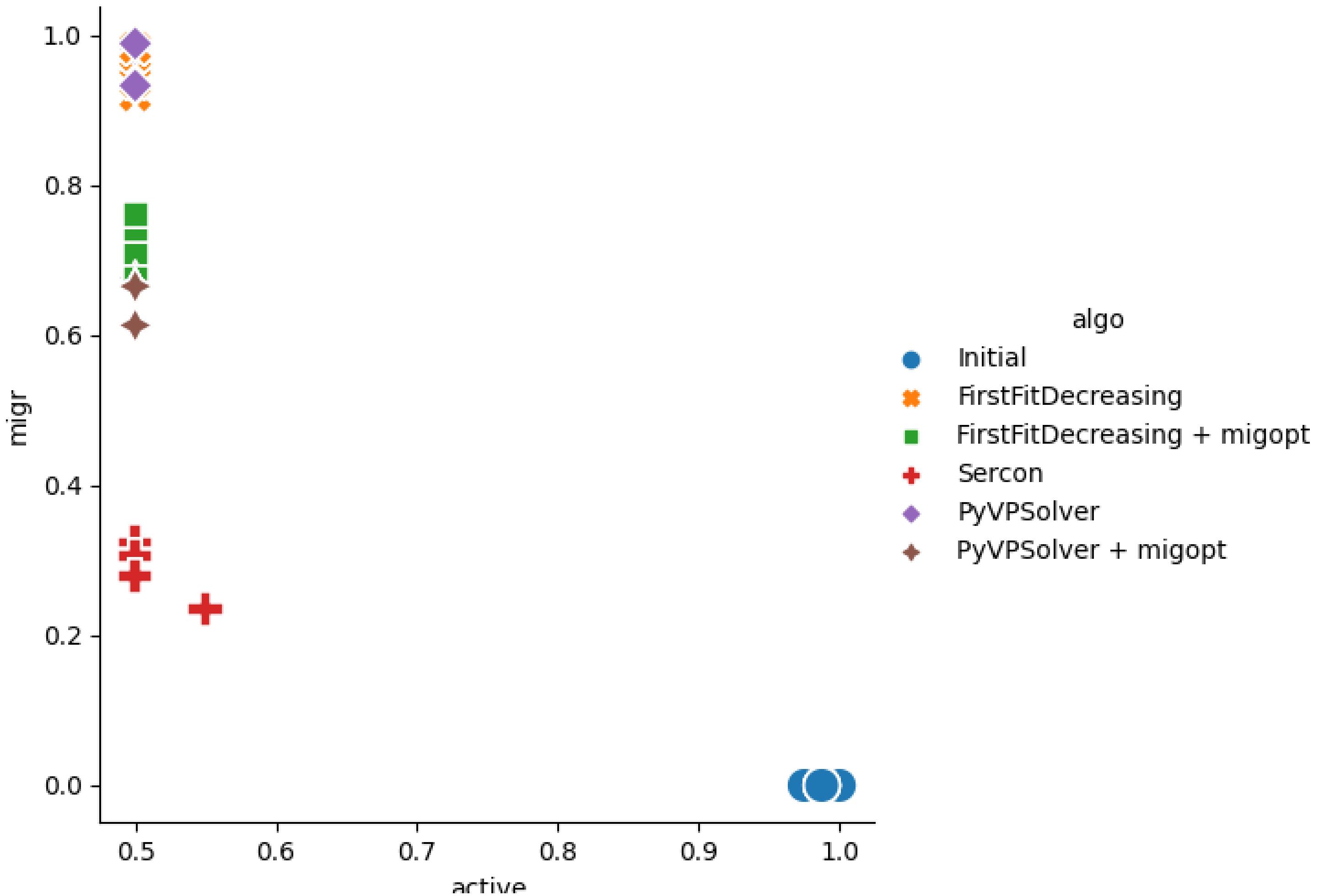
Test types

- Shrink test type
  - Fill hosts with random virtual machines from distribution until first fail
  - Reduce each virtual machine requests by random factor
- Full-pack test type
  - Fill hosts with random virtual machines from distribution until full

Two variants of distributions of virtual machine sizes.

They are based upon real-world statistics of large cloud provider.

Results (on 1 test)



Future research

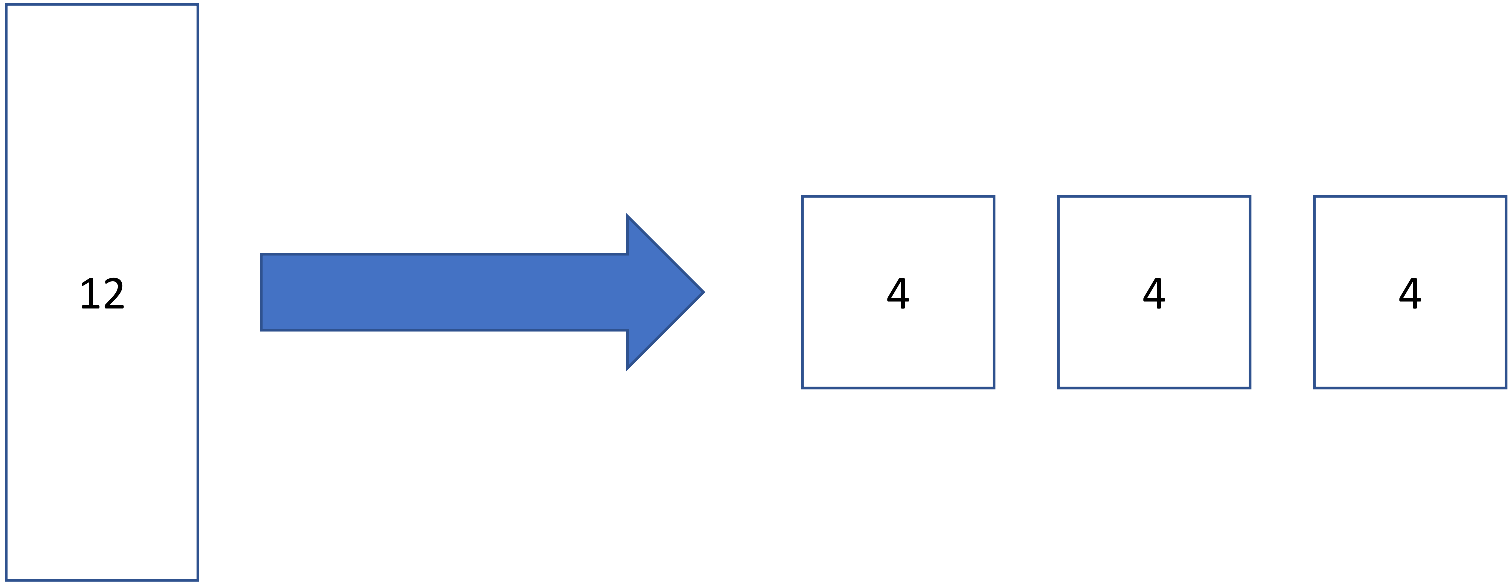
- Different scores:
  - fixed VM amount score: how many VMs on certain size can be put into this configuration.
  - balancing score: how far the load distribution across the hosts from uniform.
- Better first step heuristic than FFD.
- Migration optimizer improvements.

Main points

- Formulated combinatorial optimization problem
- Provided application for cloud systems
- Reduced to assignment problem
- Pareto curve in results
- Demonstrated 30% decrease in migration score

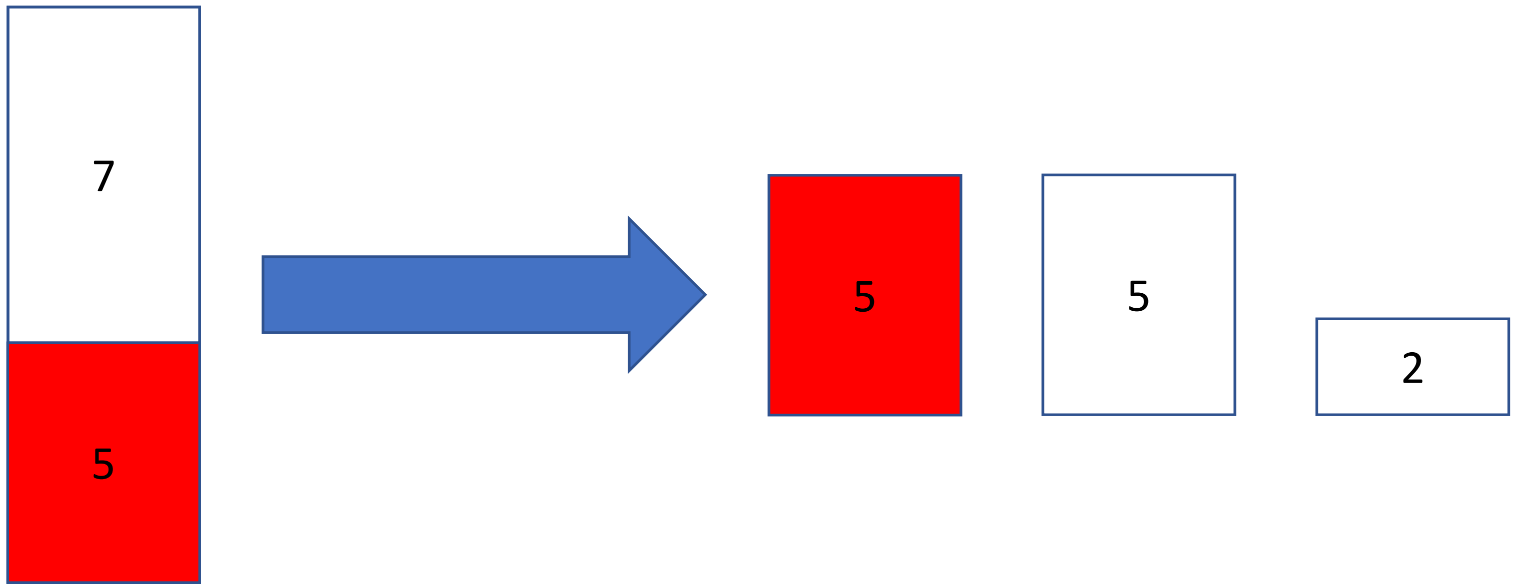
Migration optimizer ideas (questions section)

- Mapping can be arbitrarily changed if active hosts remain the same.
- Hole system (akin to partition of unity in topology):
  One can partition host to non-overlapping parts of smaller size.
- Each hole can accommodate VMs of size not bigger than itself.



Hole making rules

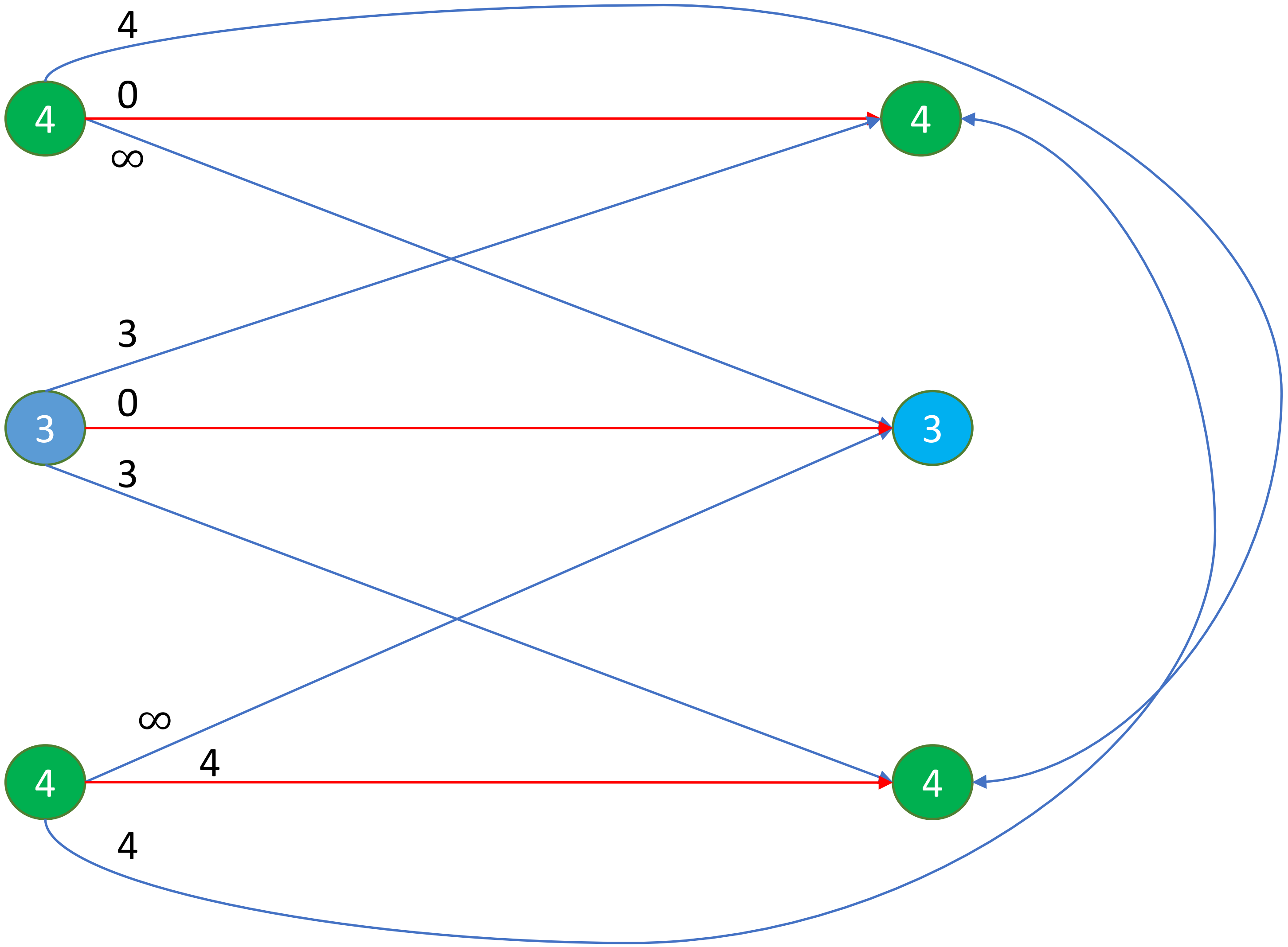
- VM of size n produces hole of size n
- Free space is divided in holes of largest VM size + remainder



Initial and final hole system

- Initial hole system is done from initial mapping.
- Final hole system is done from final mapping, using only active hosts.
- Mapping change between initial and final mapping is a matching between holes in initial and final hole systems: which VM goes where.
- Since we have weights assigned to migrations, it is natural to assign weights to the edges.
    - infinity, if source hole does not fit on destination hole
    - 0, if source hole and destination hole are on same host
    - memory, if source hole and destination hole are on different hosts

Min-weight maximum matching aka Assignment problem



Optimal matching in red.

Total weight:

0 + 0 + 4 = 4