VM Placement & Migration

Gabriel Scalosub

Borrowed extensively from:

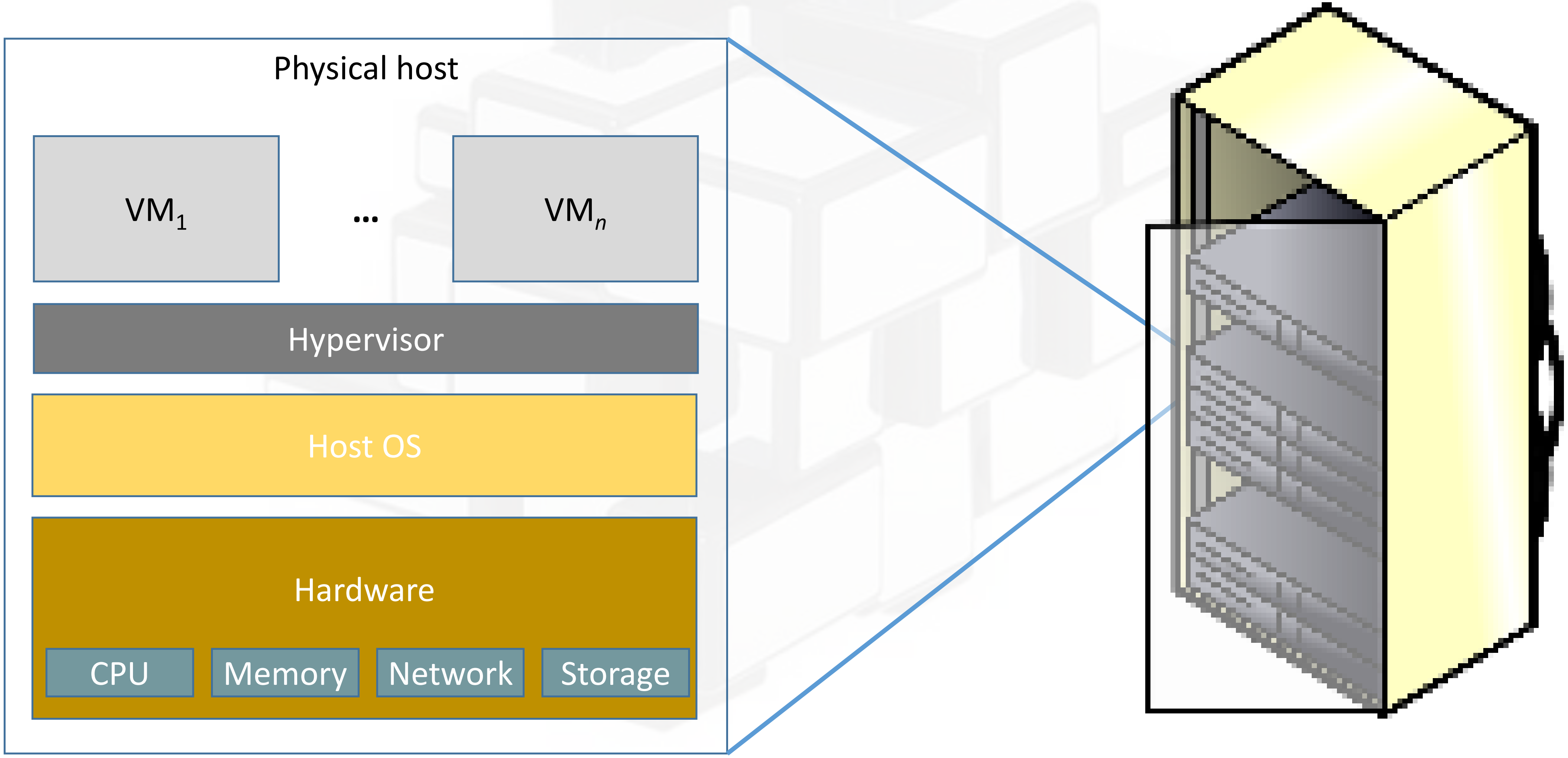
Itai Segall, Danny Raz

and various other papers/resources (see list at the end)

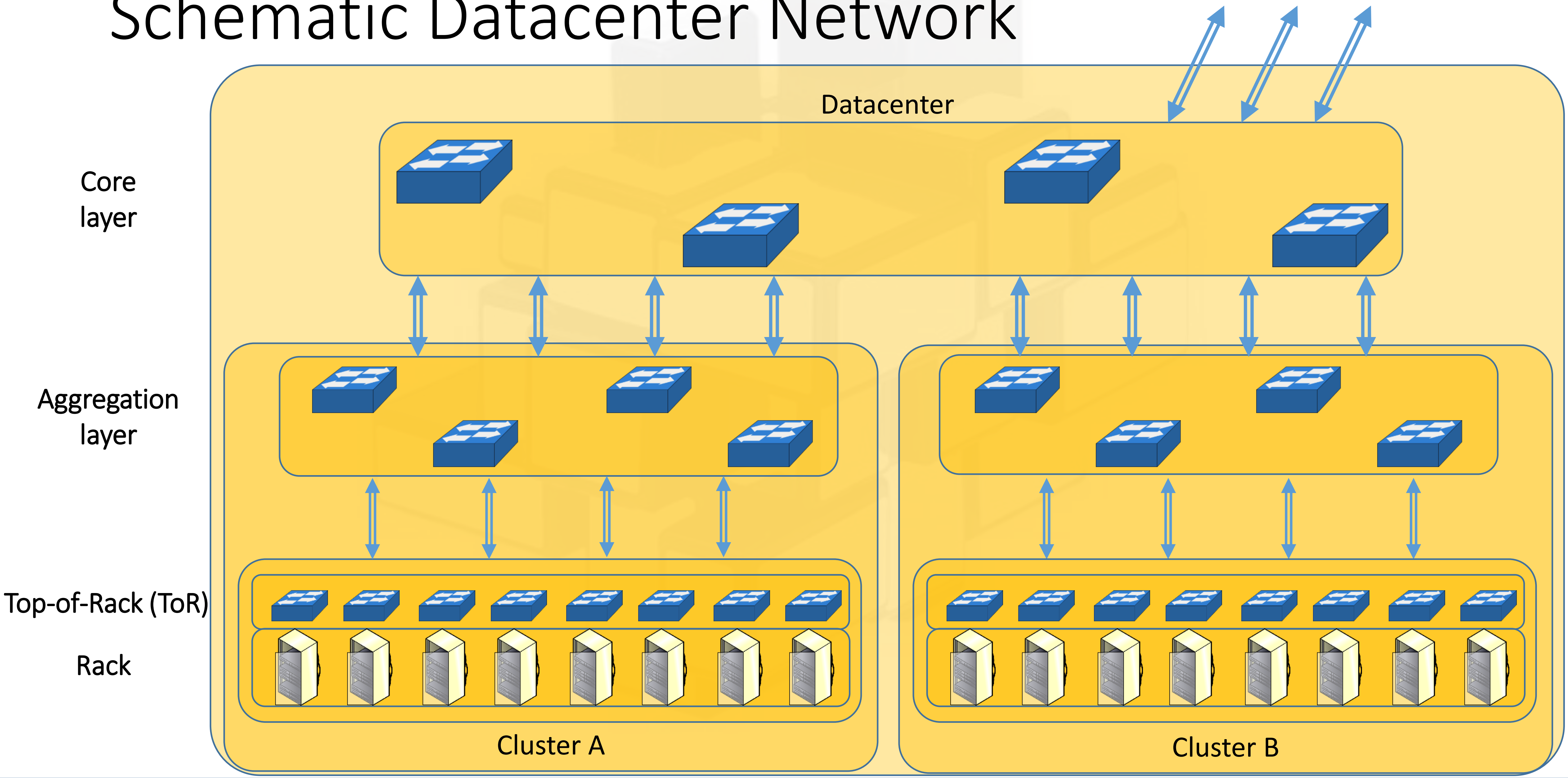
Outline

- Introduction
  - Datacenters, VM requirements and infrastructure characteristics
- VM placement
  - (Multidimensional) bin packing
  - Networking aware
- VM migration

Schematic Cloud Infrastructure

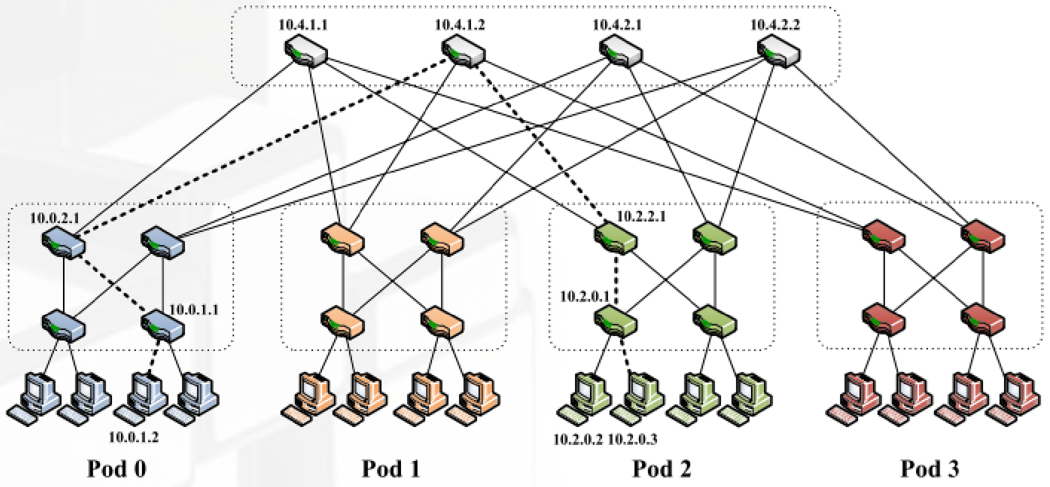


Schematic Datacenter Network



Datacenters Topologies

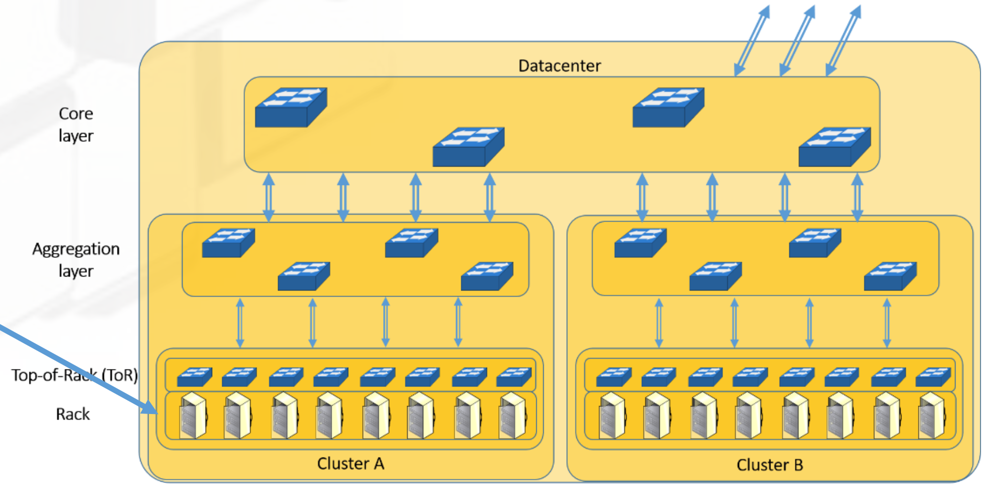
- “Fat-trees” are ubiquitous
- Main objective:
  - Maximize potential of concurrent connections
  - Resilience
- Intra-rack vs. Inter-rack



Al-Fares et al., “A Scalable, Commodity Data Center Network Architecture”, SIGCOMM 2008

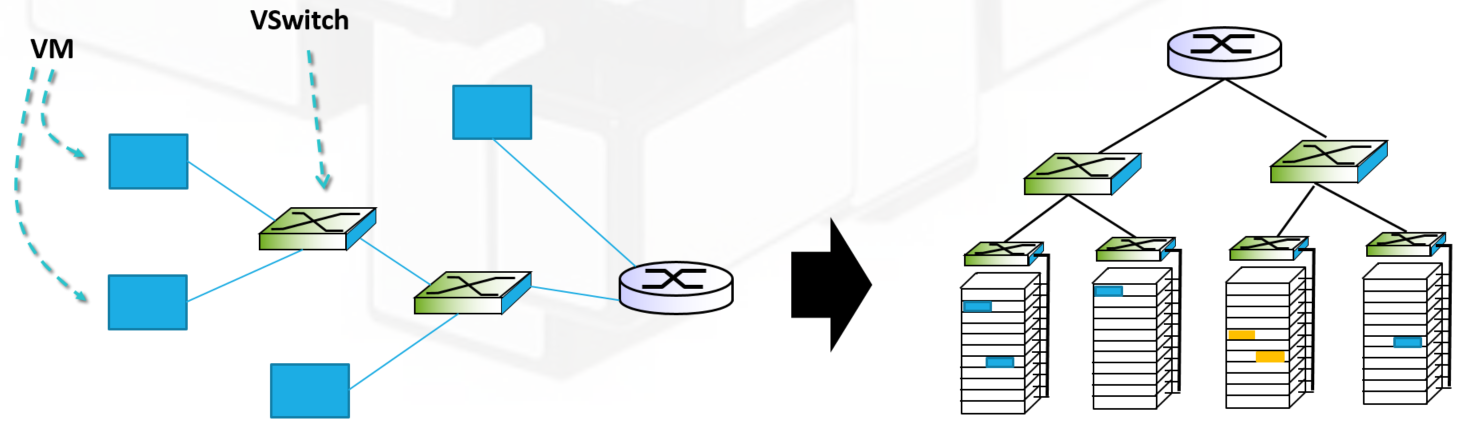
Locality	All
Rack	12.9
Cluster	57.5
DC	11.9
Inter-DC	17.7
Percentage	

Roy et al., “Inside the Social Network’s (Datacenter) Network”, SIGCOMM 2015



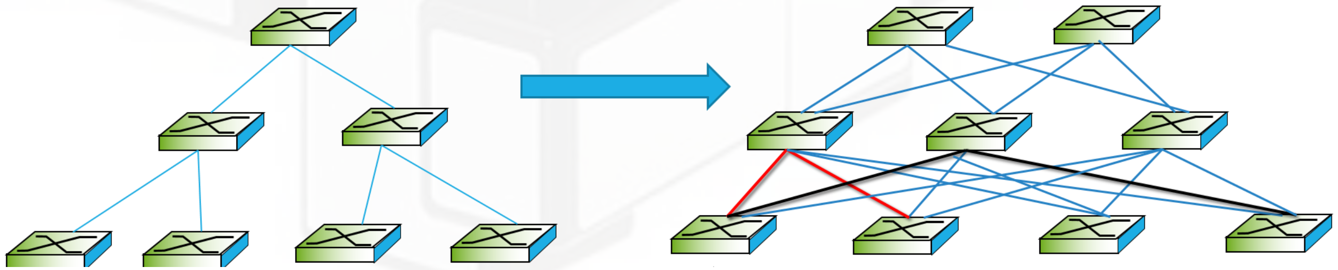
Placement at a High Level

- Tenants have demands for resources
- Placement: decide how to allocate physical resources to tenants
  - Where to run VMs?
  - How to split network bandwidth?
  - How much resources to allocate to each VM?
- Virtual datacenter embedding (VDC)



Topology & Infrastructure vs. Placement

- Classic topology & Infrastructure:
  - Hardware based
  - Application agnostic
  - Single DC
  - Goal: Make resources “uniform”
    - Any selection equally good, single continuous pool of resources
- Placement and resource allocation
  - Software based
  - (Re-)configurable and application-aware
  - Also multiple DCs
  - Goal: Manage resources
    - Potentially heterogeneous
  - In recent years also for networks (SDN)

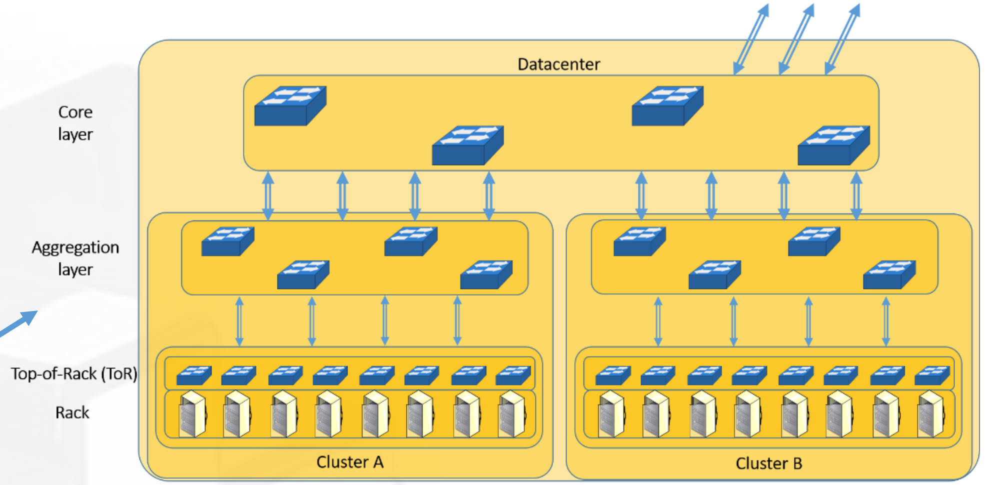


VM Requirements

- CPU
  - #cores $\leq$ #vCPUs
    - vCPU:core ratio (oversubscription)
      - Should be small for CPU-intensive workloads
  - cache
- Memory
  - Major effect on performance
- Acceleration
  - E.g., GPU, TPU
  - For accelerating workloads
    - ML, video processing, big-data analytics
- Storage
  - Directly attached / network
- Network
  - VLAN
  - Bandwidth
  - Latency
- Specific HW architecture
  - E.g., x86, ARM

Workload Requirements

- Virtual network
  - Bandwidth
  - Latency
  - East-West vs. North-South
- Customer policies
  - E.g., VM must be located in US-located DC
- (Dis)affinity
  - E.g., Isolation: both VMs cannot be on same physical host
    - Security, fault-tolerance



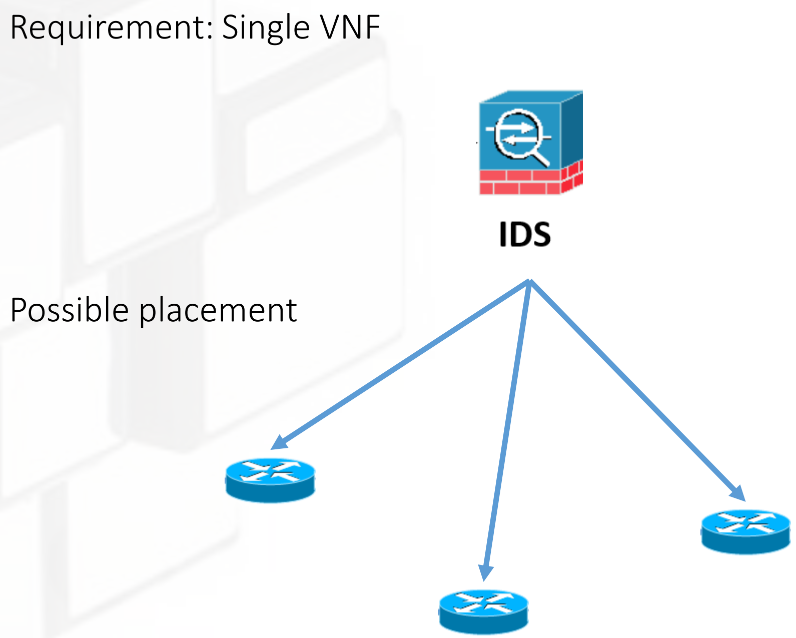
Brooks, "An Introduction to AWS GovCloud (US)", 2016

Meltdown

Spectre

Virtual Network Functions (VNFs)

- Beyond “simple” VM placement
- Simple VNF
  - Single location, “in isolation”
  - E.g., choosing where to place an IDS



Virtual Network Functions (VNFs)

- Beyond "simple" VM placement
- Simple VNF
  - Single location, "in isolation"
  - E.g., choosing where to place an IDS
- VNF chaining
  - Logical traffic steering
  - Implemented by actual placement
    - And forwarding rules… (SDN)

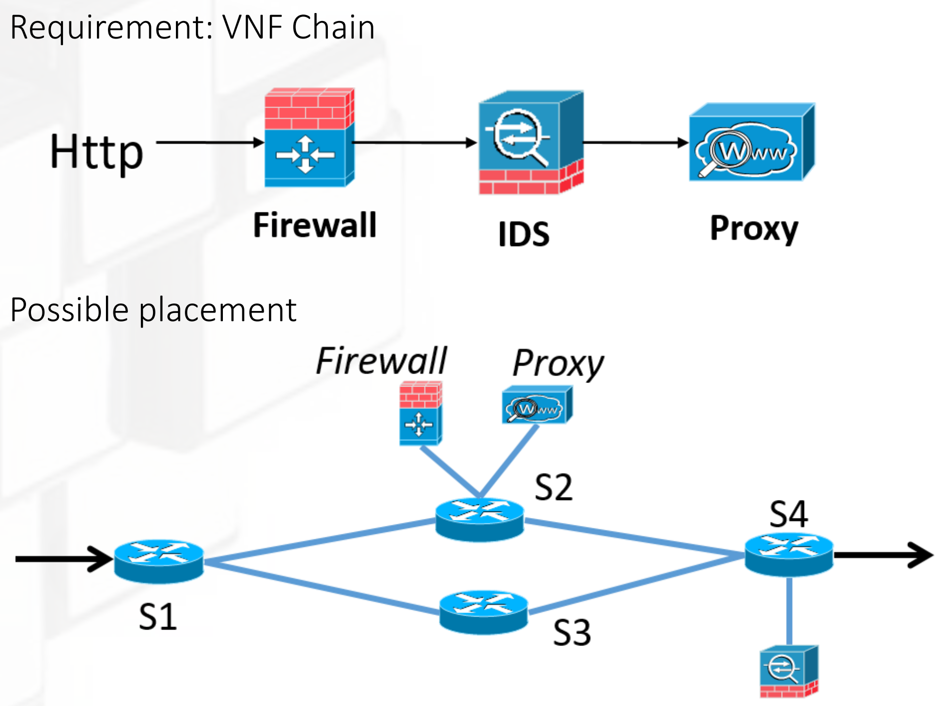


Image: Li & Qian (2016)

- Concretization of abstract Network Function Virtualization (NFV)
  - ≠ SDN

Infrastructure Characteristics

https://aws.amazon.com/ec2/instance-types/
https://www.ec2instances.info/

- E.g., Amazon's EC2
- Non uniform instances
  - Bare-metal
  - HW, virtual resources, pricing…
- Multiple locations

Instance Family	Current Generation Instance Types
General purpose	t2.nano | t2.micro | t2.small | t2.medium | t2.large | t2.xlarge | t2.2xlarge | m4.large | m4.xlarge | m4.2xlarge | m4.4xlarge | m4.10xlarge | m4.16xlarge | m5.large | m5.xlarge | m5.2xlarge | m5.4xlarge | m5.12xlarge | m5.24xlarge
Compute optimized	c4.large | c4.xlarge | c4.2xlarge | c4.4xlarge | c4.8xlarge | c5.large | c5.xlarge | c5.2xlarge | c5.4xlarge | c5.9xlarge | c5.18xlarge
Memory optimized	r4.large | r4.xlarge | r4.2xlarge | r4.4xlarge | r4.8xlarge | r4.16xlarge | x1.16xlarge | x1.32xlarge | x1e.xlarge | x1e.2xlarge | x1e.4xlarge | x1e.8xlarge | x1e.16xlarge | x1e.32xlarge
Storage optimized	d2.xlarge | d2.2xlarge | d2.4xlarge | d2.8xlarge | h1.2xlarge | h1.4xlarge | h1.8xlarge | h1.16xlarge | i3.large | i3.xlarge | i3.2xlarge | i3.4xlarge | i3.8xlarge | i3.16xlarge
Accelerated computing	f1.2xlarge | f1.16xlarge | g3.4xlarge | g3.8xlarge | g3.16xlarge | p2.xlarge | p2.8xlarge | p2.16xlarge | p3.2xlarge | p3.8xlarge | p3.16xlarge

Region & Number of Availability Zones

US East
N. Virginia (6),
Ohio (3)

US West
N. California (3),
Oregon (3)

Asia Pacific
Mumbai (2),
Seoul (2),
Singapore (3),
Sydney (3),
Tokyo (4),
Osaka-Local (1)[1]

Canada
Central (2)

China
Beijing (2),
Ningxia (3)

Europe
Frankfurt (3),
Ireland (3),
London (3),
Paris (3)

South America
São Paulo (3)

AWS GovCloud (US-West) (3)

Amazon EC2 | Overview | Features | Pricing | Instance Types | FAQs | Getting Started | Resources

General Purpose
Compute Optimized
Memory Optimized
Accelerated Computing
Storage Optimized
Instance Features
Measuring Instance Performance

Model	vCPU*	Mem (GiB)	Storage (GiB)	Dedicated EBS Bandwidth (Mbps)	Network Performance (Gbps)
c5.large	2	4	EBS-Only	Up to 3,500	Up to 10
c5.xlarge	4	8	EBS-Only	Up to 3,500	Up to 10
c5.2xlarge	8	16	EBS-Only	Up to 3,500	Up to 10
c5.4xlarge	16	32	EBS-Only	3,500	Up to 10
c5.9xlarge	36	72	EBS-Only	7,000	10
c5.18xlarge	72	144	EBS-Only	14,000	25
c5d.large	2	4	1 x 50 NVMe SSD	Up to 3,500	Up to 10
c5d.xlarge	4	8	1 x 100 NVMe SSD	Up to 3,500	Up to 10
c5d.2xlarge	8	16	1 x 200 NVMe SSD	Up to 3,500	Up to 10
c5d.4xlarge	16	32	1 x 400 NVMe SSD	3,500	Up to 10
c5d.9xlarge	36	72	1 x 900 NVMe SSD	7,000	10
c5d.18xlarge	72	144	2 x 900 NVMe SSD	14,000	25

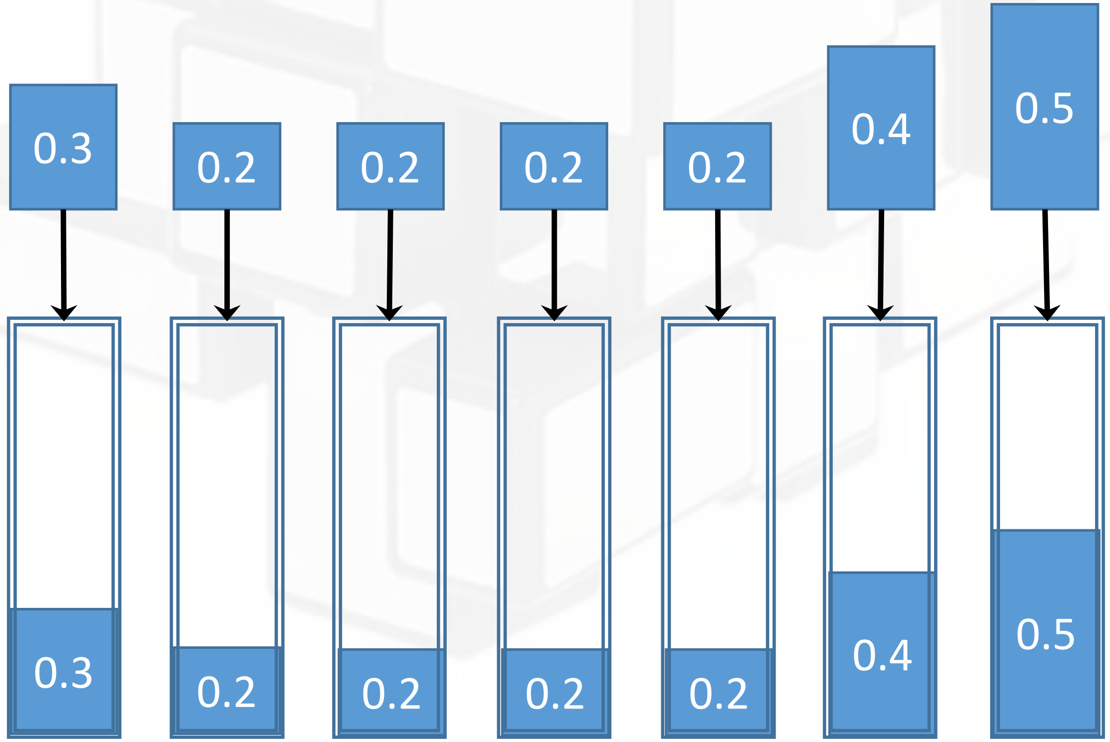
https://aws.amazon.com/about-aws/global-infrastructure/

Outline

- Introduction
  - Datacenters, VM requirements and infrastructure characteristics
- VM placement
  - (Multidimensional) bin packing
  - Networking aware
- VM migration

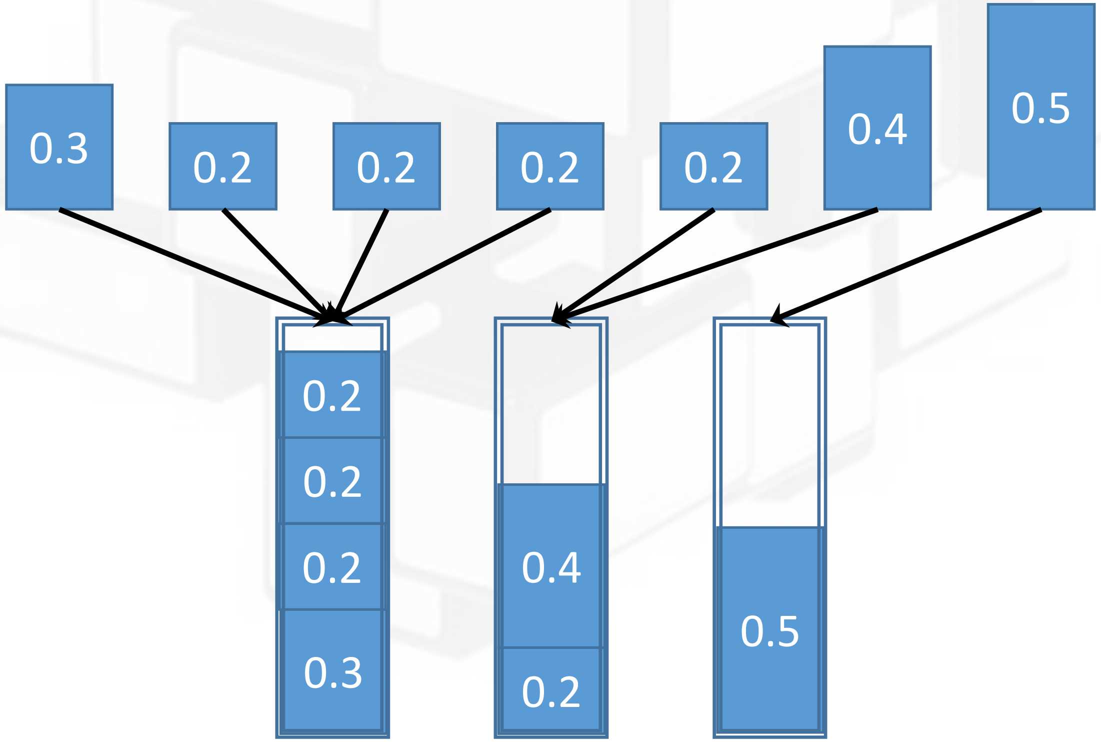
(Short) Introduction to Bin Packing

- Problem definition
  - Input: n items with sizes $a_1, \ldots, a_n$, $\forall a_i \in [0,1]$
  - Goal: pack items in minimum number of unit-size bins



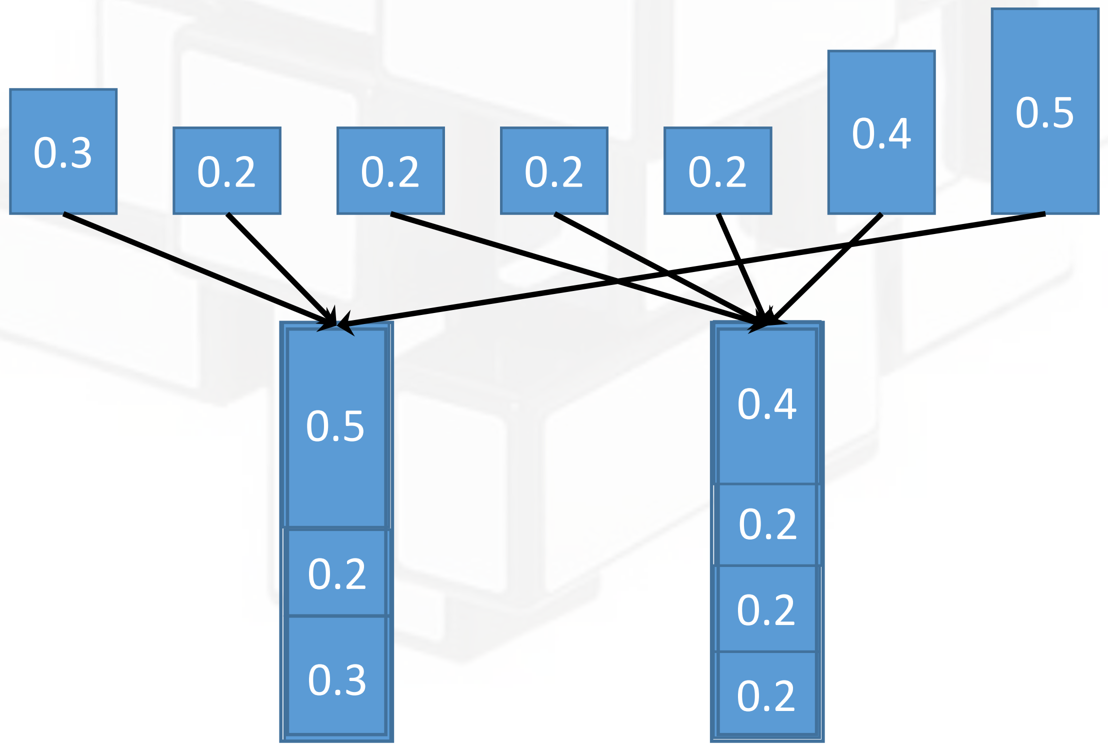
(Short) Introduction to Bin Packing

- Problem definition
  - Input: n items with sizes $a_1, \dots, a_n$, $\forall a_i \in [0,1]$
  - Goal: pack items in minimum number of unit-size bins



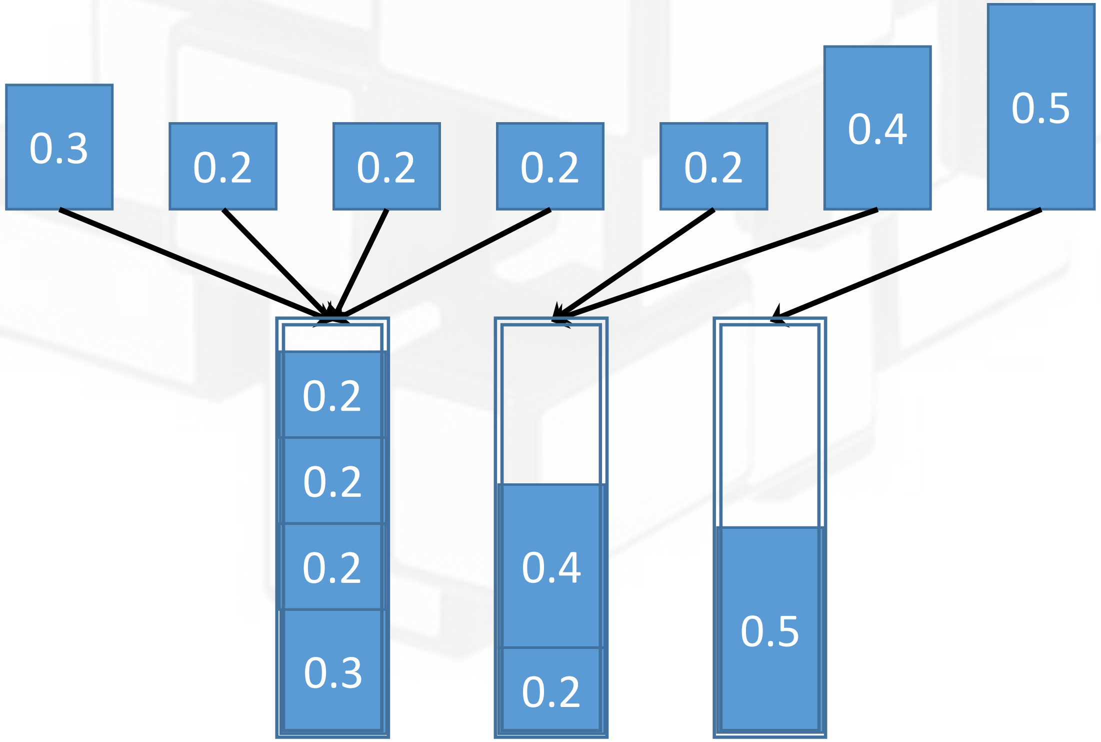
(Short) Introduction to Bin Packing

- Problem definition
  - Input: n items with sizes $a_1, \ldots, a_n$, $\forall a_i \in [0,1]$
  - Goal: pack items in minimum number of unit-size bins



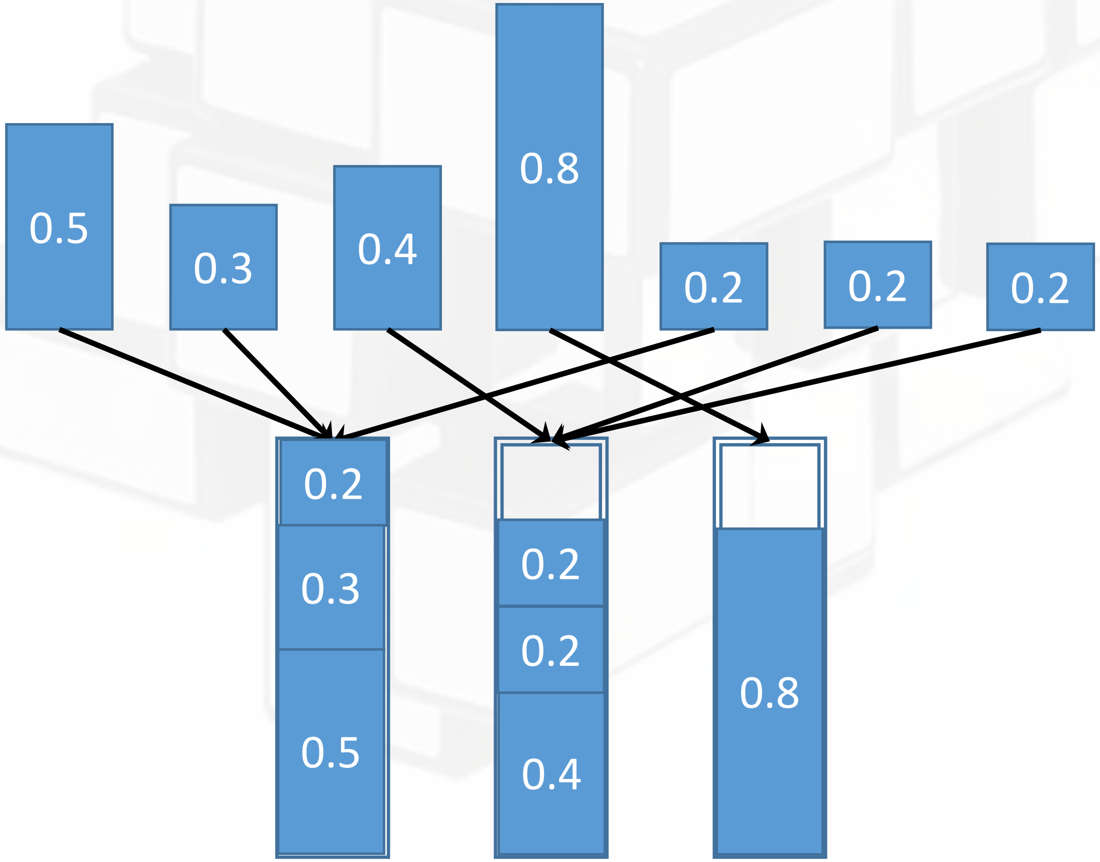
(Short) Introduction to Bin Packing

- First-fit algorithm
  - First bin in which the item fits (order of opening bins)



(Short) Introduction to Bin Packing

- First-fit algorithm
  - First bin in which the item fits (order of opening bins)



(Short) Introduction to Bin Packing

- **Theorem**: First-fit (FF) algorithm is a 2-approximation
  - Number of bins used $\leq 2 \cdot$ minimum number of bins required (OPT)
- **Proof**:
  - Assume FF uses m bins
  - At least $m - 1$ bins are at least half-full
    - 2 bins less than half full -> FF would have put the items together in first bin
  - Load of any solution (also OPT) $= \sum_i a_i > \frac{m-1}{2}$
  - OPT $> \frac{m-1}{2}$
    - OPT = Number of bins used by OPT
  - $2 \cdot$ OPT $> m - 1$
  - $m \leq 2 \cdot$ OPT
    - m and OPT are integers

$\blacksquare$

(Short) Introduction to Bin Packing

- **Theorem**: No algorithm for BP can have approximation strictly less than 3/2
  - Note: If A $< \frac{3}{2}$ OPT then $\frac{2}{3}$ A $<$ OPT
- **Proof**:
  - The partition / subset-sum problem:
    - Given positive numbers $a_1, \ldots, a_n$, can they be partitioned into two sets of equal sum
      - WLOG, $\sum_i a_i = 2$
      - Sum of items in each set is exactly $(\sum_i a_i)/2 = 1$
  - If such an algorithm A exists, it can be used to solve the partition problem:
    - Run A on input to partition
    - If it ends up using 2 bins: Items can be partitioned
    - If it ends up using at least 3 bins: BP-OPT must use *strictly more* than 2/3 · 3 = 2 bins
      - BP-OPT ≥ 3
      - Items cannot be partitioned
  - The partition problem is NP-complete, so cannot be done unless P=NP

■

(Short) Introduction to Bin Packing

- Better approximation guarantees for “large” instances / special cases
  - Asymptotic PTAS (close to OPT as OPT becomes larger)
  - Hardness essentially for “small” instances…
- Online
  - First-Fit (order of opening), Best-Fit (minimum residual space), Worst-Fit (waterlevel)
  - Next-Fit (only current bin is an option, no going-back to skipped bins)
    - Memory efficiency vs. performance
  - Bin partitioning by size
    - E.g., Harmonic: maintain separate bins for items of sizes $\in \left(\frac{1}{k+1}, \frac{1}{k}\right)$
- Offline: heuristics and tighter analysis
  - E.g., ordering items in decreasing order before assignment
    - First-Fit-Decreasing (FFD) / Best-Fit-Decreasing (BFD)

If Only Life Could Be That Simple

- Bin packing: Single concern
- Real life: Multiple concerns!!
  - Height/width/length
    - Geometric bin packing
  - CPU, memory, storage, networking
    - Vector bin packing

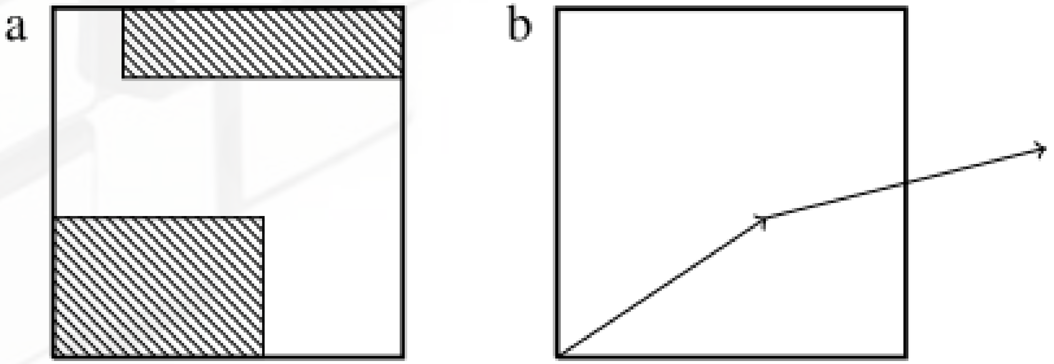
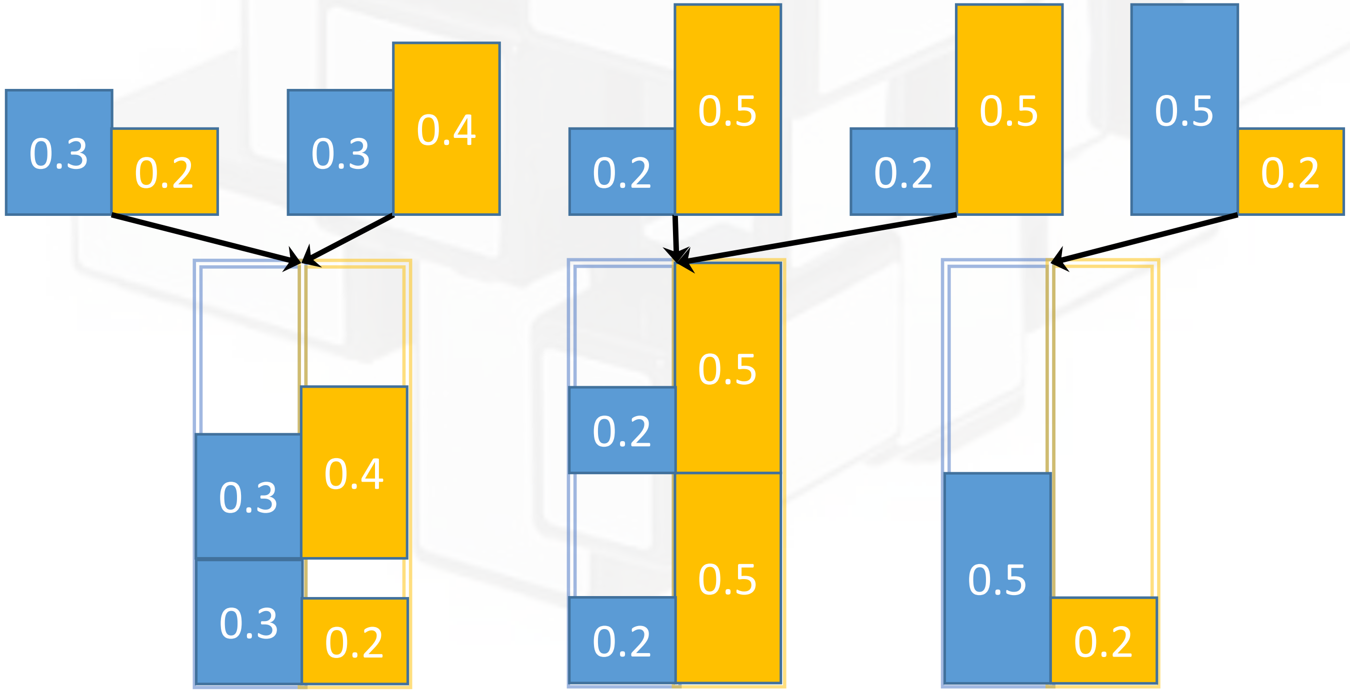


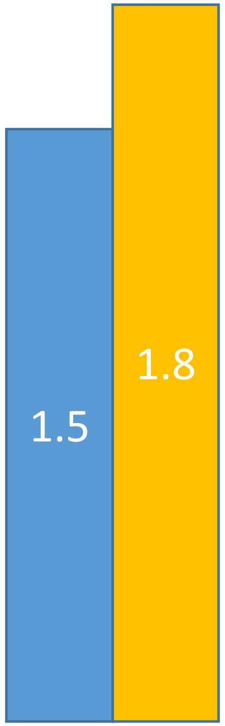
Fig. 1. Two rectangles of size $\frac{3}{5} \times \frac{2}{5}$ and $\frac{4}{5} \times \frac{1}{5}$ can be packed into one unit square bin. However, vectors $(\frac{3}{5}, \frac{2}{5})$ and $(\frac{4}{5}, \frac{1}{5})$ cannot be packed into $(1, 1)$ vector bin.

Christensen et al., “Approximation and online algorithms for multidimensional bin packing: A survey”, Computer Science Review 24:63-79, 2017

Multidimensional Bin Packing (Vectors)

- Problem definition
  - Input: n items represented by vectors $a_1, \dots, a_n$, $\forall a_i \in [0,1]^d$
    - a_{ij} is item i's size in its j-th dimension
  - Goal: pack items in minimum number of unit-vector bins





Overall load is small.
Can you find a better solution?

Multidimensional Bin Packing (Vectors)

- Hard to approximate better than $d^{1-\epsilon}$
  - Reduction from graph coloring
  - Focus on constant d
- $O(\log d)$-approximation algorithms (for constant d)
  - (multiple) LP-based, using rounding
  - Partitioning items by sizes
  - Matching
  - Resource augmentation

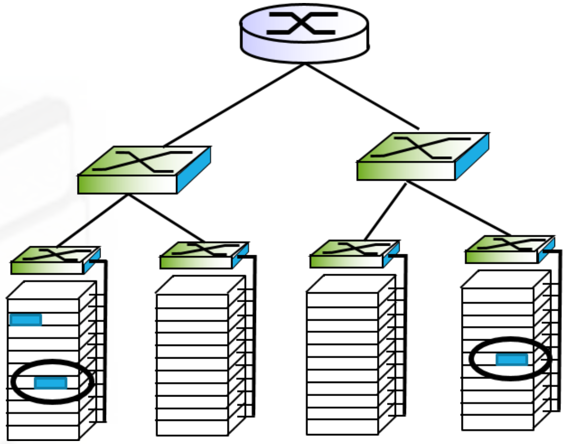
Multidimensional Bin Packing (Vectors)

Panigrahy et al., “Heuristics for Vector Bin Packing”, 2011
Pires and Barán, “Virtual Machine Placement Literature Review”, 2015

- Heuristics
  - Reduction to “standard” (1D) bin packing
  - Applying FFD-like approaches
- Focus on sub-objectives
  - Maximize revenue
  - Minimize energy-consumption
  - Minimize customer rejection-rate
  - Minimize latency
  - Ensure service robustness
  - …
- Which should we pick??

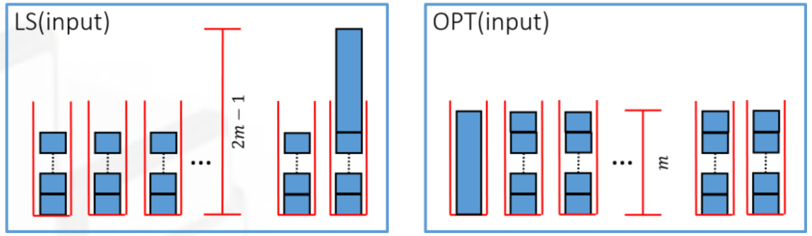
Placement Goes Beyond Just Bin Packing

- Multidimensional bin packing
  - Allocates resources locally
- But,
  - Indifferent to network traffic requirements
    - E.g., East-West traffic load due to placement
  - VM density on host affects resources multiplexing
    - Memory mapping, CPU sharing (scheduling)
    - VM performance depends on other co-located VMs' performance
  - Customer workload dynamics
    - Greedy (present only) vs. proactive allocation (potentially wasteful)
    - Migration?
      - Not without cost…



(Some) Related Problems

Vazirani, Approximation algorithms, Springer, 2001

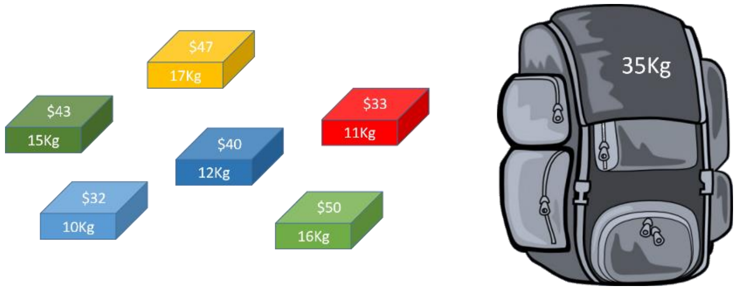


Should look familiar….

- Makespan
  - Input: n items with sizes $s_i \in \mathbb{R}^+$ and m machines
  - Goal: distribute items across machines while minimizing maximum machine load
  - Models:
    - Minimizing oversubscription, validating SLA-conformance

Should also look familiar….

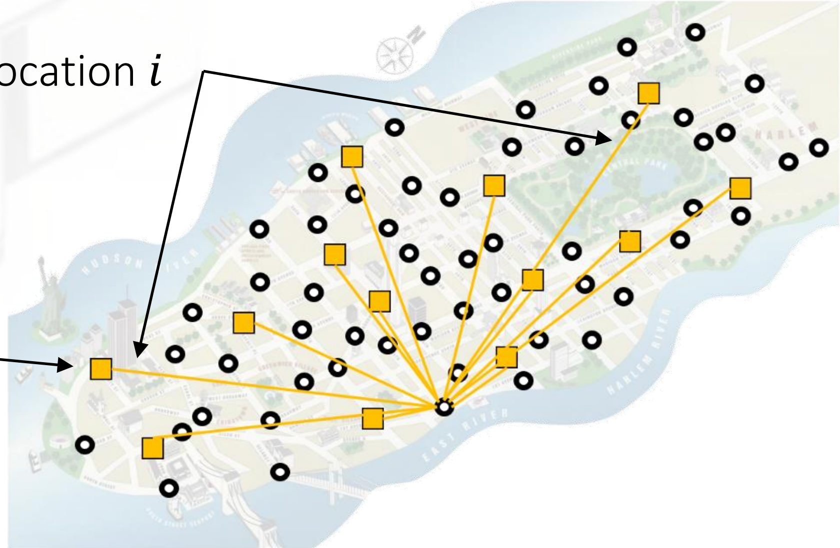
- Knapsack
  - Input: n items with sizes $s_i \in \mathbb{R}^+$ and profits $p_i \in \mathbb{R}^+$, knapsack size B
  - Goal: Pick subset of items with overall size at most B, and maximum overall profit
  - Variants:
    - Multiple knapsacks, multidimensional knapsack, offline/online
  - Models:
    - Maximize utilization of available resources



(Some) Related Problems

Vazirani, Approximation algorithms, Springer, 2001

- Facility location
  - Input:
    - n potential facility locations with opening costs $f_i \in \mathbb{R}^+$, and
    - m demand locations with connection cost $c_{ij} \in \mathbb{R}^+$ between demand j and facility location i
  - Goal: Decide which facility locations to open
    - Minimize: overall opening costs + overall connection costs of demands to their closest open facility



(Some) Related Problems

Vazirani, Approximation algorithms, Springer, 2001

- Facility location
  - Input:
    - n potential facility locations with opening costs $f_i \in \mathbb{R}^+$, and
    - m demand locations with connection cost $c_{ij} \in \mathbb{R}^+$ between demand j and facility location i
  - Goal: Decide which facility locations to open
    - Minimize: overall opening costs + overall connection costs of demands to their closest open facility
  - Example 1:

opening single location i, opening cost f_i

large connection costs!!

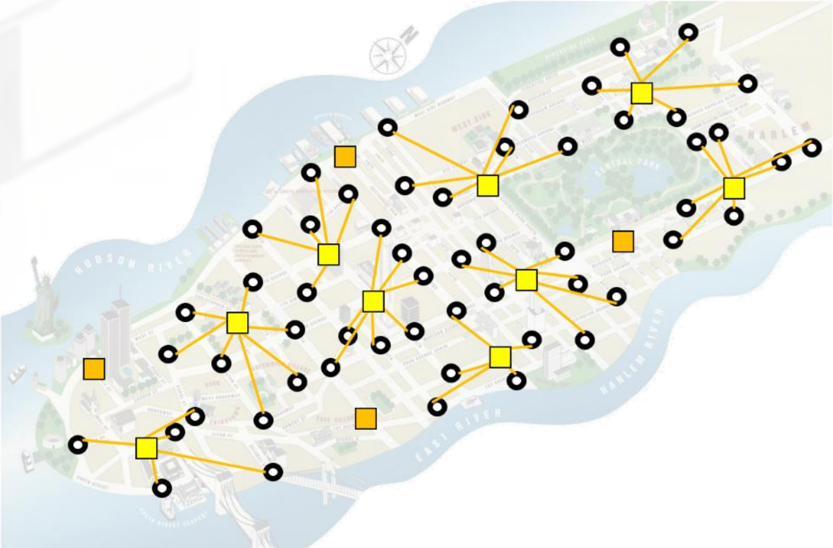
(Some) Related Problems

Vazirani, Approximation algorithms, Springer, 2001

- Facility location
  - Input:
    - n potential facility locations with opening costs $f_i \in \mathbb{R}^+$, and
    - m demand locations with connection cost $c_{ij} \in \mathbb{R}^+$ between demand j and facility location i
  - Goal: Decide which facility locations to open
    - Minimize: overall opening costs + overall connection costs of demands to their closest open facility
  - Example 2:

opening multiple locations i, opening cost f_i

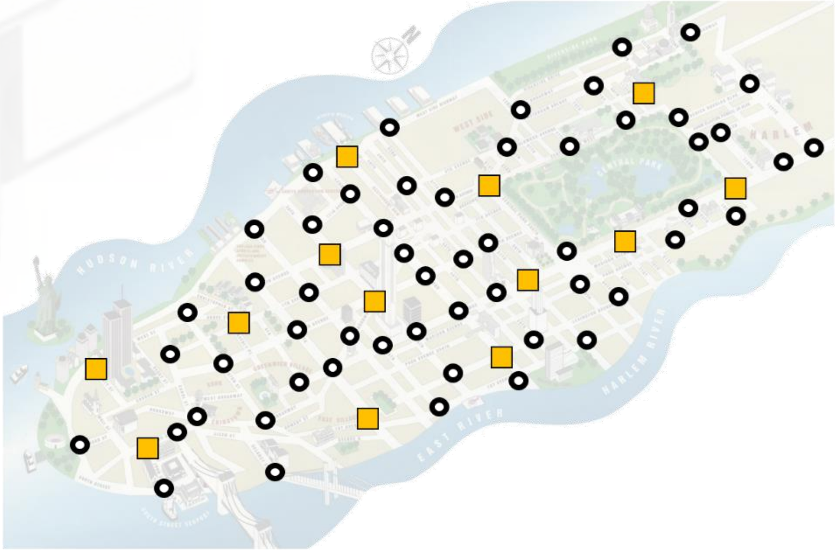
larger opening costs but
much smaller connection costs!!



(Some) Related Problems

Vazirani, Approximation algorithms, Springer, 2001

- Facility location
  - Input:
    - n potential facility locations with opening costs $f_i \in \mathbb{R}^+$, and
    - m demand locations with connection cost $c_{ij} \in \mathbb{R}^+$ between demand j and facility location i
  - Goal: Decide which facility locations to open
    - Minimize: overall opening costs + overall connection costs of demands to their closest open facility
  - Variants:
    - Capacitated (each facility has capacity)
  - Models:
    - Networking performance + placement cost

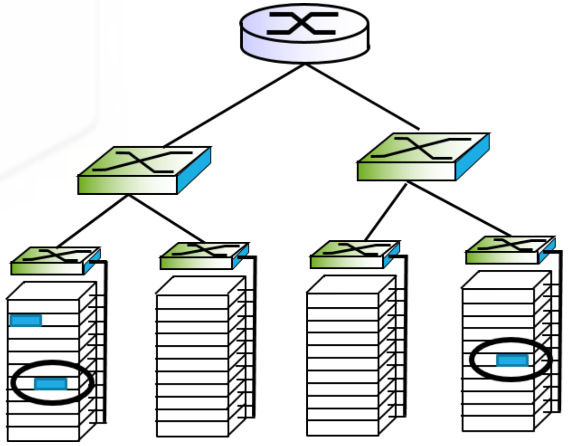


Placement in Real Life

- E.g., OpenStack
    - Open source suite for managing clouds and VMs
    - Filter for subset of hosts based on (dis)affinity
        - Minimum requirements, isolation
    - Optimizes for utilized memory percentage
        - Place where function is maximized: stack VMs
            - À la Best-Fit
        - Place where function is minimized: spread VMs
            - À la Worst-Fit (waterlevel)
    - More on this later in the course

Networking-aware Placement

- Incorporate traffic/network requirements in problem formulation
  - “Simple”: Mixed integer linear program (MILP)
    - Linear constraints and objective
    - Some variables are constrained to be integers (usually {0,1})
  - “Harder”: Mixed integer quadratic programs
  - These are all HARD!
    - Applicable to relatively small instances
    - Heuristics may be (and are) applied



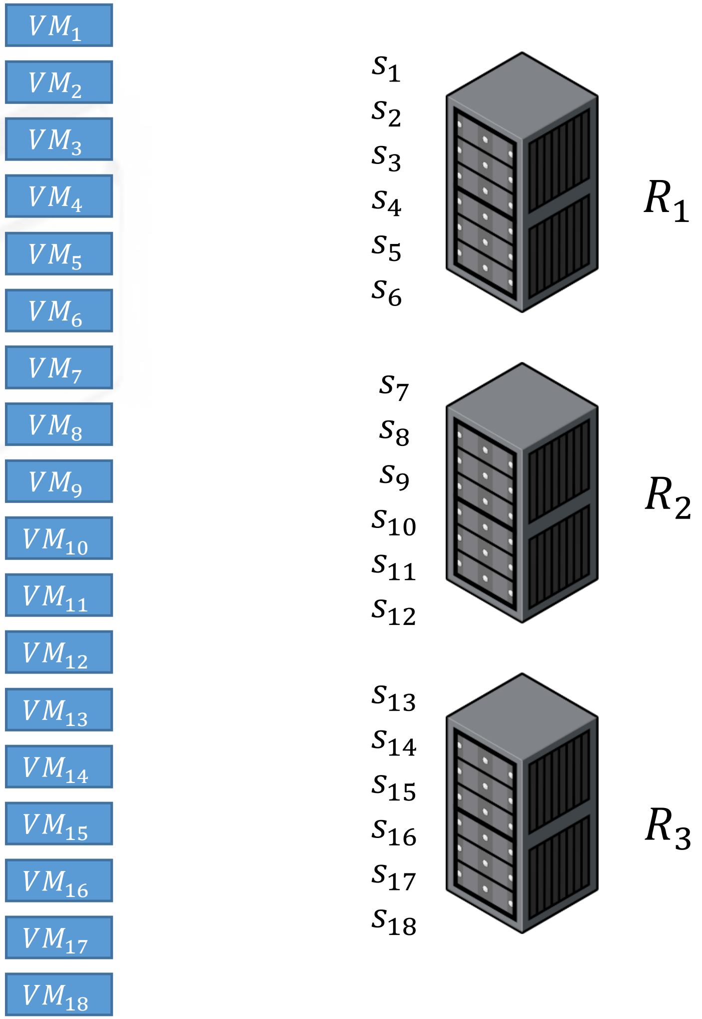
Networking-aware Placement Example

Meng et al., "Improving the scalability of data-center networks with traffic aware virtual machine placement", INFOCOM 2010

- Server physical "slots" $s_1, \dots, s_n$
  - Split among various physical servers
  - Alternatively, servers on racks $R_1, R_2, \dots$
- VMs $v_1, \dots, v_n$

#slots = #VMs

- Placement:
  - Permutation $\pi: \{1, \dots, n\} \mapsto \{1, \dots, n\}$
  - VM j is placed in slot $\pi(j)$
  - Permutation $\equiv$ matching



Networking-aware Placement Example

Meng et al., "Improving the scalability of data-center networks with traffic aware virtual machine placement", INFOCOM 2010

- All VMs have identical local resource requirements
  - For simplicity, no additional constraints
- VM pairs consume different BW:
  - D_{ij} = traffic rate from v_i to v_j
    - location agnostic, depends on application
- Different location-pairs incur different cost:
  - C_{ij} = cost per traffic unit from s_i to s_j
    - application agnostic, depends on slots location
    - E.g., intra-rack cost is 0, inter-rack cost is 1
- Goal: find permutation (matching) minimizing

$$\sum_{i,j} D_{ij} C_{\pi(i)\pi(j)}$$

VMs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0.3	0.4	0.6	0.2	0.3	0.3	0.7	0.9	0.3	0.6	0.7	0.8	0.9	0.4	0	0.3	0.2	0.4
2	0.1	0.5	0.2	1	0.1	1	0.7	0.8	0.1	1	0.5	0.6	0.3	0	0.5	0.7	0.7	0.8
3	0.4	0.8	0.6	0.9	0.5	0.2	1	0.8	1	0.3	0.4	0.9	0.4	0.1	0.1	0.6	0.9	0.7
4	0.7	0.5	0.3	0.2	0.3	0.2	1	0.3	1	0.9	0.5	0.9	0.1	0.8	0.9	0	0.1	0.3
5	0.5	0.3	0.5	1	0.8	0.9	0.6	0.4	0.6	0.4	0.3	0.7	0.1	0.7	0.2	0.5	0.4	0.2
6	0.9	1	0.1	0.3	0.1	0.3	0.6	0.7	0.2	0.6	0.6	1	0.4	0.9	0.8	0.6	0.1	0.9
7	0.9	0.8	0.8	0.4	0.6	0.2	0.8	0.3	0.4	0.9	0.2	0.8	1	0.8	0.8	0.6	0.7	0.5
8	0.9	0.3	0.5	0.7	0.3	0.9	0.2	0.1	1	0.7	0.1	0.9	1	0.8	1	0.6	0.1	0.4
9	0.3	0.9	0	0.1	0.7	0.9	0.2	0	0.2	0.6	0.4	0.1	0.4	0.6	0.3	0.6	0.1	0.8
10	1	0.3	0.8	0.6	0.2	0.7	0.6	0.7	0.3	0.2	0.7	0.3	0.6	0.1	0.6	0.6	0.4	0.4
11	1	0.8	0	0.1	0.6	0.8	0.6	0.1	0.2	0.3	0.2	0.3	0.9	0.2	0.3	0.1	0.4	0.2
12	1	0.4	0.3	0.4	0.4	0.4	0.8	0.6	0.6	0.2	0.2	0.9	0.2	0.2	0.8	0.6	0.7	0.8
13	0.3	0.3	0.6	0.8	0.8	0.9	0.5	0.9	0.9	0.7	0.4	0.3	0.9	0.8	0.9	0.2	0.5	0.8
14	0.5	0.7	0	0.9	0.1	0.1	0.4	0.9	0.8	0.6	0.2	0.8	0.4	0.3	1	0.7	0.3	0.7
15	0.6	0.7	0.4	0.8	0.9	0.6	0.6	0.6	0.1	0.3	0.7	0.9	0	0.6	0.6	0.7	0.2	0.5
16	0.5	0.2	0.4	0.4	0.9	0.1	0.2	0.5	0.1	0.7	0.8	0.1	0.7	0.7	0.4	0.4	0.4	0.1
17	0.9	0	0.6	0.1	0.5	0.2	0.2	0.3	0.3	0.3	1	0.6	0.8	1	0	0.2	0.1	0.8
18	0.2	0.5	0.7	0.9	0.2	0.7	0.5	0.3	0.6	0.1	0.2	0.8	0.5	0.3	0.6	0.3	0.8	0.6

slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1							1	1	1	1	1	1	1	1	1	1	1	1
2							1	1	1	1	1	1	1	1	1	1	1	1
3			0				1	1	1	1	1	1	1	1	1	1	1	1
4							1	1	1	1	1	1	1	1	1	1	1	1
5							1	1	1	1	1	1	1	1	1	1	1	1
6							1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1							1	1	1	1	1	1
8	1	1	1	1	1	1							1	1	1	1	1	1
9	1	1	1	1	1	1			0				1	1	1	1	1	1
10	1	1	1	1	1	1							1	1	1	1	1	1
11	1	1	1	1	1	1							1	1	1	1	1	1
12	1	1	1	1	1	1							1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1						
14	1	1	1	1	1	1	1	1	1	1	1	1						
15	1	1	1	1	1	1	1	1	1	1	1	1			0			
16	1	1	1	1	1	1	1	1	1	1	1	1						
17	1	1	1	1	1	1	1	1	1	1	1	1						
18	1	1	1	1	1	1	1	1	1	1	1	1						

Networking-aware Placement Example

Meng et al., "Improving the scalability of data-center networks with traffic aware virtual machine placement", INFOCOM 2010

- Goal: find permutation (matching) minimizing (change of auxiliary variables)

$$\sum_{i,i'} D_{ii'} C_{\pi(i)\pi(i')}$$

- Quadratic Programming:
  - $x_{ij} \in \{0,1\}$ represents whether VM v_i is in slot s_j
  - Objective is now

$$\sum_{i,i',j,j'} D_{ii'} C_{jj'} x_{ij} x_{i'j'}$$

  - Feasibility constraints (matching)
    - $\forall j, \sum_i x_{ij} = 1$ (exactly one VM per slot)
    - $\forall i, \sum_j x_{ij} = 1$ (exactly one slot per VM)

VMs	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1	0.3	0.4	0.6	0.2	0.3	0.3	0.7	0.9	0.3	0.6	0.7	0.8	0.9	0.4	0	0.3	0.2	0.4
2	0.1	0.5	0.2	1	0.1	1	0.7	0.8	0.1	1	0.5	0.6	0.3	0	0.5	0.7	0.7	0.8
3	0.4	0.8	0.6	0.9	0.5	0.2	1	0.8	1	0.3	0.4	0.9	0.4	0.1	0.1	0.6	0.9	0.7
4	0.7	0.5	0.3	0.2	0.3	0.2	1	0.3	1	0.9	0.5	0.9	0.1	0.8	0.9	0	0.1	0.3
5	0.5	0.3	0.5	1	0.8	0.9	0.6	0.4	0.6	0.4	0.3	0.7	0.1	0.7	0.2	0.5	0.4	0.2
6	0.9	1	0.1	0.3	0.1	0.3	0.6	0.7	0.2	0.6	0.6	1	0.4	0.9	0.8	0.6	0.1	0.9
7	0.9	0.8	0.8	0.4	0.6	0.2	0.8	0.3	0.4	0.9	0.2	0.8	1	0.8	0.8	0.6	0.7	0.5
8	0.9	0.3	0.5	0.7	0.3	0.9	0.2	0.1	1	0.7	0.1	0.9	1	0.8	1	0.6	0.1	0.4
9	0.3	0.9	0	0.1	0.7	0.9	0.2	0	0.2	0.6	0.4	0.1	0.4	0.6	0.3	0.6	0.1	0.8
10	1	0.3	0.8	0.6	0.2	0.7	0.6	0.7	0.3	0.2	0.7	0.3	0.6	0.1	0.6	0.6	0.4	0.4
11	1	0.8	0	0.1	0.6	0.8	0.6	0.1	0.2	0.3	0.2	0.3	0.9	0.2	0.3	0.1	0.4	0.2
12	1	0.4	0.3	0.4	0.4	0.4	0.8	0.6	0.6	0.2	0.2	0.9	0.2	0.2	0.8	0.6	0.7	0.8
13	0.3	0.3	0.6	0.8	0.8	0.9	0.5	0.9	0.9	0.7	0.4	0.3	0.9	0.8	0.9	0.2	0.5	0.8
14	0.5	0.7	0	0.9	0.1	0.1	0.4	0.9	0.8	0.6	0.2	0.8	0.4	0.3	1	0.7	0.3	0.7
15	0.6	0.7	0.4	0.8	0.9	0.6	0.6	0.6	0.1	0.3	0.7	0.9	0	0.6	0.6	0.7	0.2	0.5
16	0.5	0.2	0.4	0.4	0.9	0.1	0.2	0.5	0.1	0.7	0.8	0.1	0.7	0.7	0.4	0.4	0.4	0.1
17	0.9	0	0.6	0.1	0.5	0.2	0.2	0.3	0.3	0.3	1	0.6	0.8	1	0	0.2	0.1	0.8
18	0.2	0.5	0.7	0.9	0.2	0.7	0.5	0.3	0.6	0.1	0.2	0.8	0.5	0.3	0.6	0.3	0.8	0.6

slot	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
1							1	1	1	1	1	1	1	1	1	1	1	1
2							1	1	1	1	1	1	1	1	1	1	1	1
3							1	1	1	1	1	1	1	1	1	1	1	1
4				0			1	1	1	1	1	1	1	1	1	1	1	1
5							1	1	1	1	1	1	1	1	1	1	1	1
6							1	1	1	1	1	1	1	1	1	1	1	1
7	1	1	1	1	1	1							1	1	1	1	1	1
8	1	1	1	1	1	1							1	1	1	1	1	1
9	1	1	1	1	1	1							1	1	1	1	1	1
10	1	1	1	1	1	1				0			1	1	1	1	1	1
11	1	1	1	1	1	1							1	1	1	1	1	1
12	1	1	1	1	1	1							1	1	1	1	1	1
13	1	1	1	1	1	1	1	1	1	1	1	1						
14	1	1	1	1	1	1	1	1	1	1	1	1						
15	1	1	1	1	1	1	1	1	1	1	1	1						
16	1	1	1	1	1	1	1	1	1	1	1	1				0		
17	1	1	1	1	1	1	1	1	1	1	1	1						
18	1	1	1	1	1	1	1	1	1	1	1	1						

Networking-aware Placement Example

Meng et al., "Improving the scalability of data-center networks with traffic aware virtual machine placement", INFOCOM 2010

- Algorithm outline
  - Inequality-based intuition
    - If $0 \le a_1 \le \cdots \le a_n$ and $0 \le b_1 \le \cdots \le b_n$, then for any permutation π

$$\sum_{i=1}^{n} a_i b_{n-i+1} \le \sum_{i=1}^{n} a_i b_{\pi(i)} \le \sum_{i=1}^{n} a_i b_i$$

  - Meaning: map high-rate VM pairs to low-latency server pairs
  - Divide-and-conquer
    - Partition slots and VMs into equal-size clusters
      - Minimize weight of inter-cluster edges
    - Map VM *clusters* to slot *clusters* (bijection)
    - For each pair of clusters, solve problem on sub-instance

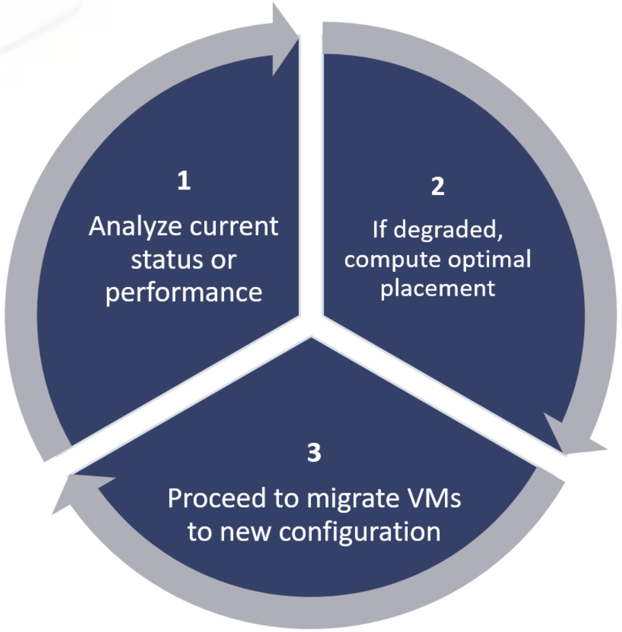
Outline

- Introduction
  - Datacenters, VM requirements and infrastructure characteristics
- VM placement
  - (Multidimensional) bin packing
  - Networking aware
- VM migration

VM Migration

- Standard placement workflow:

- VM migration workflow and considerations:
  - Time until migration is complete
    - And induced performance degradation whilst
  - Robustness of new configuration
    - Avoid frequent re-configurations / oscillation



VM Migration

- VM migration is much cheaper than physical migration
  - Copy snapshot of VM to new machine; Restart
  - Make-before-break
    - Shutdown old location once new location is ready
  - Plan for short downtime (sub-seconds)
    - Longer downtime might be unacceptable for the application
  - Even cheaper for containers...
- Careful:
  - Need to migrate network as well (virtual network configuration throughout)
    - Move/clone switches
    - Update forwarding tables

(Partial) Bibliography

- Vazirani, Approximation algorithms, Springer, 2001
- Christensen et al., “Approximation and online algorithms for multidimensional bin packing: A survey”, Comp. Sci. Rev. 24:63-79, 2017
- Bansal et al. “Improved Approximation for Vector Bin Packing”, SODA 2016
- Panigrahy et al., “Heuristics for Vector Bin Packing”, 2011
- Jennings and Stadler, “Resource Management in Clouds: Survey and Research Challenges”, J. Network Syst. Manage. 23(3): 567-619, 2015
- Usmani and Singh, “A Survey of Virtual Machine Placement Techniques in a Cloud Data Center”, Proc. Comp. Sci. 78:491-498, 2016
- Pires and Barán, “Virtual Machine Placement Literature Review”, 2015
- Li and Qian, “A survey of network function placement”, CCNC 2016
- Meng et al., "Improving the scalability of data-center networks with traffic aware virtual machine placement”, INFOCOM 2010
- Ghorbani et al., “Transparent, Live Migration of a Software-Defined Network”, SOCC 2014