Scalable Constraint-based Virtual Data Center Allocation

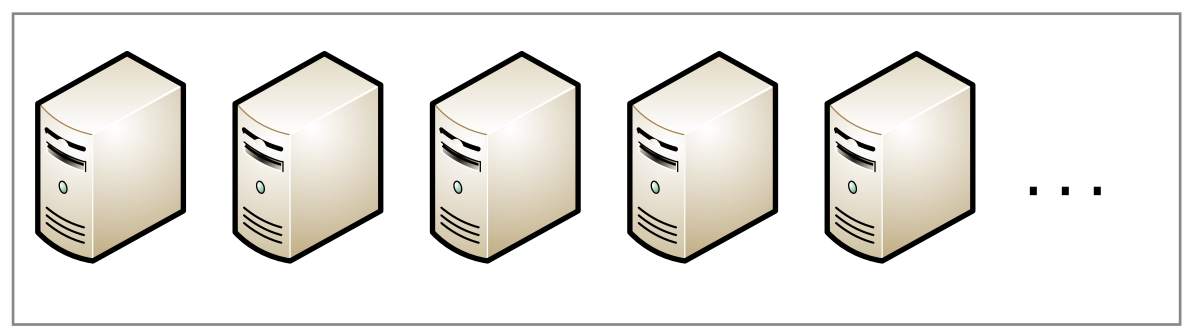
Sam Bayless
Nodir Kodirov
Ivan Beschastnikh
Holger H. Hoos
Alan J. Hu

Computer Science
University of British Columbia

Data centers, data centers, for all

Data centers, data centers, for all

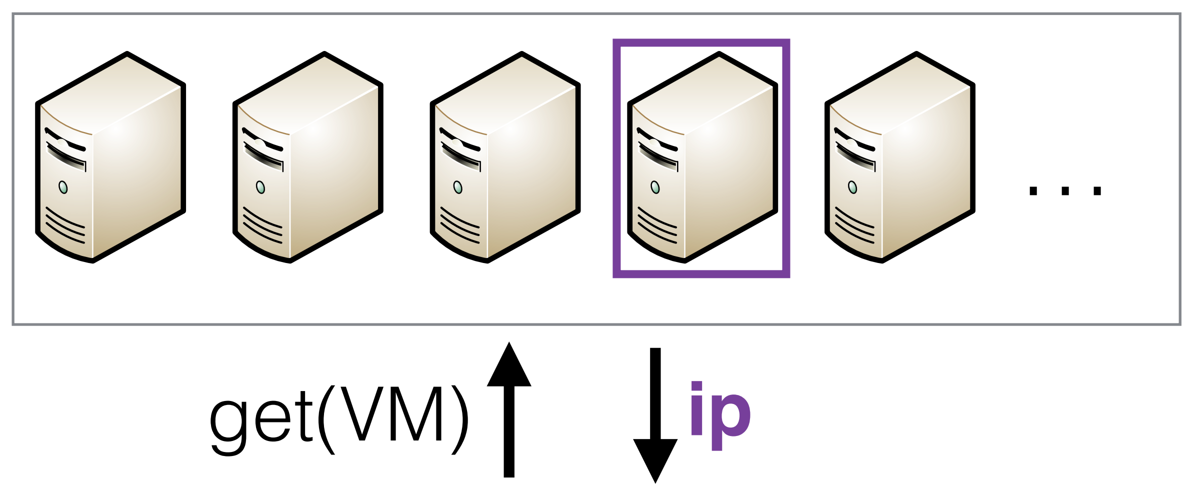
Flexible abstraction:



Provider

Data centers, data centers, for all

Flexible abstraction:



Provider

get(VM)

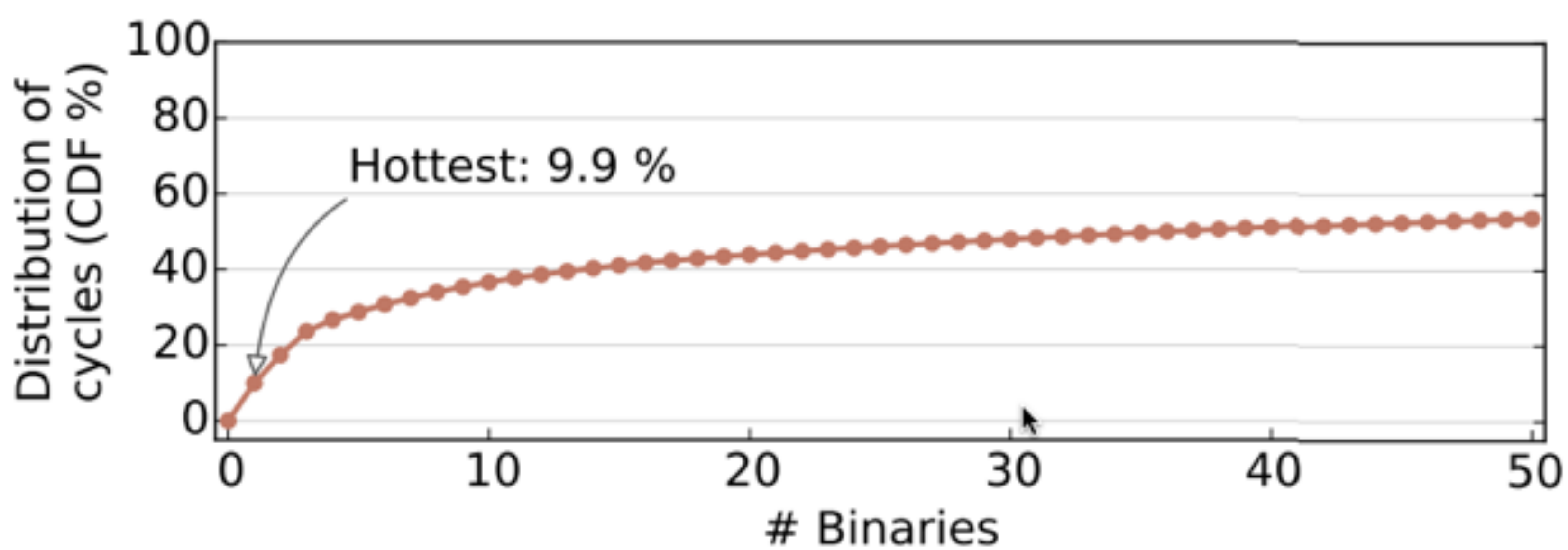
ip

Customer

Data centers, data centers, for all

- Cisco: traffic flowing through data centers will **triple** between 2014 and 2019 (reaching 10.4 ZB/year)

- Wide variety of applications being hosted [Benson et al. IMC'10, Kanev et al. ISCA'15]



More capacity + capabilities lead to more complex workloads

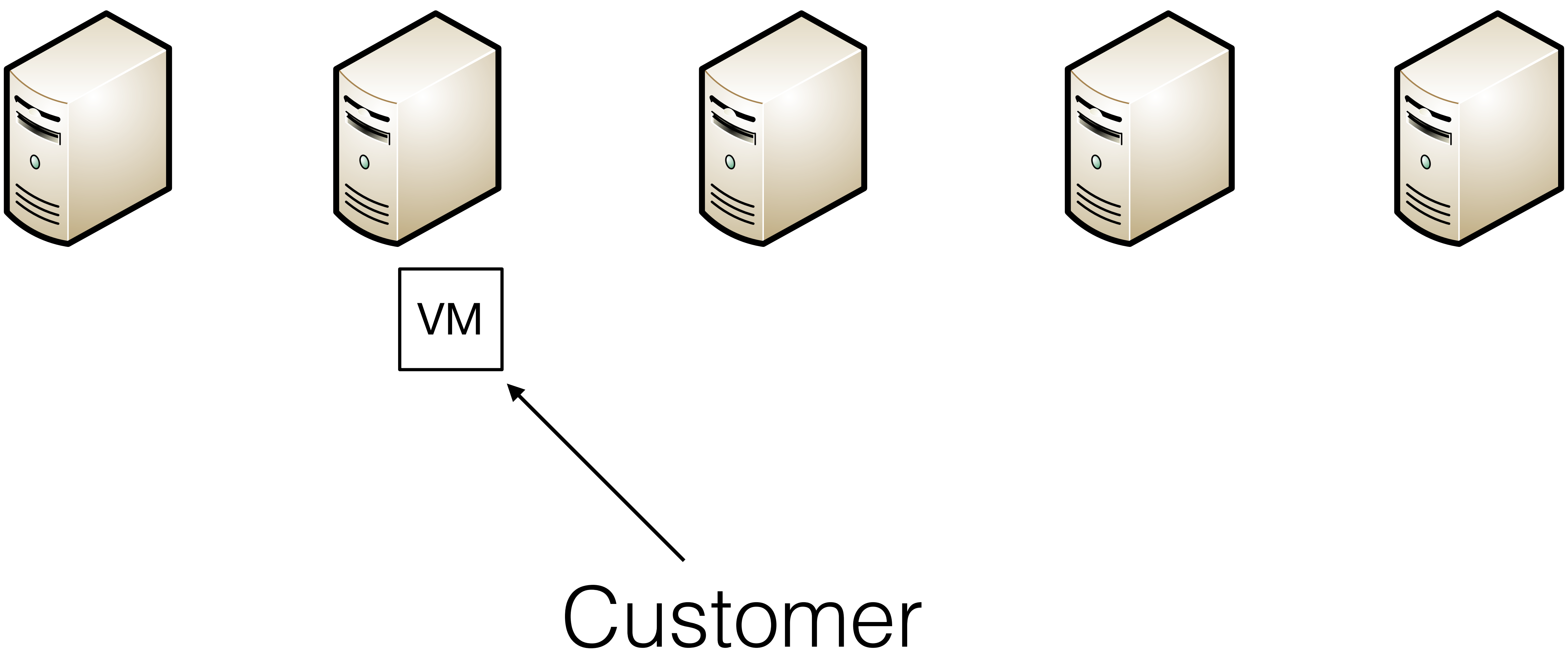
- Complex workload examples

    ➡ *Allocate a web-server, cache, database in a particular topology and with enough bandwidth to satisfy a certain QoS*

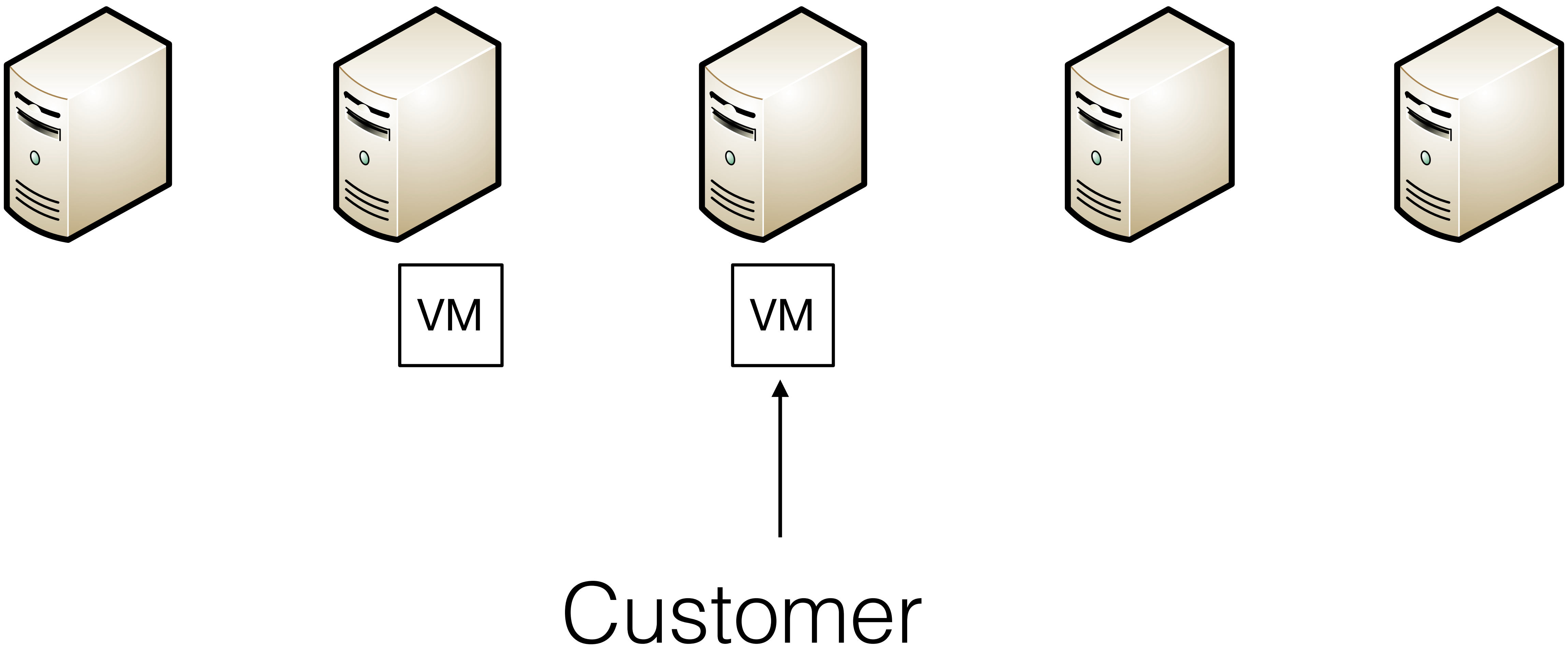
    ➡ *Deploy a distributed compute task in which some nodes communicate a lot, and others rarely*

    ➡ *Allocate a chain of NFV elements some of which require special hardware (GPUs)*

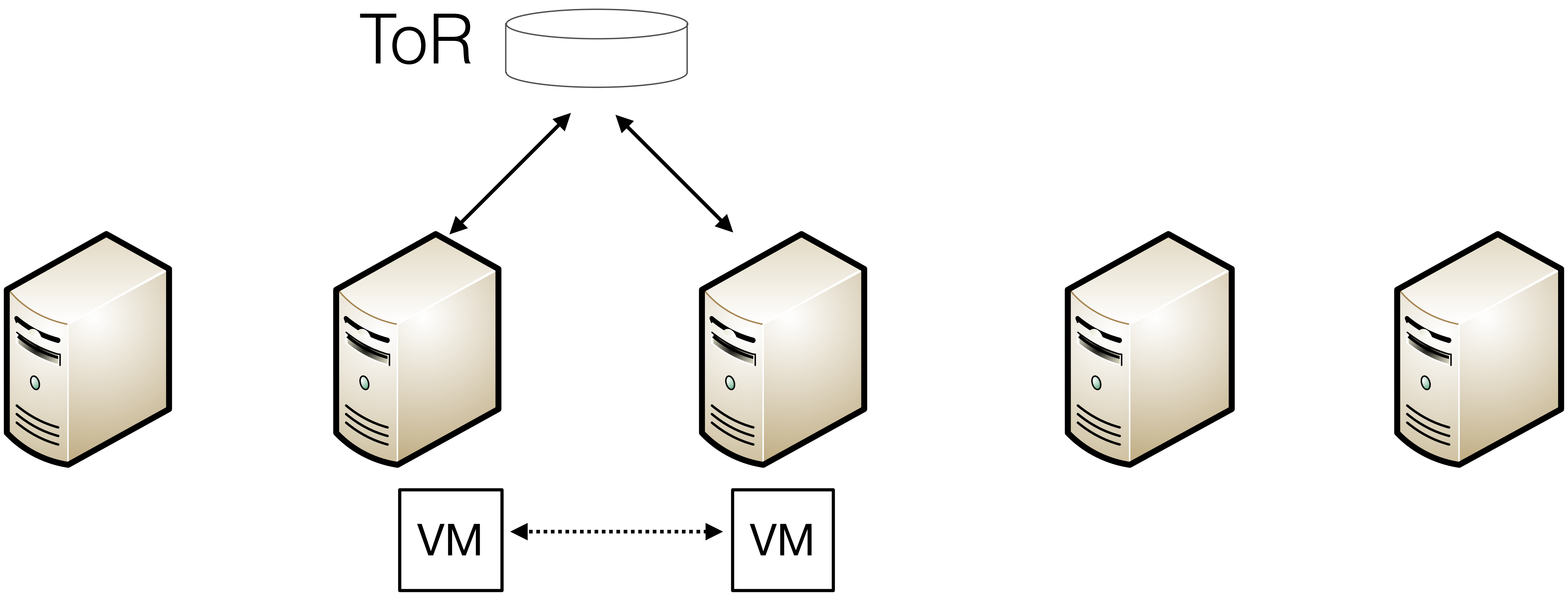
VM allocation



VM allocation

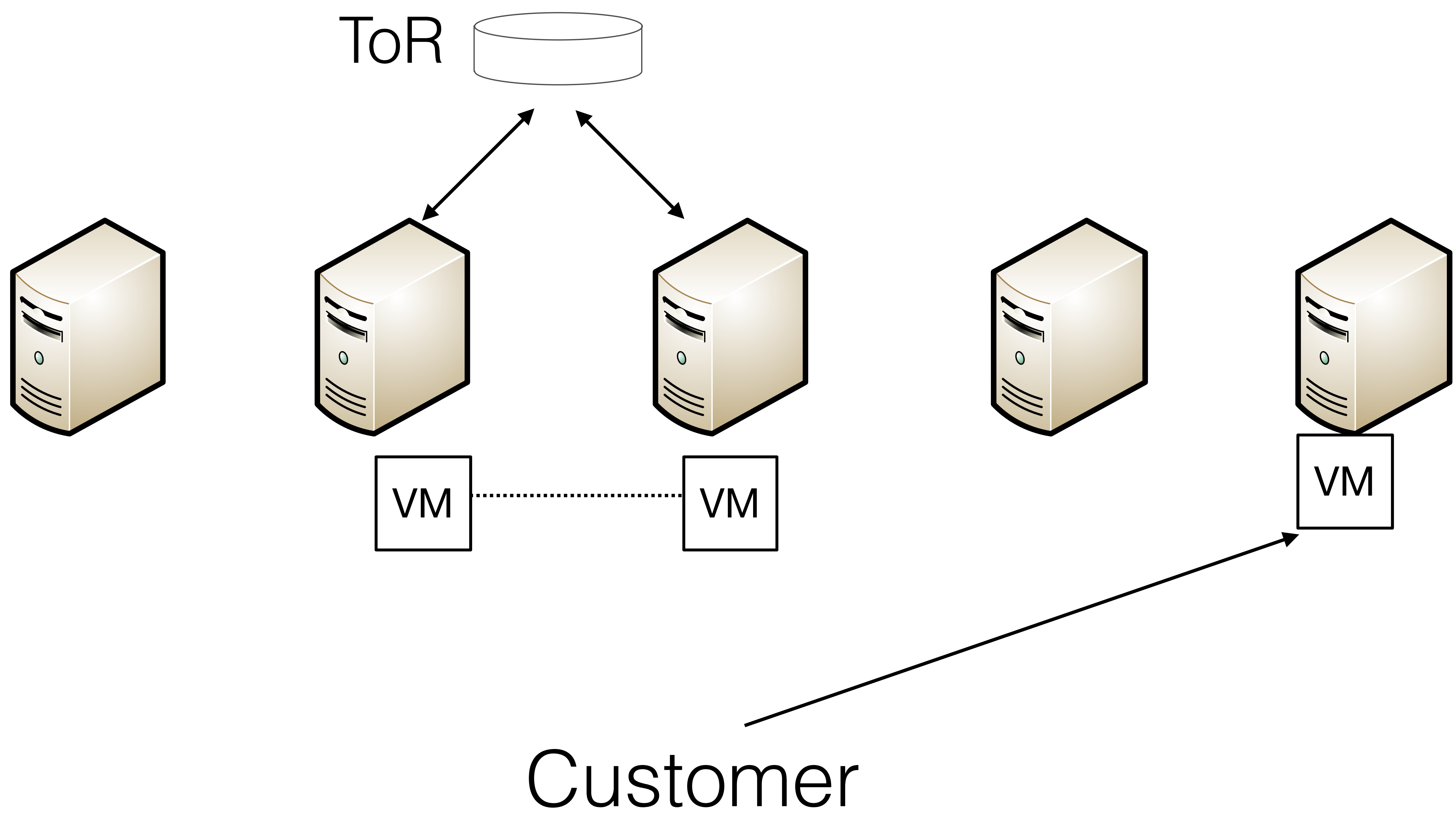


VM allocation

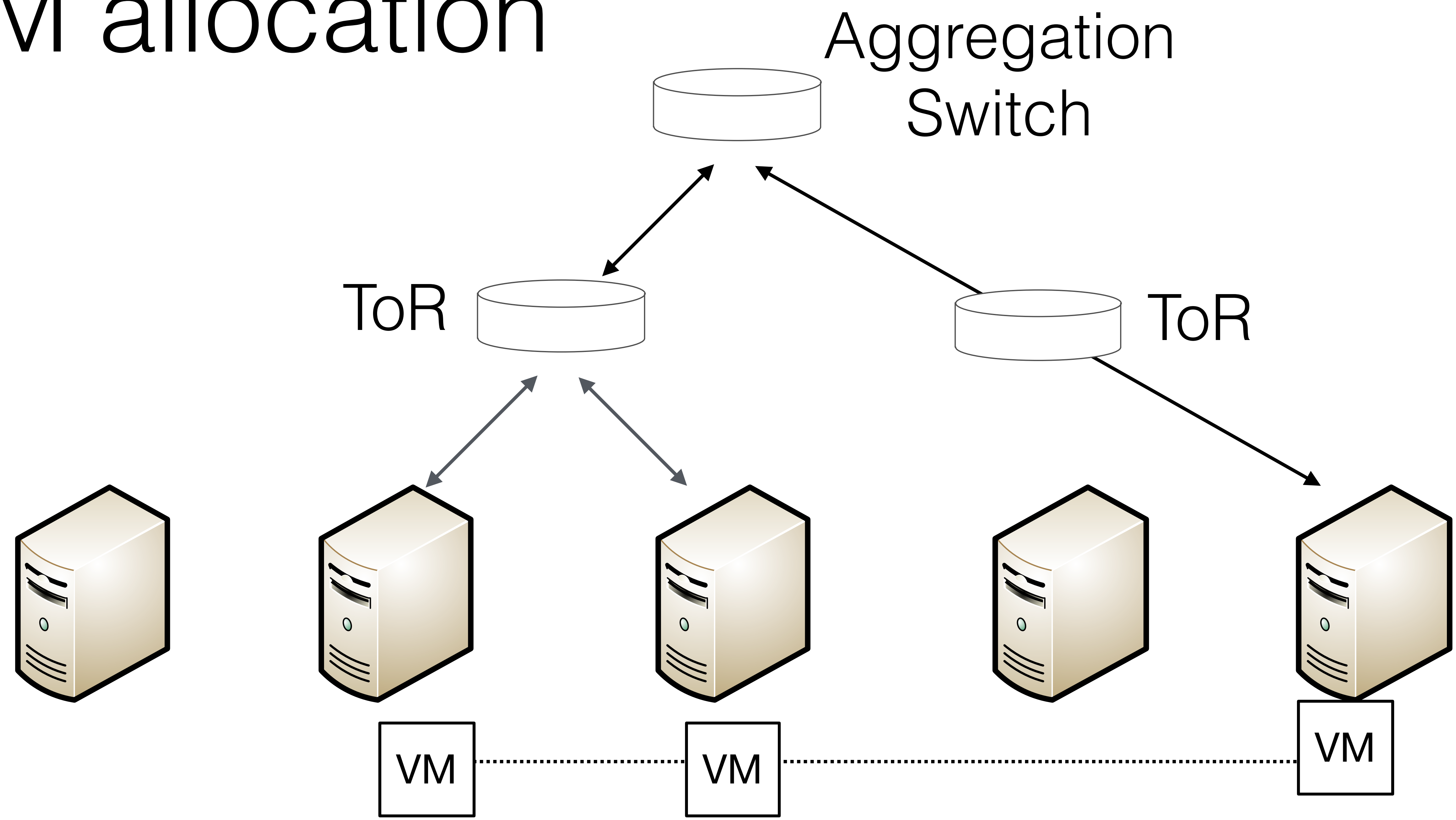


Customer

VM allocation

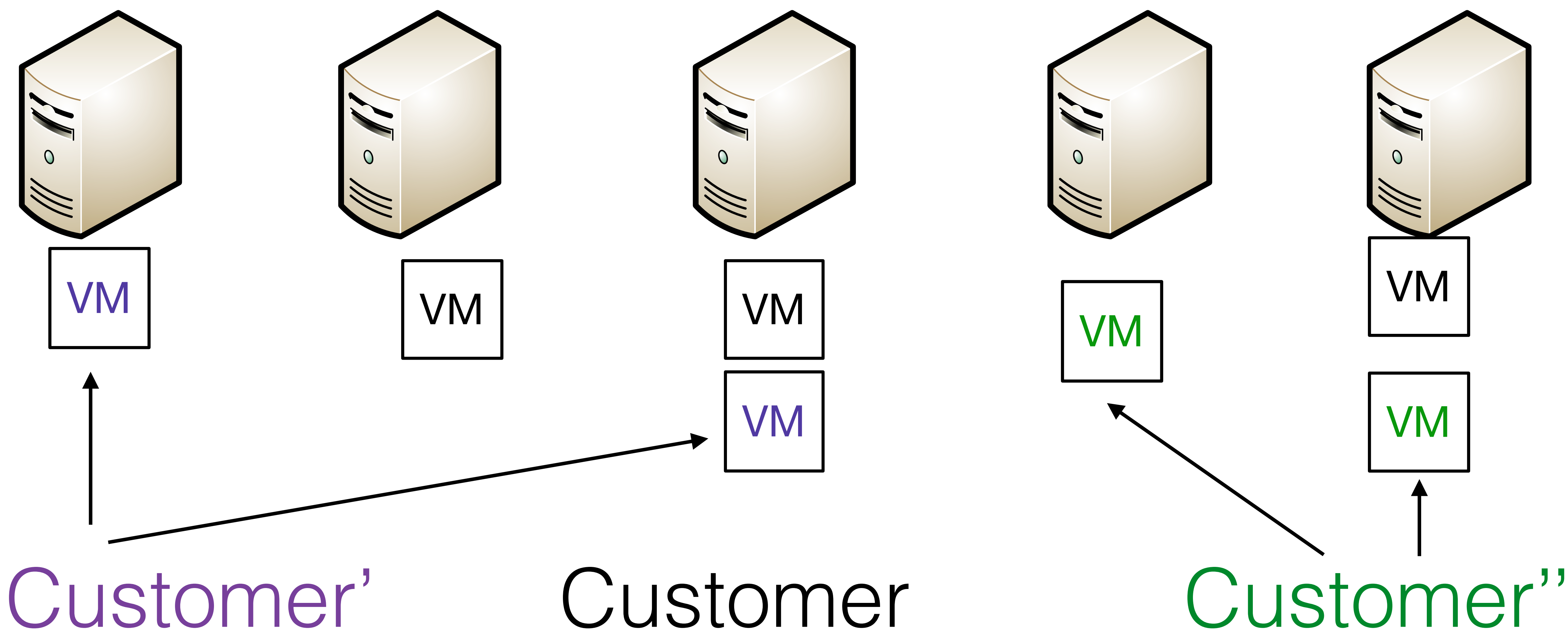


VM allocation

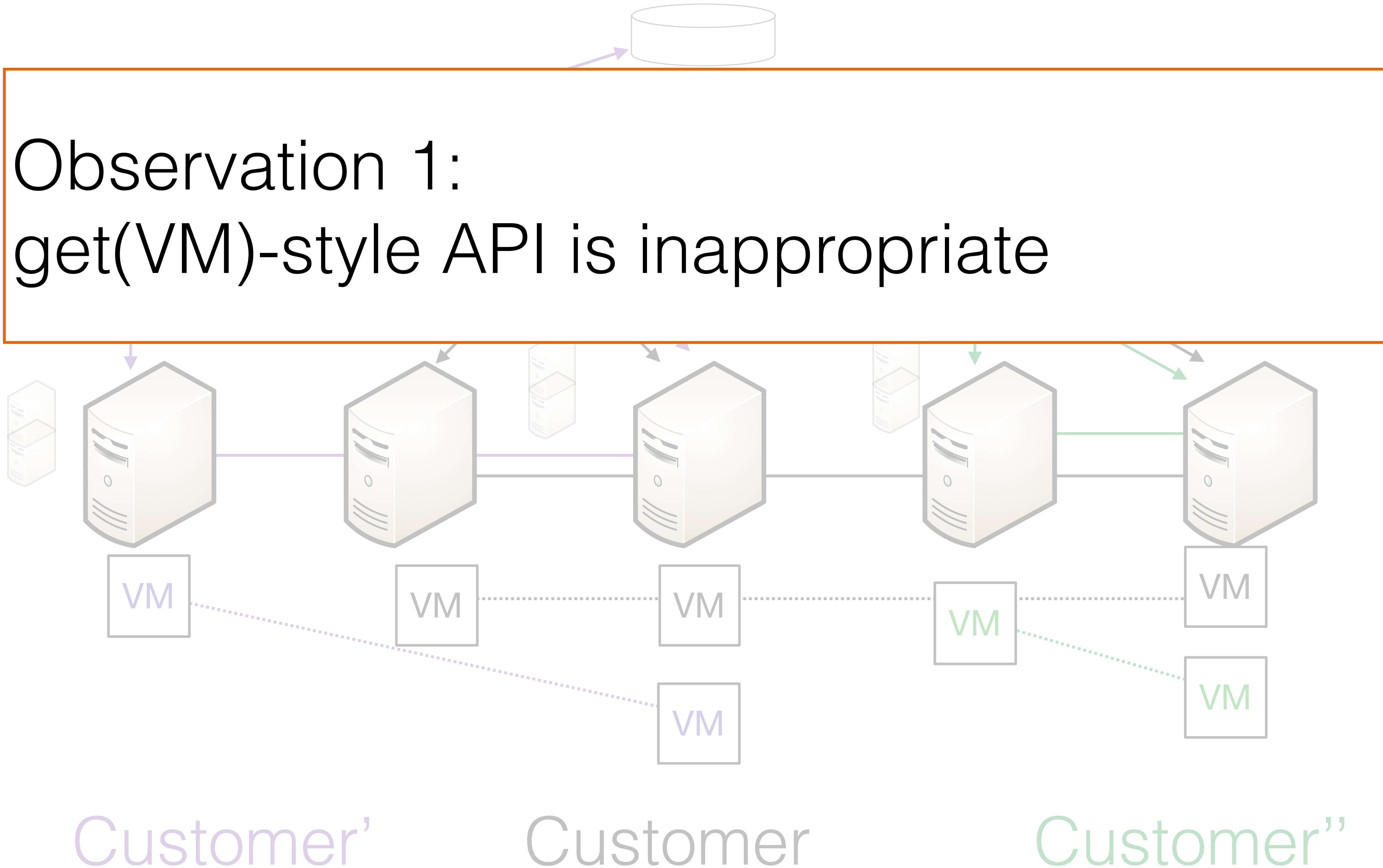


Customer

VM allocation: multi-tenancy



VM allocation



Observation 1:
get(VM)-style API is inappropriate

VM allocation

Observation 1:
get(VM)-style API is inappropriate

Observation 2:
It is more effective to allocate virtual data centers (VDCs), than virtual machines (VMs)

Customer’ Customer Customer’’

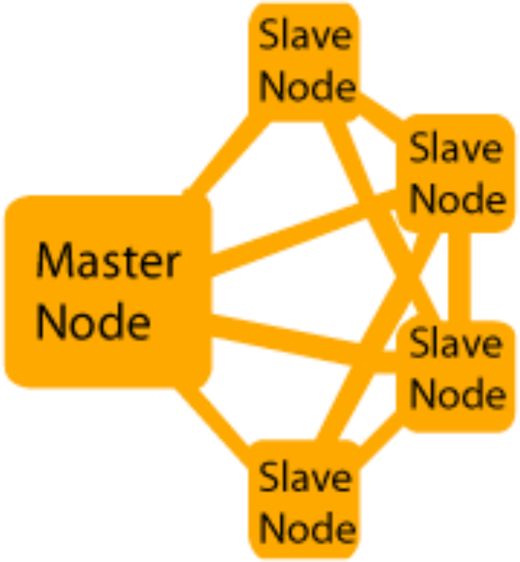
As DCs evolve, so must the programming models and allocation mechanisms

- Allocation one VM at a time: get(VM)
  - Sub-optimal for the provider <u>and</u> the customer
  - More info about an allocation: helps the provider plan and to effectively pack the data center
  - Customers benefit since they get the properties that they ultimately need

As DCs evolve, so do the programming models and allocation mechanisms

- Allocation one VM at a time
    - Sub-optimal for the provider <u>and</u> the customer
    - More info about an allocation: helps the provider plan and to effectively pack the data center
    - Customers benefit since they get the properties that they ultimately need

AWS Lambda



Amazon EMR (Hadoop)

In this talk

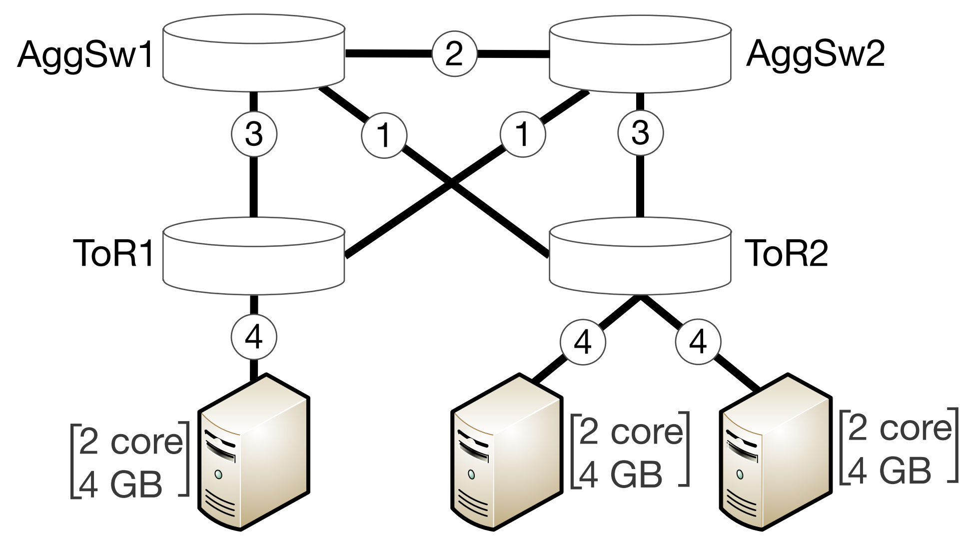
- Introduce virtual data center (VDC) allocation
- Discuss prior work (there is lots of it, mostly in the networking community)
- Describe NetSolver: our approach to solve VDC allocation (based on MonoSAT SMT-solver)
- Show how NetSolver compares to other approaches

What's the problem?

- Multi-path VDC allocation
  - Input 1: a (directed/undirected) physical DC topology (**DC**) with edge capacities/latencies and per-host constraints (disk/memory/CPU/GPU/etc)
  - Input 2: a virtual data center (**VDC**) that describe connectivity graph between VMs, and connectivity/VM requirements
  - Output: **assignment** of VMs to hosts, and virtual edges to physical paths (possibly multi-path) s.t. all constraints (end-to-end bandw, and VM) are satisfied and respect DC

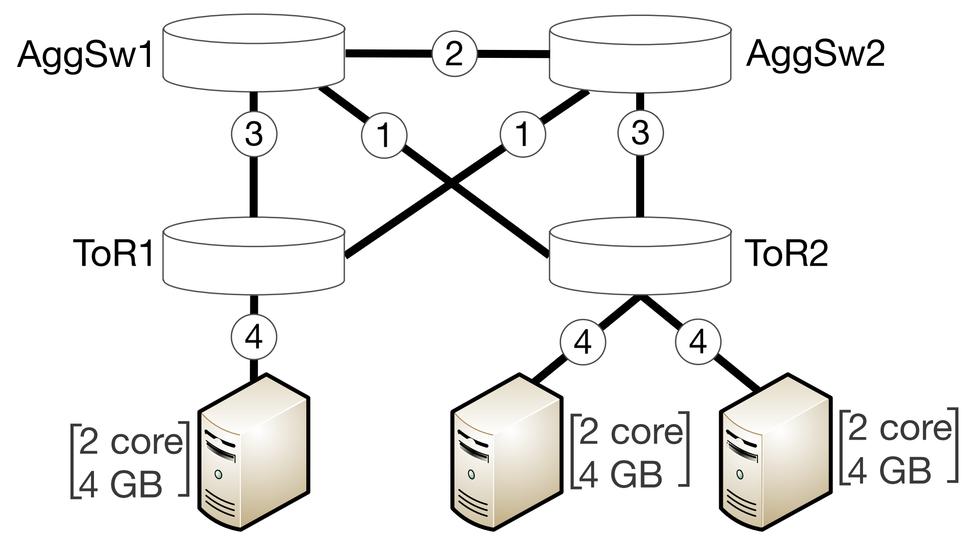
What’s the problem?

Physical DC topology:

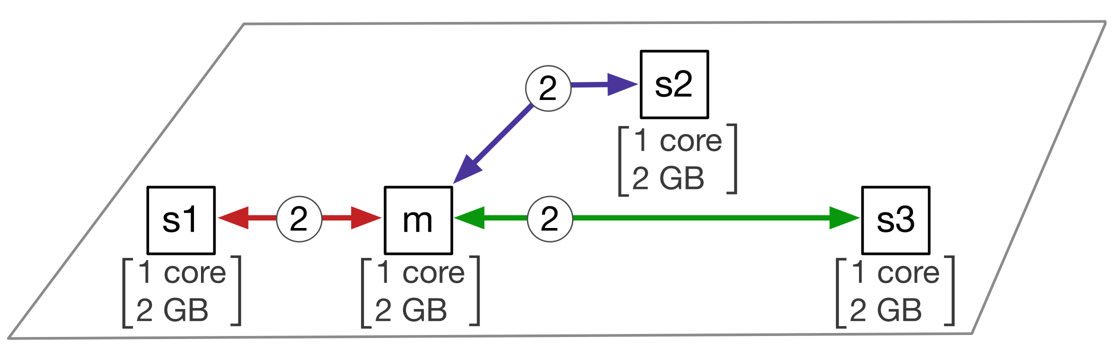


What’s the problem?

Physical DC topology:

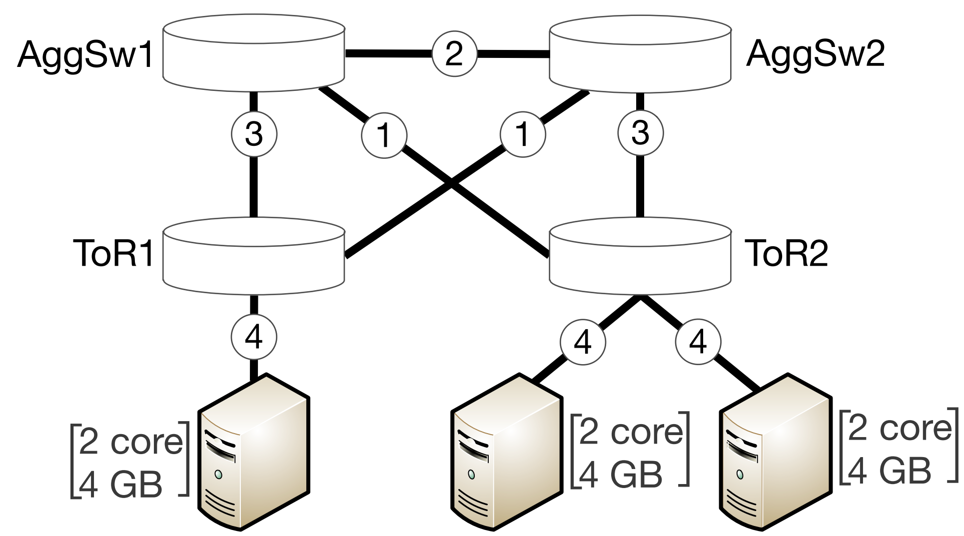


Virtual data center (VDC):



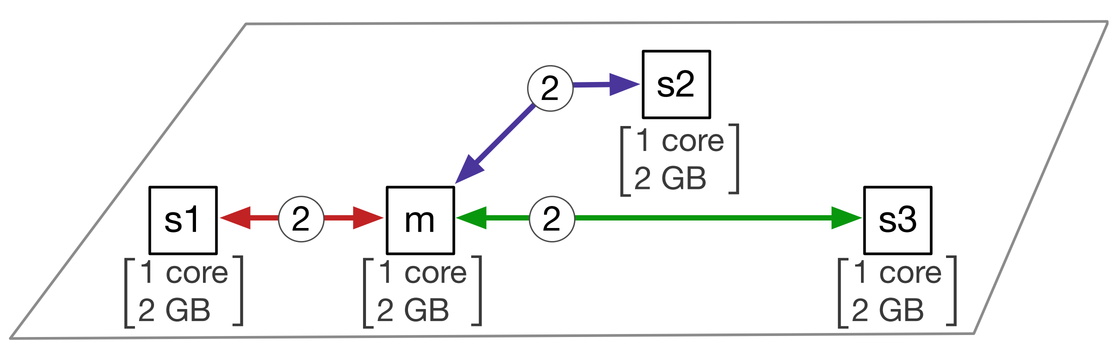
What's the problem?

Physical DC topology:



Output: **Mapping**

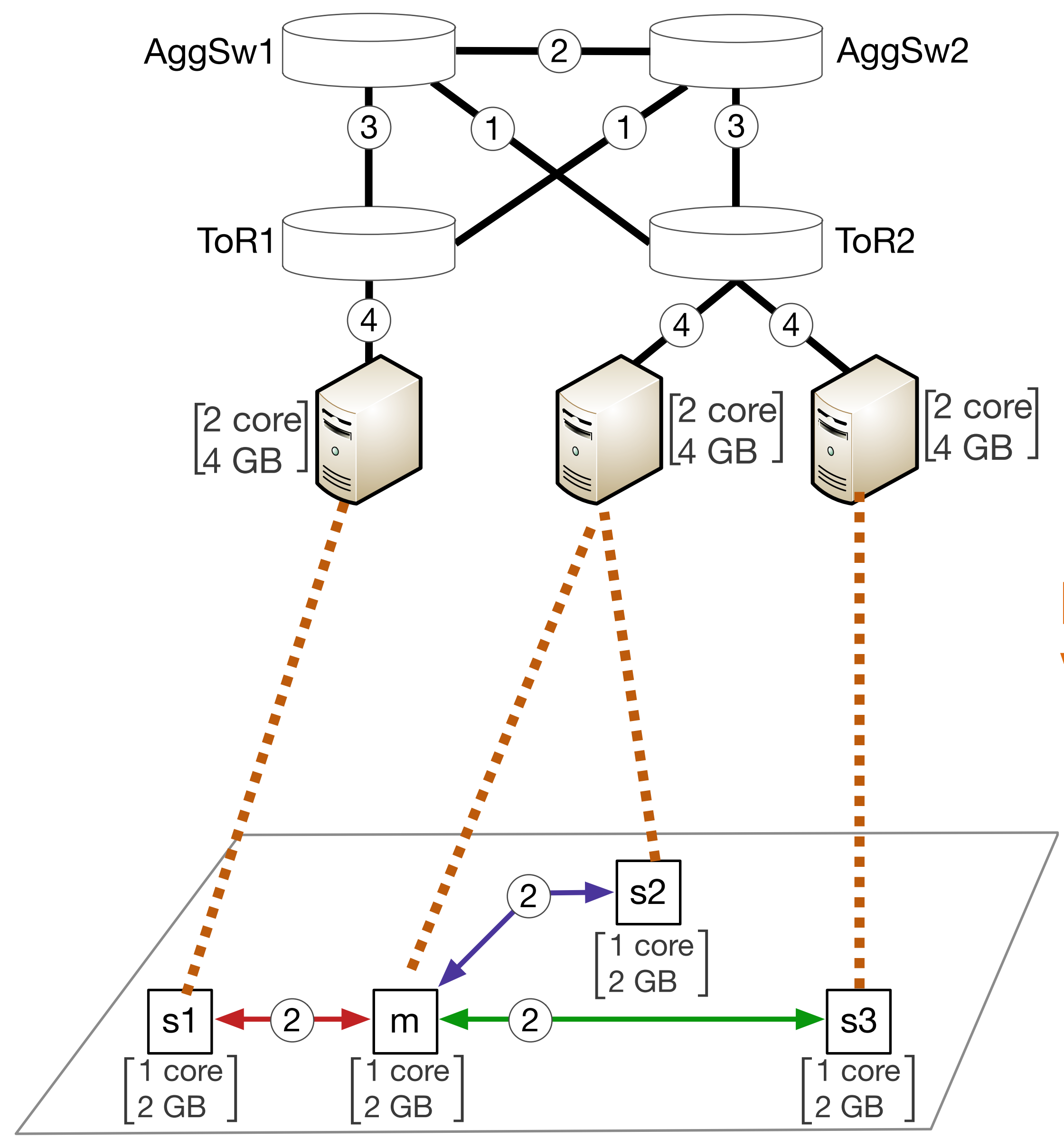
Virtual data center (VDC):



Example NetSolver solution

Physical DC topology:

Virtual data center (VDC):

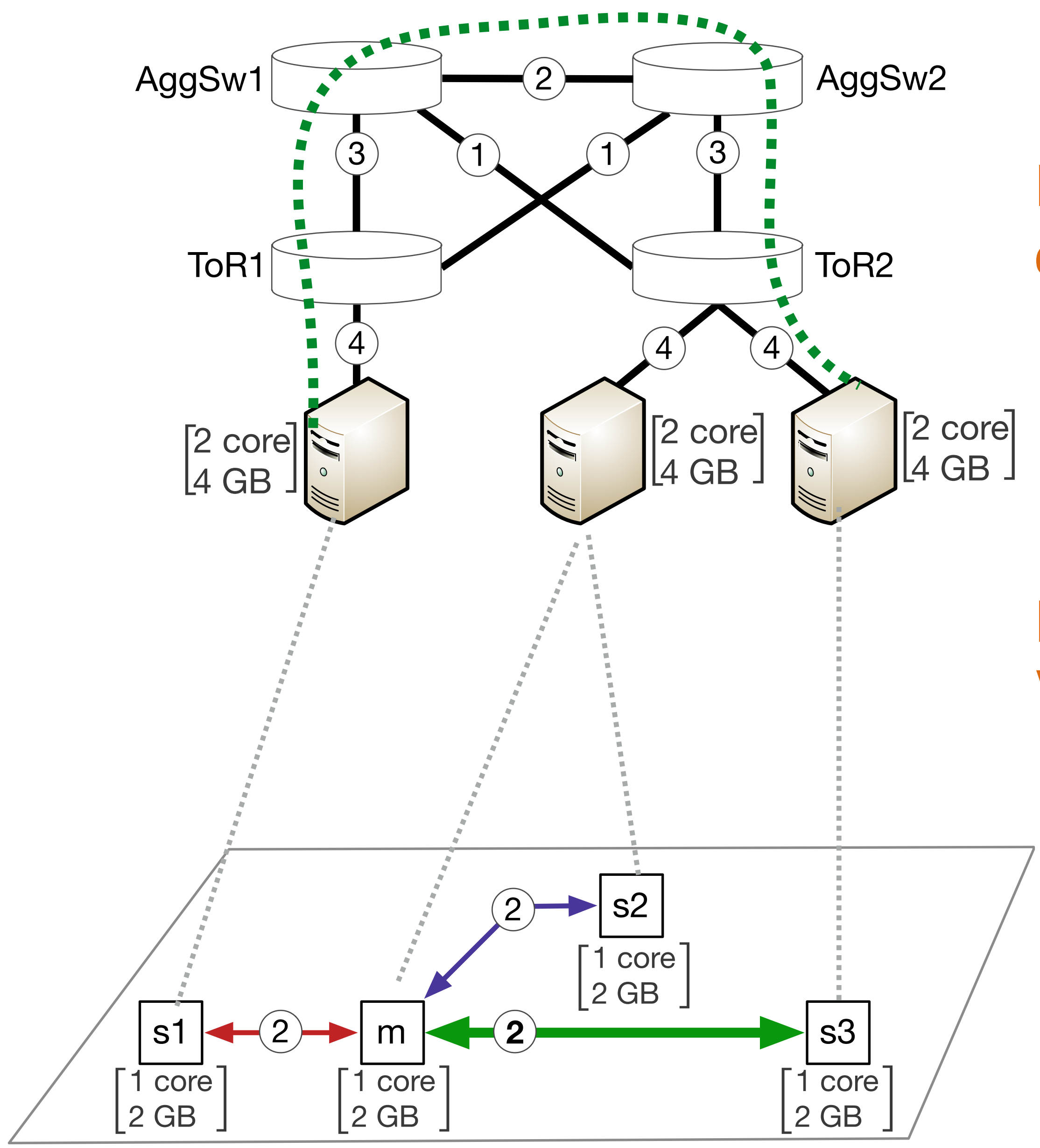


Mapping 1:
VM -> host

Example NetSolver solution

Physical DC topology:

Virtual data center (VDC):

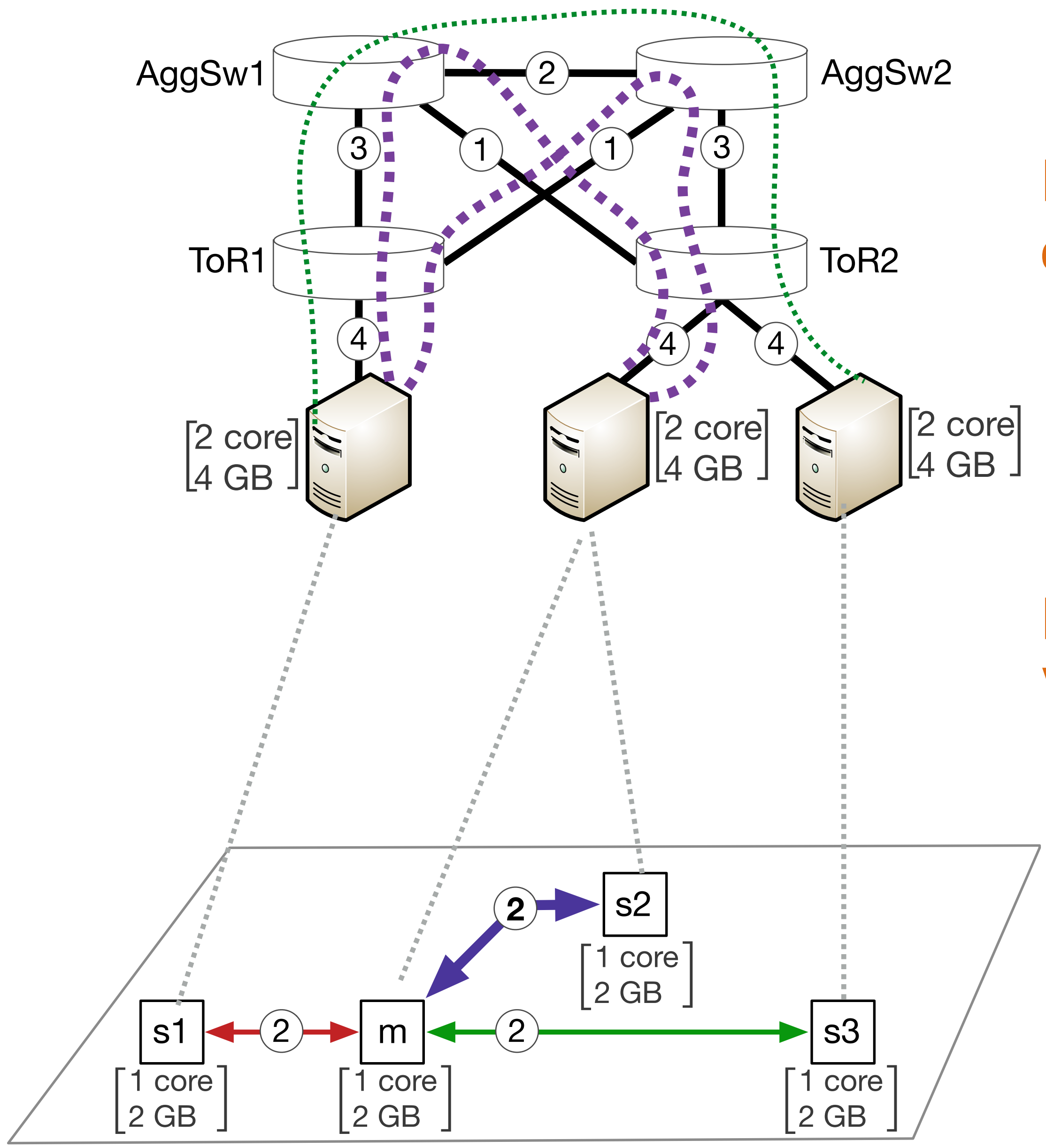


Mapping 2: VM-VM edges -> paths

Mapping 1: VM -> host

Example NetSolver solution

Physical DC topology:



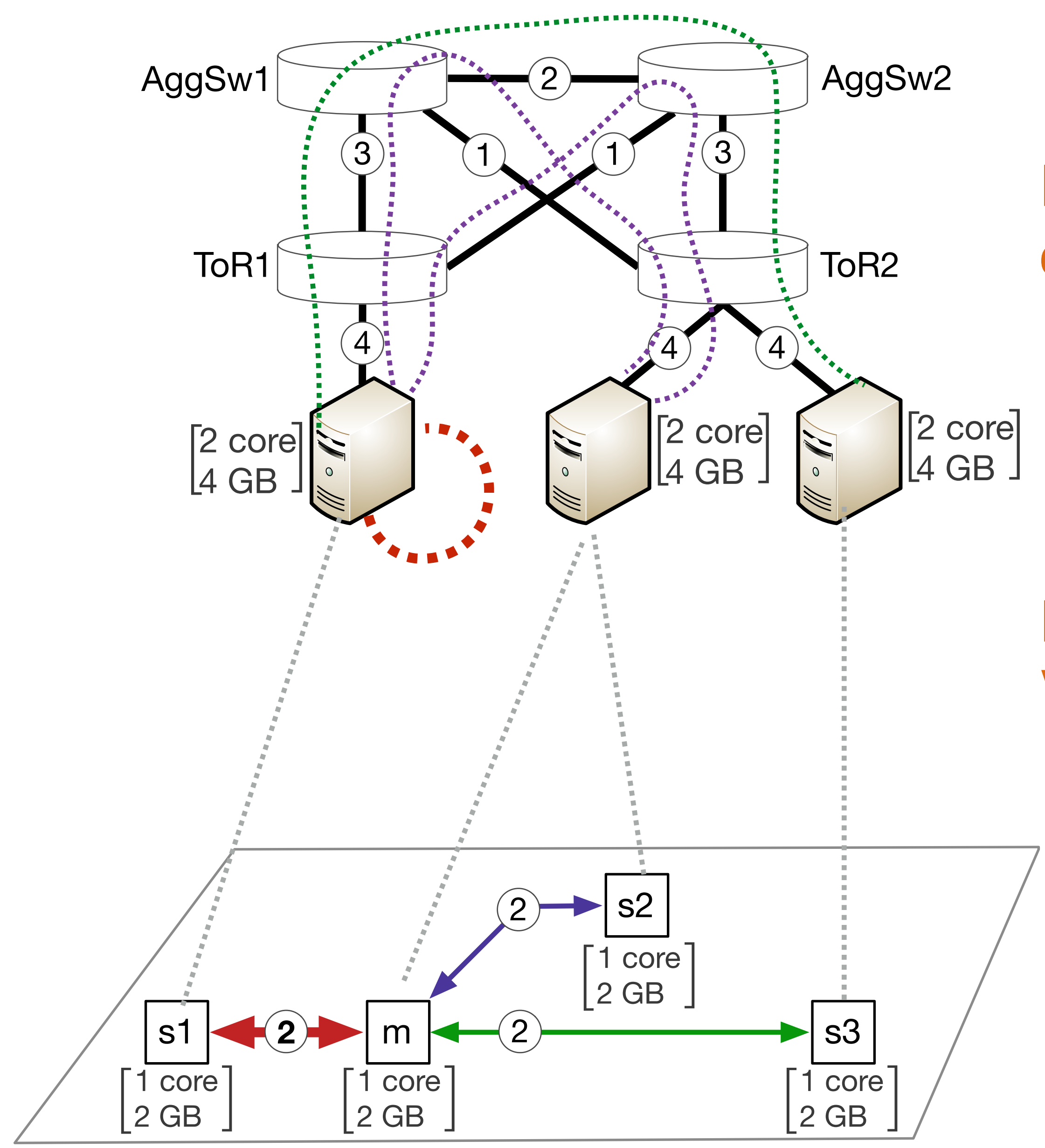
Mapping 2: VM-VM edges -> paths

Mapping 1: VM -> host

Virtual data center (VDC):

Example NetSolver solution

Physical DC topology:



Mapping 2: VM-VM edges -> paths

Mapping 1: VM -> host

Virtual data center (VDC):

Related work dimensions

- Sound: respect end-to-end bandw. guarantees
- VDC topology: Star/Hose/All
- DC topology: Tree/All
- Complete: finds a solution if a solution exists
- Multi-VM: can map more than one VM to a host
- Multi-path: supports multi-path allocations

Related work break-down

Algorithm	Sound	VDC Topology	Data Center Topology	Complete	Multi-VM	Multi-path
SecondNet [29]						
Importance Sampling [48]						
Oktopus [8]						
VDCPlanner [54]						
HVC-ACE [43]						
GAR-SP/PS [50]						
RW-MM-SP/PS [15]						
D-ViNE [16]						
ASID [36]						
VirtualRack [31]						
Z3-AR [51]						
NETSOLVER (this paper)						

29: Guo et al. CoNEXT’10
54: Zhani et al. INM’13
15: Cheng et al. CCR’11
31: Huang et al., ICC 2014
48: Tantawi, MASCOTS’12
43: Rost et al. CCR’15
16: Chowdhury et al, INFOCOM’09
8: Ballani et al. CCR’11
50: Yu et al. SIGCOMM’08
36: Lischika et al., VISA’09
51: Yuan, FMCAD’13

Related work break-down

Algorithm	Sound	VDC Topology	Data Center Topology	Complete	Multi-VM	Multi-path
SecondNet [29]						
Importance Sampling [48]						
Oktopus [8]						
VDCPlanner [54]						
HVC-ACE [43]						
GAR-SP/PS [50]						
RW-MM-SP/PS [15]						
D-ViNE [16]						
ASID [36]						
VirtualRack [31]						
Z3-AR [51]						
NETSOLVER (this paper)	✓	All	All	✓	✓	✓

29: Guo et al. CoNEXT'10

48: Tantawi, MASCOTS'12

8: Ballani et al. CCR'11

54: Zhani et al. INM'13

43: Rost et al. CCR'15

50: Yu et al. SIGCOMM'08

15: Cheng et al. CCR'11

16: Chowdhury et al, INFOCOM'09

36: Lischika et al., VISA'09

31: Huang et al., ICC 2014

51: Yuan, FMCAD'13

Related work break-down

Algorithm	Sound	VDC Topology	Data Center Topology	Complete	Multi-VM	Multi-path
SecondNet [29]	✓	All	All			
Importance Sampling [48]	✓	All	Tree		✓	
Oktopus [8]	✓	Star	All		✓	
VDCPlanner [54]	✓	All	All		✓	
HVC-ACE [43]	✓	Hose	All		✓	✓
GAR-SP/PS [50]	✓	All	$<$ 200 nodes		✓	✓
RW-MM-SP/PS [15]	✓	All	$<$ 200 nodes			✓
D-ViNE [16]	✓	All	$<$ 200 nodes			✓
ASID [36]	✓	All	$<$ 200 nodes			
VirtualRack [31]	✓	Hose	All	✓		
Z3-AR [51]	✓	All	Tree	✓	✓	
NETSOLVER (this paper)	✓	All	All	✓	✓	✓

29: Guo et al. CoNEXT'10
48: Tantawi, MASCOTS'12
8: Ballani et al. CCR'11
54: Zhani et al. INM'13
43: Rost et al. CCR'15
50: Yu et al. SIGCOMM'08
15: Cheng et al. CCR'11
16: Chowdhury et al, INFOCOM'09
36: Lischika et al., VISA'09
31: Huang et al., ICC 2014
51: Yuan, FMCAD'13

Related work break-down

Algorithm	Sound	VDC Topology	Data Center Topology	Complete	Multi-VM	Multi-path
SecondNet [29]	✓	All	All			
Importance Sampling [48]	✓	All	Tree		✓	
Oktopus [8]	✓	Star	All		✓	
VDCPlanner [54]	✓	All	All		✓	
HVC-ACE [43]	✓	Hose	All		✓	✓
GAR-SP/PS [50]	✓	All	<200 nodes		✓	✓
RW-MM-SP/PS [15]	✓	All	<200 nodes			✓
D-ViNE [16]	✓	All	<200 nodes			✓
ASID [36]	✓	All	<200 nodes			
VirtualRack [31]	✓	Hose	All	✓		
Z3-AR [51]	✓	All	Tree	✓	✓	
NETSOLVER (this paper)	✓	All	All	✓	✓	✓

29: Guo et al. CoNEXT'10

48: Tantawi, MASCOTS'12

8: Ballani et al. CCR'11

54: Zhani et al. INM'13

43: Rost et al. CCR'15

50: Yu et al. SIGCOMM'08

15: Cheng et al. CCR'11

16: Chowdhury et al, INFOCOM'09

36: Lischika et al., VISA'09

31: Huang et al., ICC 2014

51: Yuan, FMCAD'13

Related work break-down

Algorithm	Sound	VDC Topology	Data Center Topology	Complete	Multi-VM	Multi-path
SecondNet [29]	✓	All	All			
Importance Sampling [48]	✓	All	Tree		✓	
Oktopus [8]	✓	Star	All		✓	
VDCPlanner [54]	✓	All	All		✓	
HVC-ACE [43]	✓	Hose	All		✓	✓
GAR-SP/PS [50]	✓	All	< 200 nodes		✓	✓
RW-MM-SP/PS [15]	✓	All	< 200 nodes			✓
D-ViNE [16]	✓	All	< 200 nodes			✓
ASID [36]	✓	All	< 200 nodes			
VirtualRack [31]	✓	Hose	All	✓		
Z3-AR [51]	✓	All	Tree	✓	✓	
NETSOLVER (this paper)	✓	All	All	✓	✓	✓

29: Guo et al. CoNEXT'10
48: Tantawi, MASCOTS'12
8: Ballani et al. CCR'11
54: Zhani et al. INM'13
43: Rost et al. CCR'15
50: Yu et al. SIGCOMM'08
15: Cheng et al. CCR'11
16: Chowdhury et al, INFOCOM'09
36: Lischika et al., VISA'09
31: Huang et al., ICC 2014
51: Yuan, FMCAD'13

Related work break-down

Algorithm	Sound	VDC Topology	Data Center Topology	Complete	Multi-VM	Multi-path
SecondNet [29]	✓	All	All			
Importance Sampling [48]	✓	All	Tree		✓	
Oktopus [8]	✓	Star	All		✓	
VDCPlanner [54]	✓	All	All		✓	
HVC-ACE [43]	✓	Hose	All		✓	✓
GAR-SP/PS [50]	✓	All	< 200 nodes		✓	✓
RW-MM-SP/PS [15]	✓	All	< 200 nodes			✓
D-ViNE [16]	✓	All	< 200 nodes			✓
ASID [36]	✓	All	< 200 nodes			
VirtualRack [31]	✓	Hose	All	✓		
Z3-AR [51]	✓	All	Tree	✓	✓	
NETSOLVER (this paper)	✓	All	All	✓	✓	✓

29: Guo et al. CoNEXT'10
54: Zhani et al. INM'13
15: Cheng et al. CCR'11
31: Huang et al., ICC 2014
48: Tantawi, MASCOTS'12
43: Rost et al. CCR'15
16: Chowdhury et al, INFOCOM'09
8: Ballani et al. CCR'11
50: Yu et al. SIGCOMM'08
36: Lischika et al., VISA'09
51: Yuan, FMCAD'13

MonoSAT background

- Switch to other slide-deck

The MonoSAT constraint solver

MonoSAT is an SMT solver for monotonic theories.
MonoSAT supports:

- Graph constraints (shortest paths, maximum flows...)
- Finite state machines & string acceptance
- Temporal logic (CTL) synthesis
- 2D polygonal geometry constraints
- Bounded integer & cardinality constraints
- Propositional logic (Boolean satisfiability)
- Has C++, Python, and Java bindings

Finite Monotonic Predicates

A predicate p is positive monotonic iff:

- : $p(\ldots, x, \ldots), x \leq y \implies p(\ldots, y, \ldots)$

A predicate p is negative monotonic iff:

- : $\neg p(\ldots, x, \ldots), x \leq y \implies \neg p(\ldots, y, \ldots)$

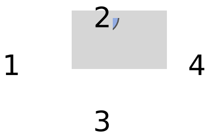
Monotonic theories

Many useful predicates are monotonic:

- Graph Predicates:
  - Reachability
  - Shortest paths
  - Maximum $s - t$ flow
  - Minimum Spanning Tree
  - Acyclicity

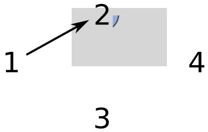
Monotonic predicates

'Reachability' is monotonic with respect to edges:



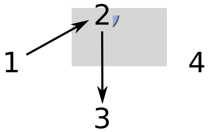
Monotonic predicates

'Reachability' is monotonic with respect to edges:



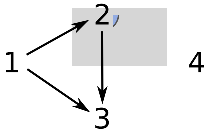
Monotonic predicates

'Reachability' is monotonic with respect to edges:



Monotonic predicates

'Reachability' is monotonic with respect to edges:

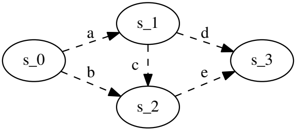


Graph constraints in MonoSAT

MonoSAT supports constraints over one or more finite graphs:

- Combines arbitrary Boolean constraints with high performance graph constraints.
- Supported graph constraints
  - Reachability
  - Shortest paths
  - Maximum *s*-*t* flow
  - Minimum spanning tree
  - Acyclicity
- Graphs can be directed
- Edges can have bit vector weights/capacities
- Scales to 100,000s of nodes and edges

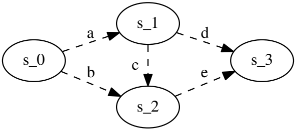
Graph constraints in MonoSAT



$$(\neg a \vee \neg b) \wedge (\neg d \vee \neg e) \wedge reaches(s_0, s_3) \wedge \neg reaches(s_1, s_3)$$

Figure : A directed graph with edge inclusion controlled by Booleans $\{a, b, c, d, e\}$, and a formula constraining the graph.

Graph constraints in MonoSAT



$$(\neg a \vee \neg b) \wedge (\neg d \vee \neg e) \wedge reaches(s_0, s_3) \wedge \neg reaches(s_1, s_3)$$

Figure : A directed graph with edge inclusion controlled by Booleans $\{a, b, c, d, e\}$, and a formula constraining the graph.

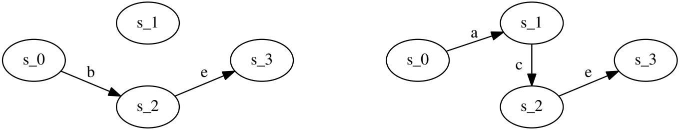
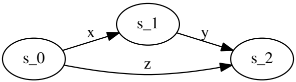


Figure : Satisfying (left) and unsatisfying (right) solutions.

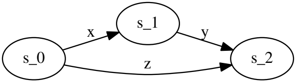
Weighted graph constraints in MONOSAT



$$(x > 1) \wedge (x < y) \wedge (y < 4) \wedge (z = y) \wedge (shortestPath(s_0, s_2) \leq 3)$$

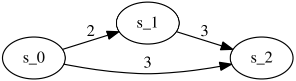
Figure : A directed graph with variable edge weights, and a formula constraining those weights.

Weighted graph constraints in MonoSAT



$$(x > 1) \wedge (x < y) \wedge (y < 4) \wedge (z = y) \wedge (shortestPath(s_0, s_2) \leq 3)$$

Figure : A directed graph with variable edge weights, and a formula constraining those weights.



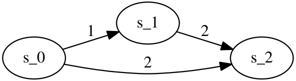
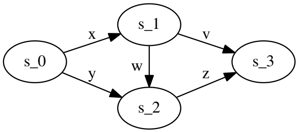


Figure : Satisfying (left) and unsatisfying (right) solutions.

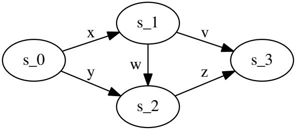
Maximum-flow graph constraints in MONOSAT



$$(x \leq z \leq 2) \wedge (x > y) \wedge (z > v) \wedge (2 \leq \mathit{maximumFlow}(s_0, s_2) \leq 3)$$

Figure : A directed graph with variable edge weights, and a formula constraining those weights.

Maximum-flow graph constraints in MONOSAT



$$(x \leq z \leq 2) \wedge (x > y) \wedge (z > v) \wedge (2 \leq \mathit{maximumFlow}(s_0, s_2) \leq 3)$$

Figure : A directed graph with variable edge weights, and a formula constraining those weights.

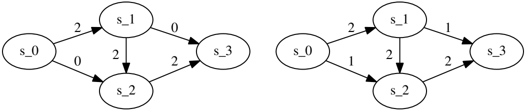
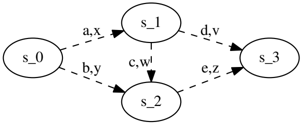


Figure : Two satisfying solutions.

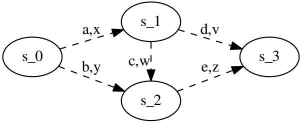
Combined graph constraints in MONOSAT



$$\neg reaches(s_0, s_2) \land shortestPath(s_2, s_3) = maximumFlow(s_0, s_3)$$

Figure : A graph with edge inclusion controlled by Booleans $\{a, b, c, d, e\}$, and edge weights $\{v, w, x, y, z\}$.

Combined graph constraints in MonoSAT



$$\neg reaches(s_0, s_2) \wedge shortestPath(s_2, s_3) = maximumFlow(s_0, s_3)$$

Figure : A graph with edge inclusion controlled by Booleans $\{a, b, c, d, e\}$, and edge weights $\{v, w, x, y, z\}$.

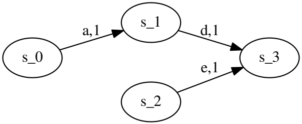


Figure : A satisfying solution.

NetSolver design

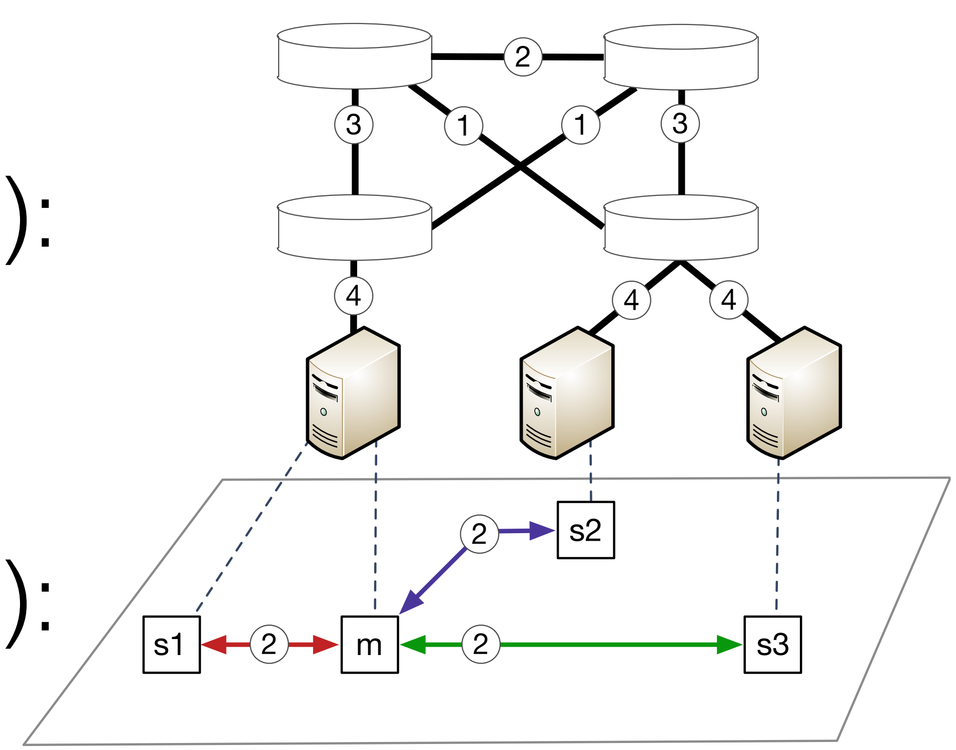
- Basic idea: encode VDC allocation as a MonoSAT query. Either outputs a solution or one does not exist
    - Global constraints: connectivity and bandwidth
    - Local constraints: VMs respect host resources
- <u>Challenge</u>: efficiency (e.g., each VM-VM path can be modeled as a max-flow constraint, these are expensive)

Global constraints

- **Assume**: that we know the VM-host assignments
- Given:
    - Directed graph $G = (V, E)$ and integer constraints $c(u, v)$ on each edge $(u, v) \in E$
    - K commodity demands, $i \in K, i = (s_i, t_i, d_i)$ representing demand d_i between $s_i \in V$ and $t_i \in V$

Graph (DC):

Demands (VDC):



Global constraints

- **Assume**: that we know the VM to host assignment
- Given:
    - Directed graph $G = (V, E)$ and integer constraints $c(u, v)$ on each edge $(u, v) \in E$
    - K commodity demands, $i \in K, i = (s_i, t_i, d_i)$ representing demand d_i between $s_i \in V$ and $t_i \in V$
- Integral multi-commodity flow problem:
    - Find feasible flow such that each d_i satisfied
    - For each edge (u, v) total flow of all capacities is $\leq c(u, v)$

Commodity flow encoding

- Create graphs $G_{1...|K|}$: one per demand with same topology as G
- For each edge $(u,v)_i \in G_i$ create a new **symbolic** capacity $c(u,v)_i \leq c(u,v)$
- **Assert:** that $\sum_i c(u,v)_i \leq c(u,v)$
- **Assert:** for each demand $i = (s_i, t_i, d_i), \text{max-flow}(s_i, t_i) \geq d$
- Solver's task: find partitioning of capacities across K graphs while satisfying lower-bounds across all demands

Modeling local constraints

- Construct a graph G that is the VDC and one node for each VM
- For each VM v and each server $s \in G$, create directed symbolic edge e_{vs} with unlimited capacity; e_{vs} controls allocation of v to servers
- **Assert**: for each VM v, exactly on e_{vs} enabled
- **Assert**: for each server s, set of VMs assigned to s obey server's local resources
- **Assert**: G satisfies flow (s_i, t_i, d_i) for each commodity constraint

Further technical innovations

- Naive encoding slow: $|V|^2$ max-flow constraints in worst case. Optimize by merging demands from same source

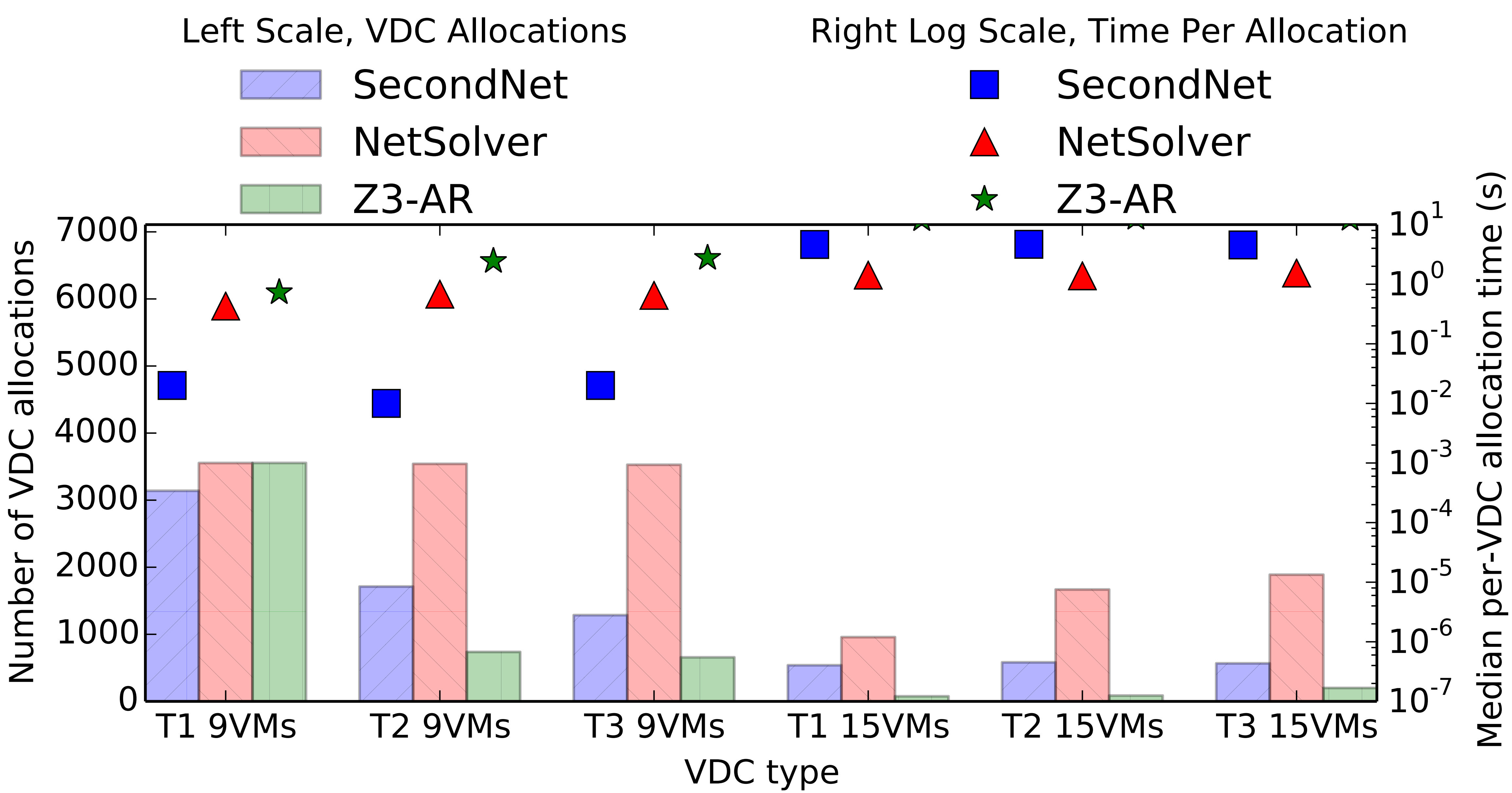
- So far assumed that VDC topology constant: only works for allocating sequence of identical VDCs

  - To allocate diverse VDCs, encode superset of VDCs and use MonoSAT's assumption mechanism to disable parts of this superset during allocation

NetSolver Evaluation

- Key questions:
  - Can a sound+complete scale to realistic topologies?
  - Are there any practical benefits to being complete?
  - How does NetSolver compare to related work?
    - SecondNet (CoNEXT’10)
    - Z3-based abstraction refinement technique (FMCAD’13)

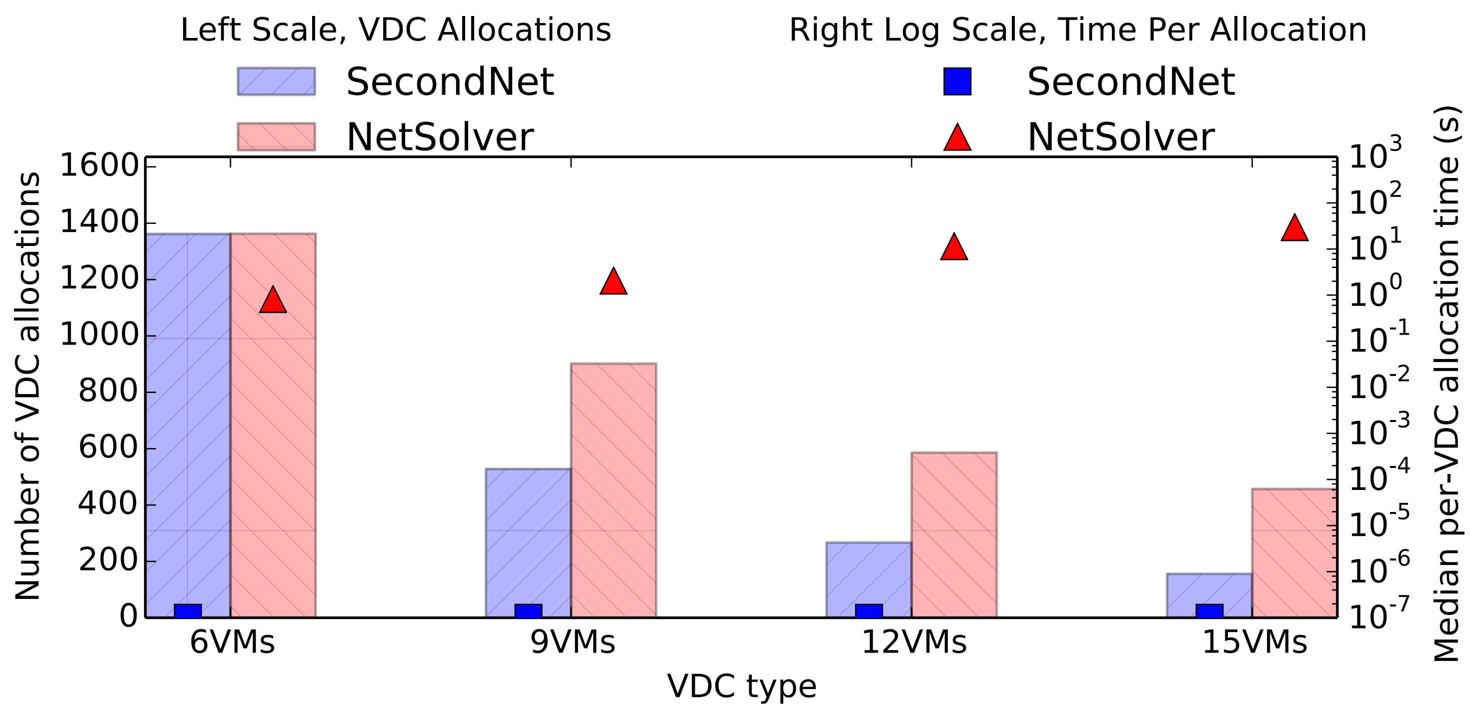
Identical VDC packing and median alloc runtime (Tree)



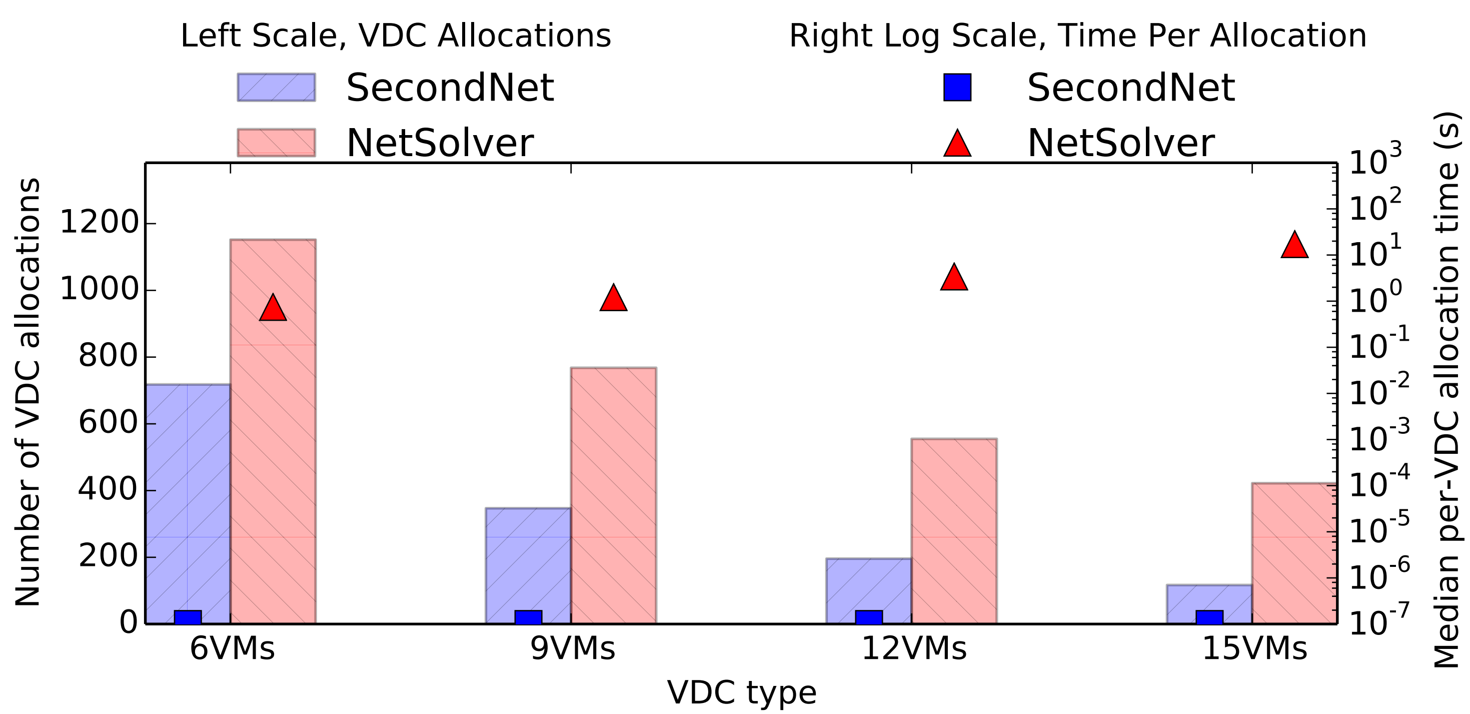
2000 servers, 16 cores; varying VDC sizes; Tree DC topologies

Identical VDC packing and median alloc runtime (BCube/FatTree)

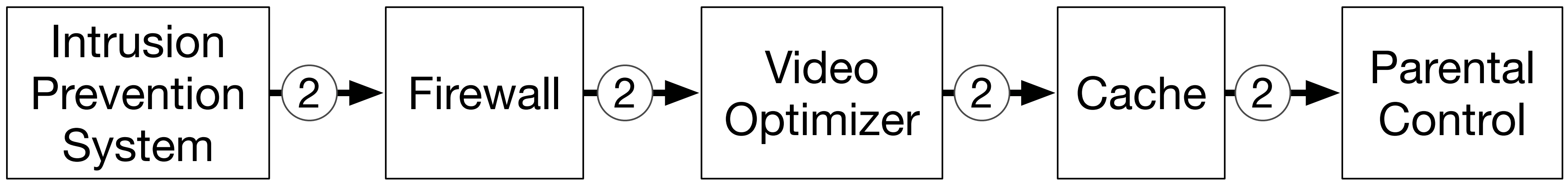
512 servers, 16 cores; varying VDC sizes; BCube DC topologies



432 servers, 16 cores; varying VDC sizes; FatTree DC topologies

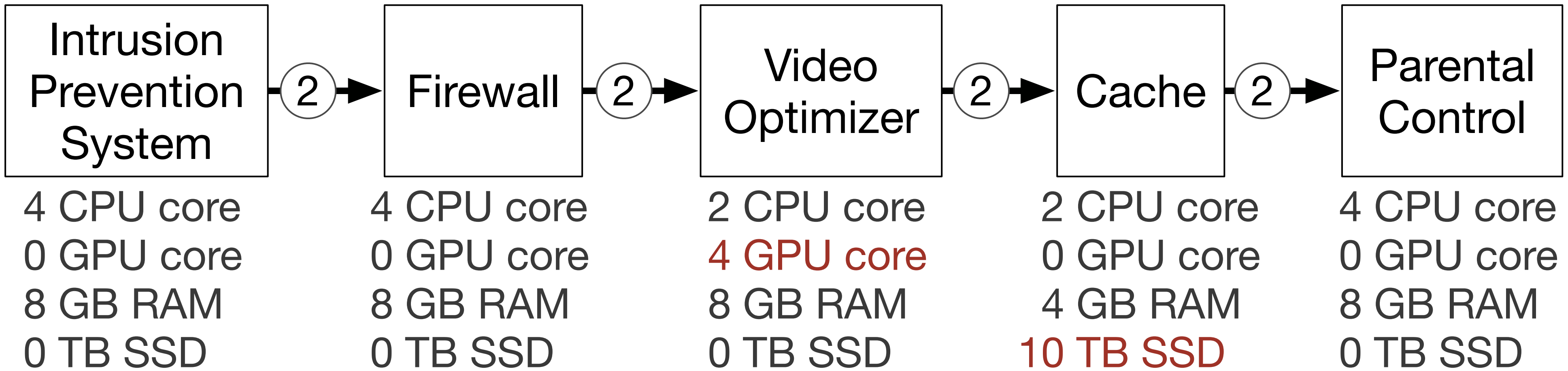


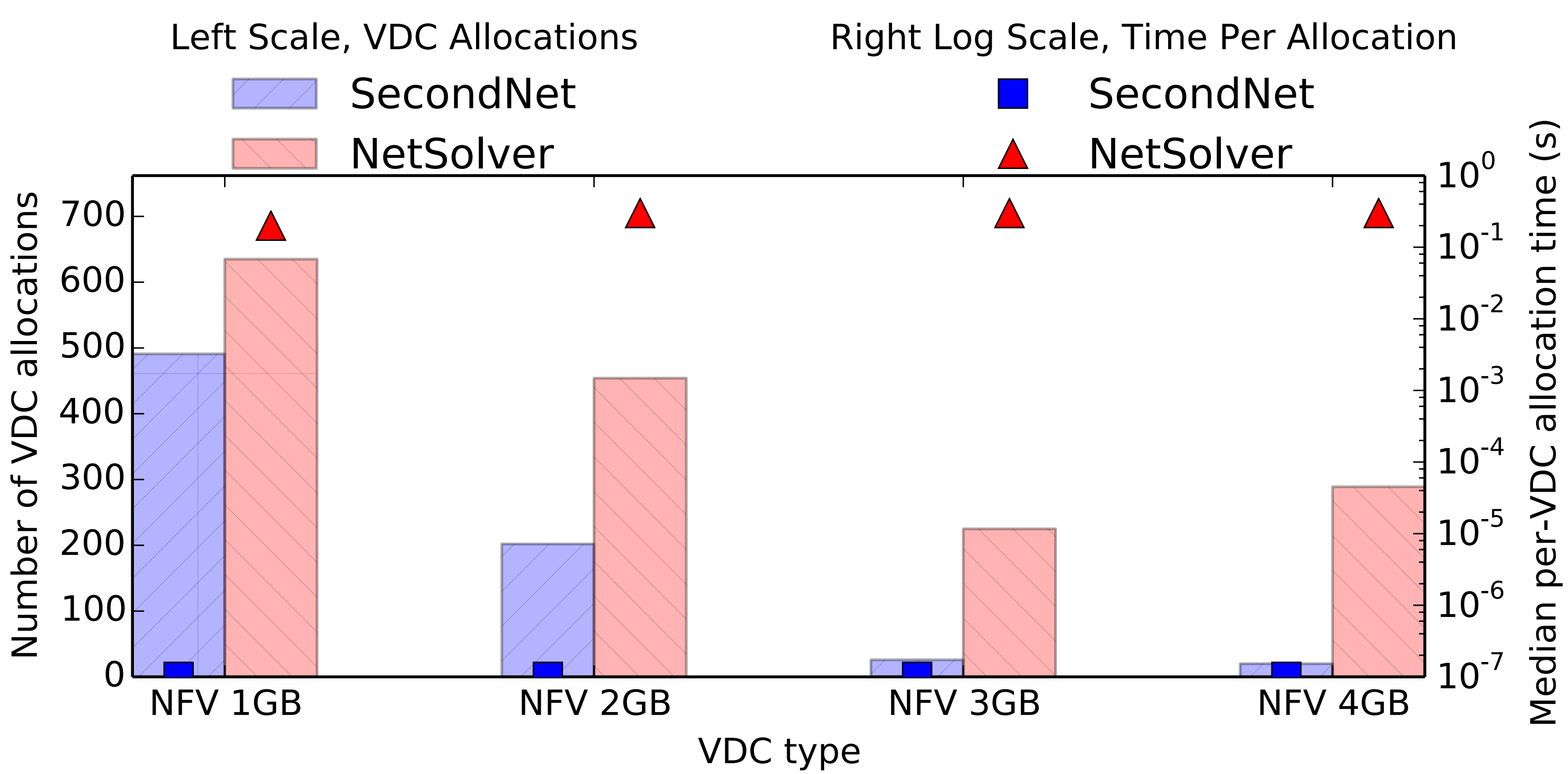
NFV chain allocation



Intrusion Prevention System	Firewall	Video Optimizer	Cache	Parental Control
4 CPU core	4 CPU core	2 CPU core	2 CPU core	4 CPU core
0 GPU core	0 GPU core	4 GPU core	0 GPU core	0 GPU core
8 GB RAM	8 GB RAM	8 GB RAM	4 GB RAM	8 GB RAM
0 TB SSD	0 TB SSD	0 TB SSD	10 TB SSD	0 TB SSD

NFV chain allocation

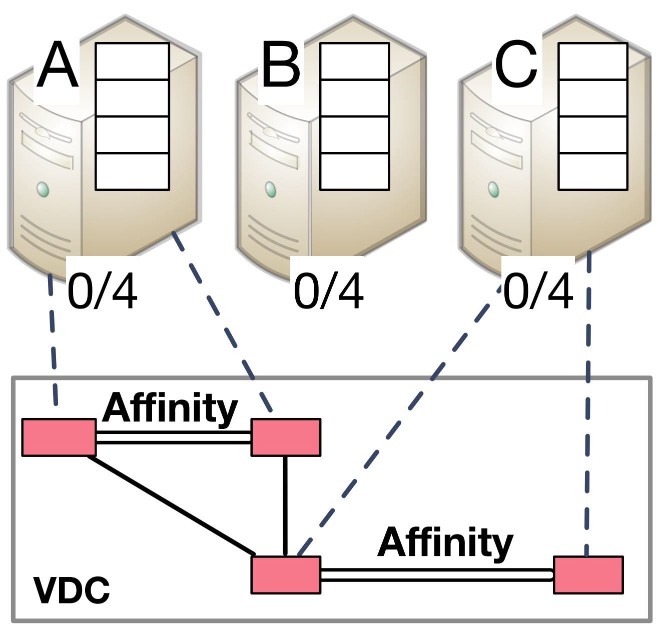




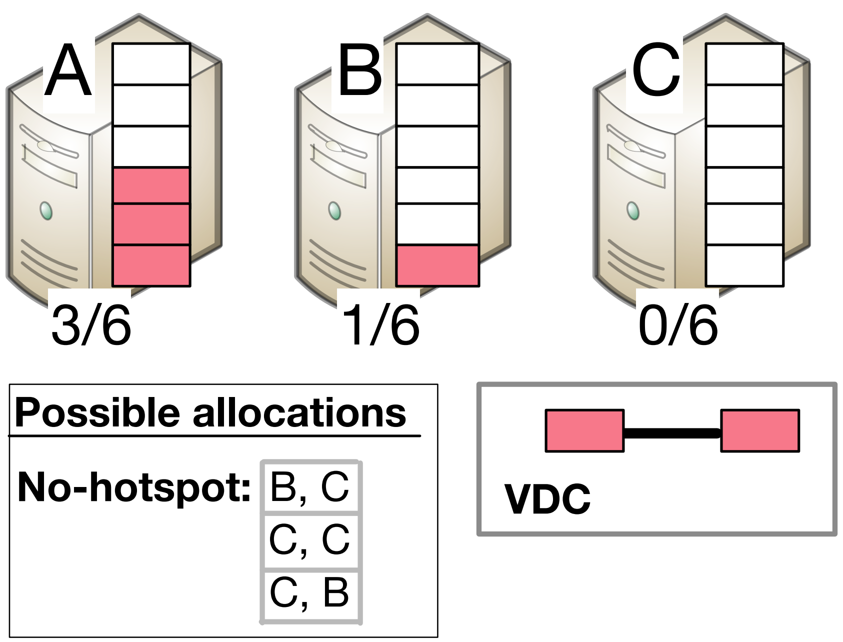
1200 servers; commercial DC topology; increasing chain bandwidth constraint

Extensibility

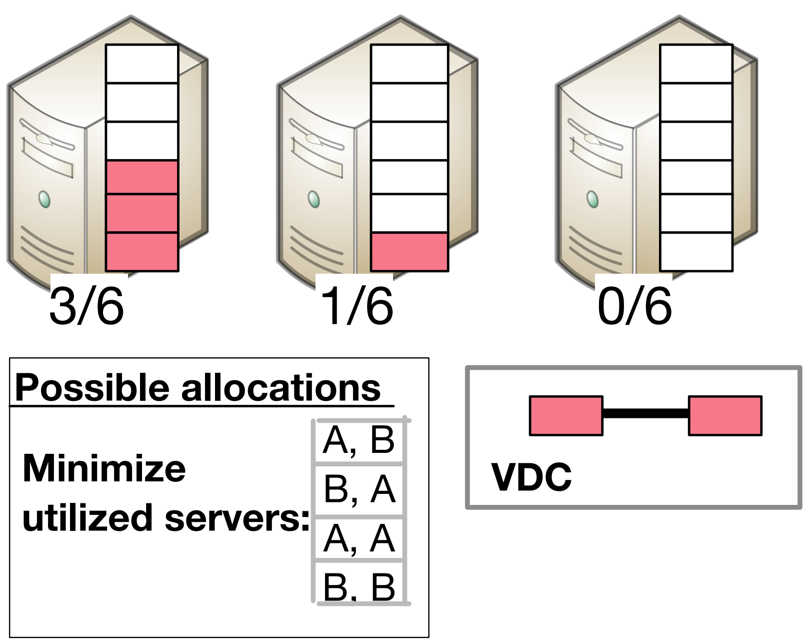
- NetSolver supports a variety of additional constraints



Affinity

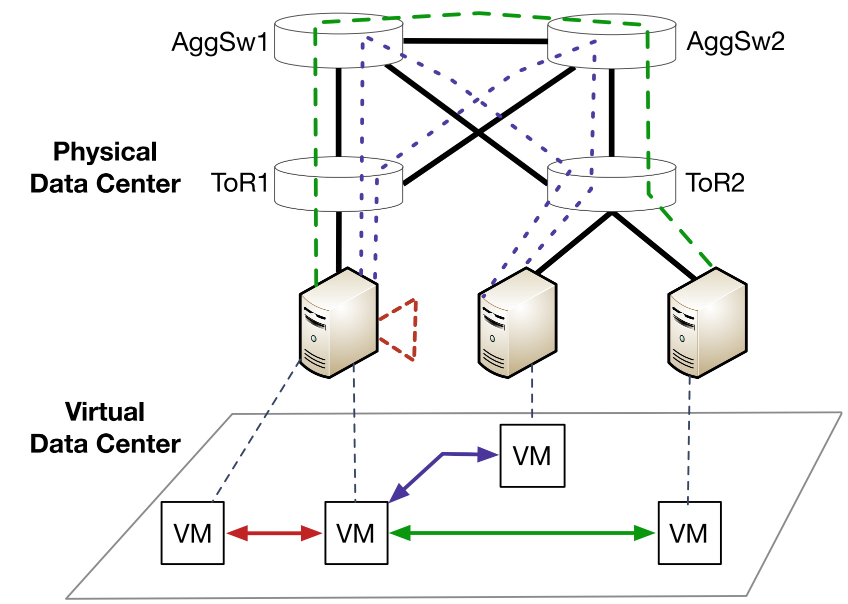


No hotspots



Minimize utilized severs

Contributions



- Developed NetSolver, a new VDC allocator
  - NetSolver encodes problem into MonoSAT. Can be reused for other problems: NFV placement, data migration, task distribution, etc
  - Improves DC capacity utilization by 300% over prior work (but slower than incomplete approaches)
  - Constraints-based approach flexibly extends to other kinds of constraints, such as (anti-)affinity