A Comprehensive Comparison of Two Resource Allocation Schemes in DCNs under the Hose Model

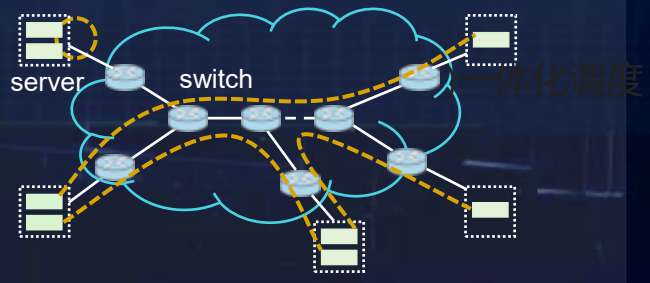
Jie Wu*, Shuo Quan， and Xiaoyao Huang

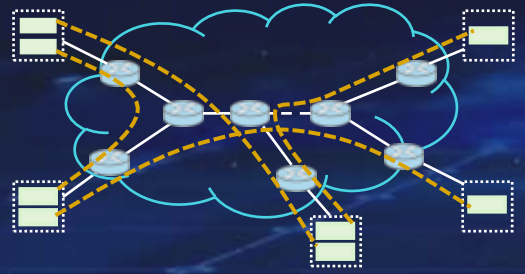
China Telecom and *Temple University

Unified Scheduling

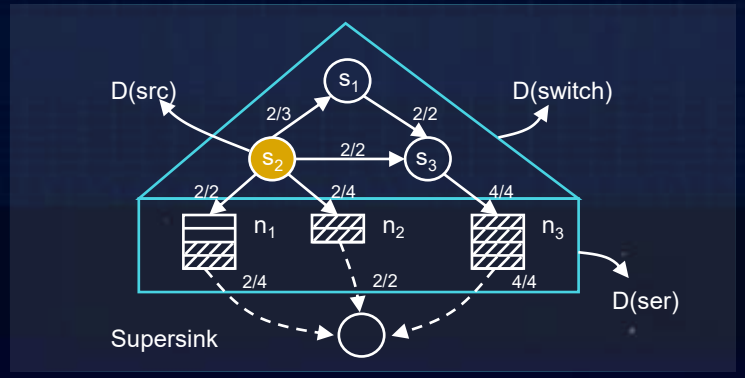
The **efficient integration** and **unified scheduling** of computing and network resources.

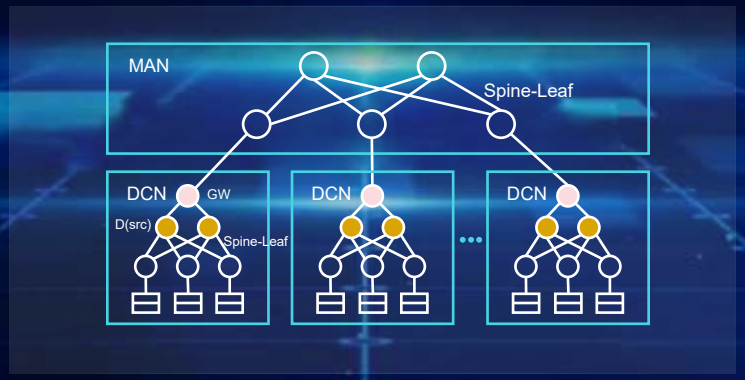
Resource Modeling



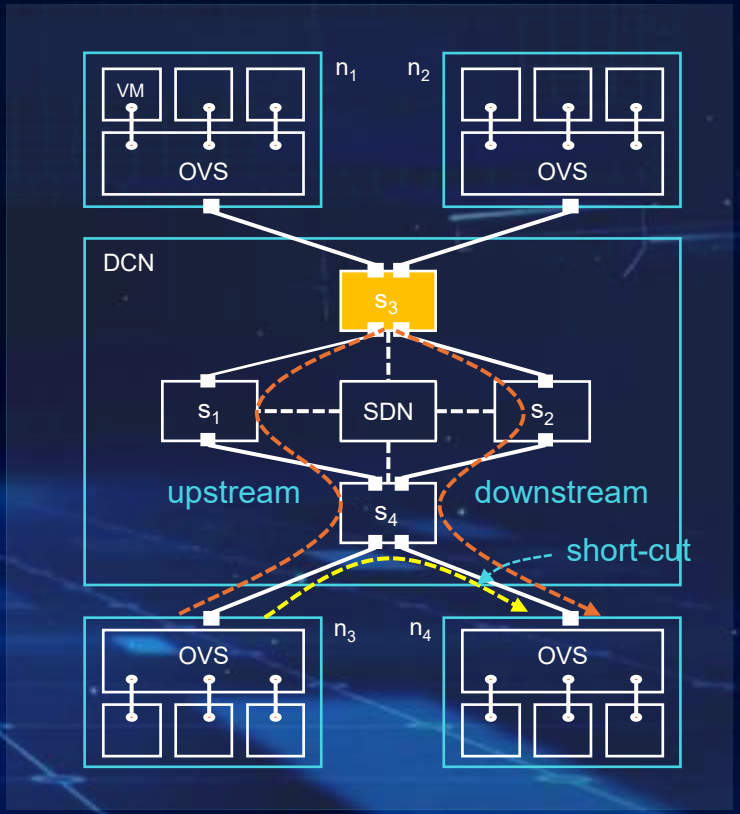


Scheduling Optimization





Traffic Management



2. Resource Modeling

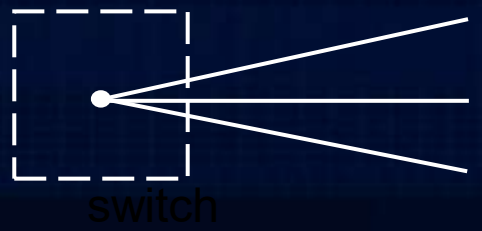
Hose model :

Aggregated traffic demand per site

B=3G multiplexing
(1,1,1), (2,1,0), (3,0,0)

Pipe model:

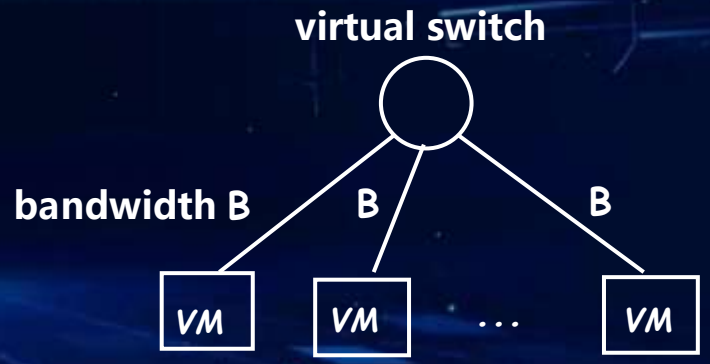
Pairwise traffic demand



instead of (1, 1, 1)

AWS EC2

Instance type	vCPUs	Memory (GB)	Bandwidth (Mbps)
m7i.xlarge	4	16	1250
m7i.2xlarge	8	32	2500
m7i.4xlarge	16	64	5000



How to allocate a slice: a chunk of compatible computing resource (VMs) and network resource (bandwidth)?

Why Not Trivial?

- Given a cable connection in a graph, each household has a maximum *occupancy limit* and each cable section has *bandwidth limit*.
- Allocate a slice that contains *N* ppl and can support all possible simultaneous pairwise video conversations.

hose model

How to Solve It (Polya)

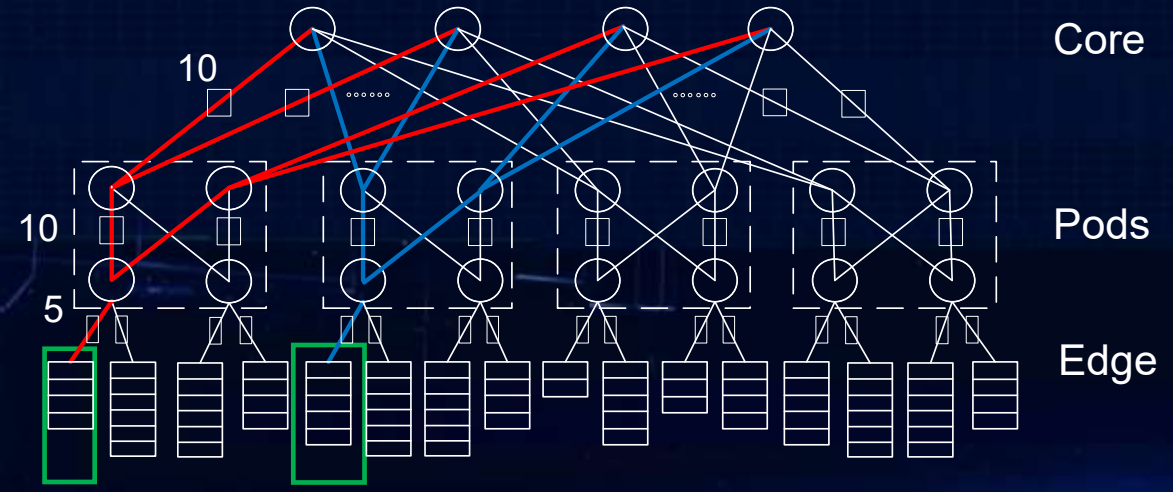
The scheduling problem is conjectured to be NP-hard for feasibility in a general network

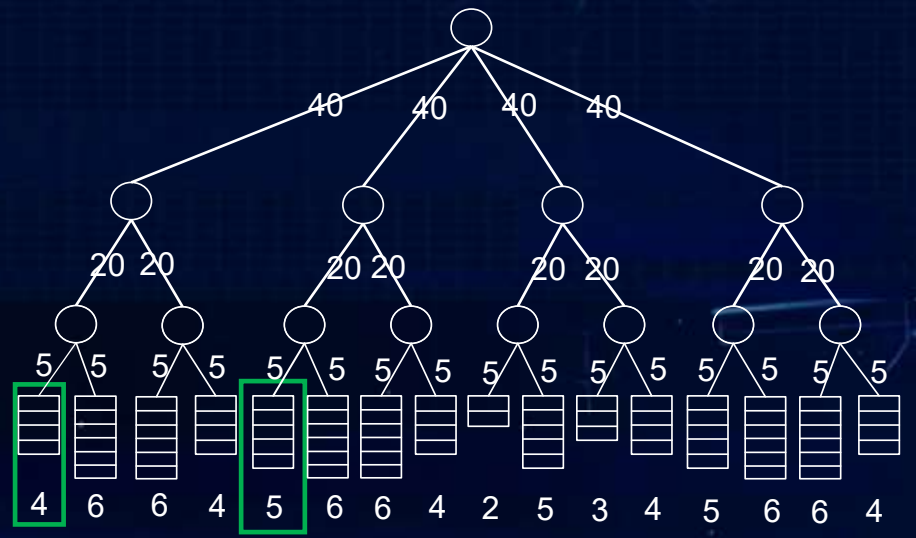
If you can't solve a problem, then there is an easier problem you can solve: find it

- Tree (typical DCN)
- Postmortem (V. Sekar, SIGCOMM 2024)

DCN: Fat-Tree

Equal-cost multi-path routing (ECMP) with 4 ports

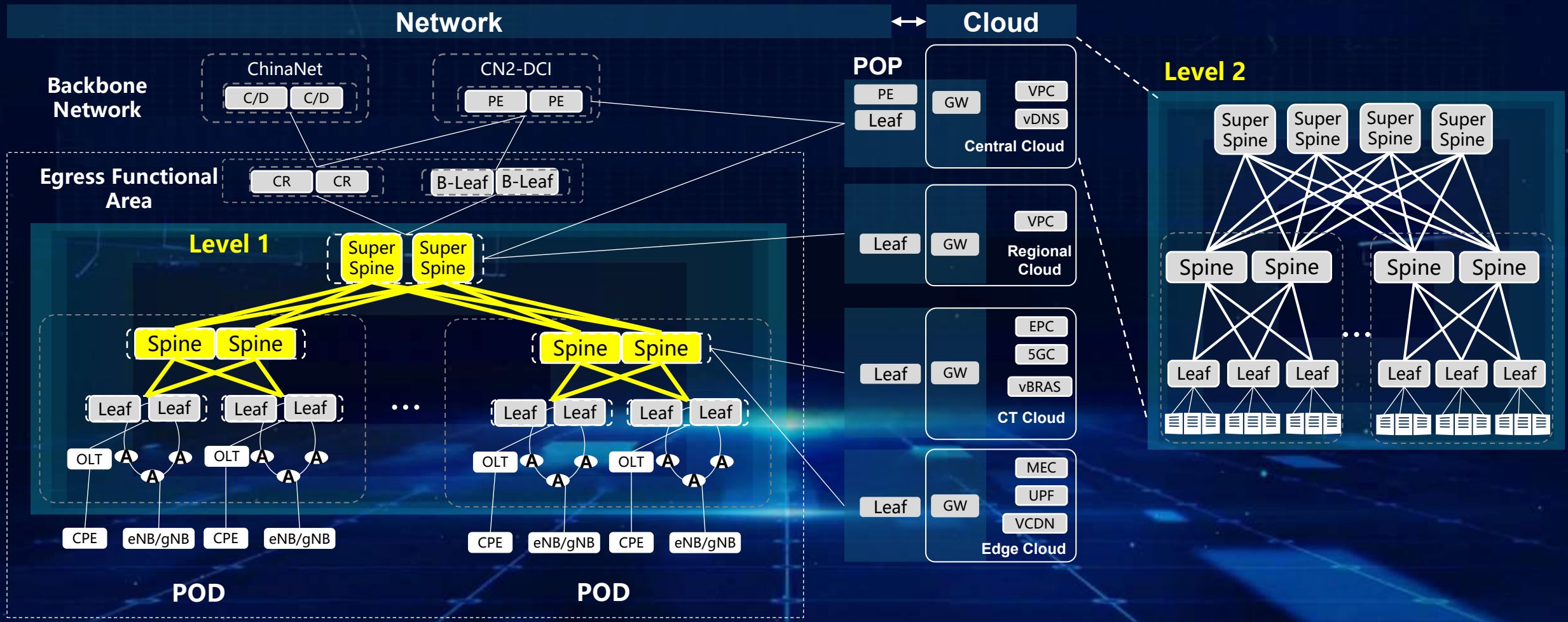




Fat-tree structure

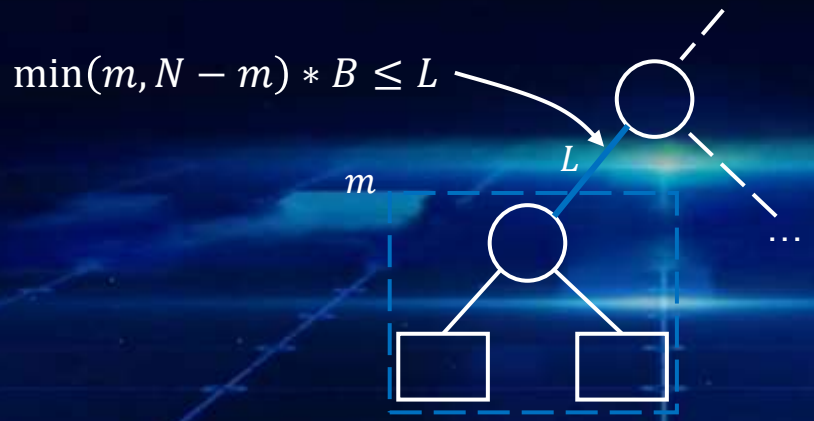
MAN: Two-level Spine-and-Leaf

Large-scale network: China Telecom MAN



3. Scheduling Optimization

- **Oktopus** (SIGCOMM 2011, with > 1K citations)
- Problem: Given *N* VMs, schedule them if a solution exists in a DCN.
- Solution:
  - Virtual cluster with an outbound link (each VM requires *B* bandwidth)
  - Outbound link capacity should be at least *min {m, N-m} * B*

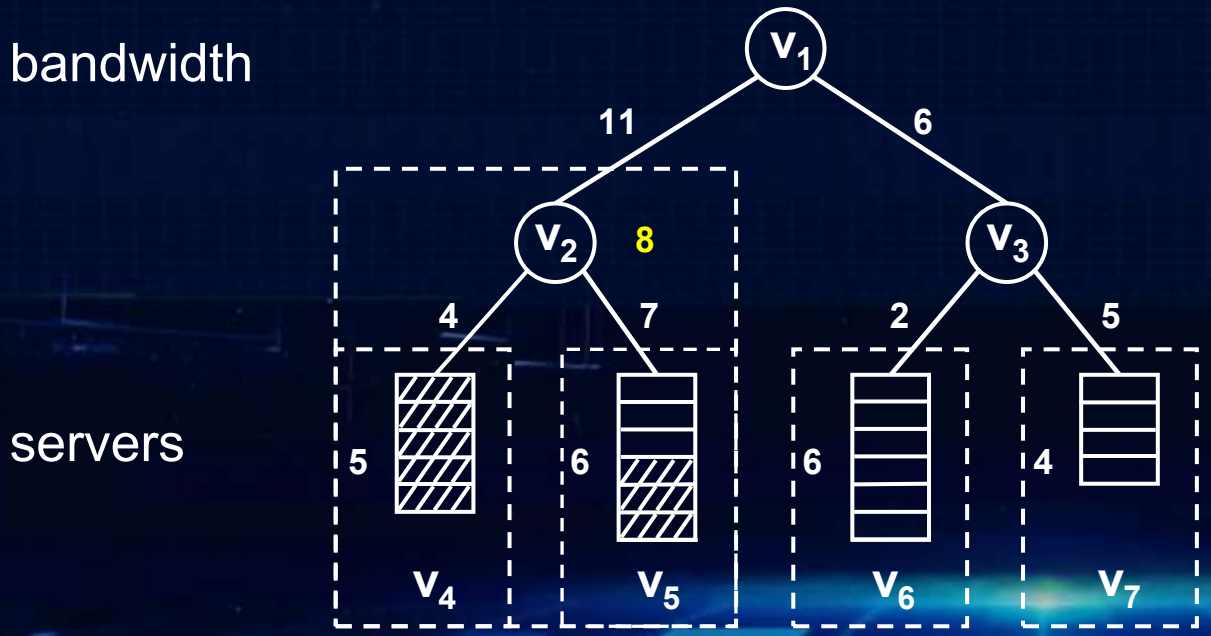


Local: *m* and global: *N*

So inside: *m* and outside *N-m*

Oktopus Solution

Examine clusters from left to right, and from lower to upper layers



When N = 8 (assuming B = 1G)

$M_{v_4} = 5$

$M_{v_5} = 6$

$M_{v_6} = 6$

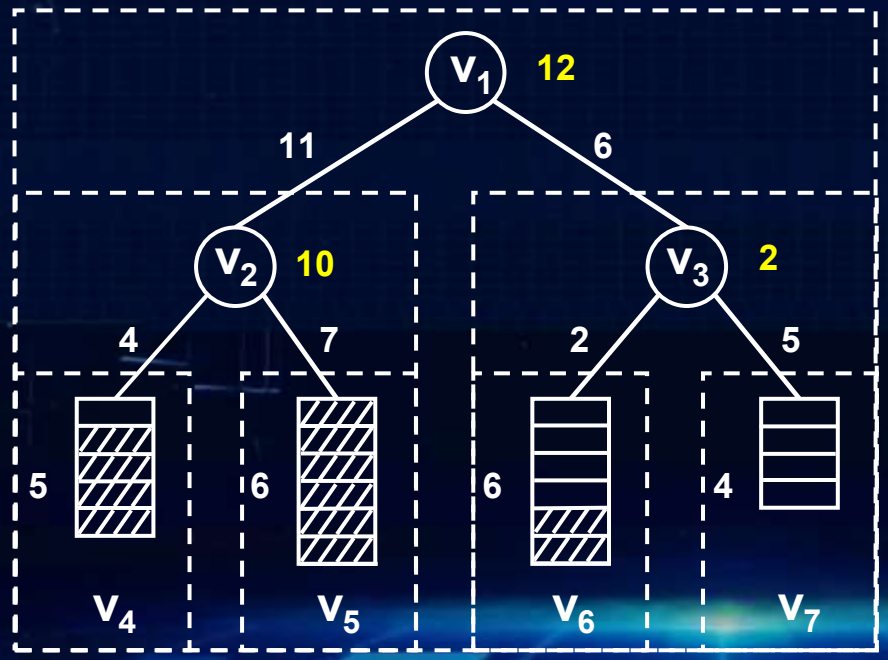
$M_{v_7} = 4$

$M_{v_2} = 11$

Avoid fragmentation, or reduce delay (hop count) and bandwidth

Oktopus Solution (Cont'd)

Different N have different schedules !



When N = 12

$M_{v_4} = 4$

$M_{v_5} = 6$

$M_{v_6} = 2$

$M_{v_7} = 4$

$M_{v_2} = 10$

$M_{v_3} = 6$

$M_{v_1} = 16$

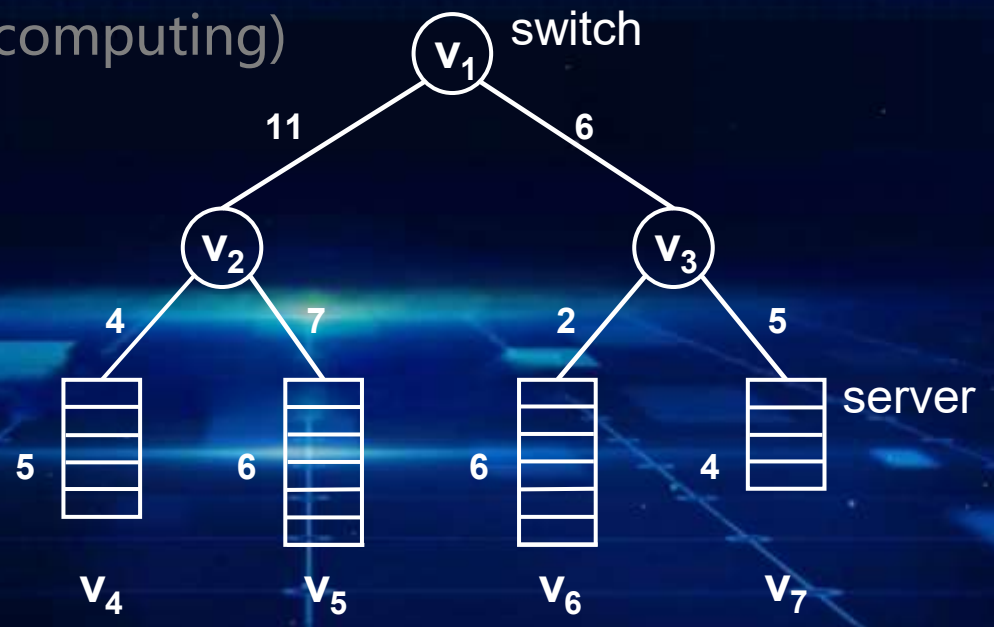
Issues: (1) No max load. (2) No support for traffic management.
(3) Sequential process in nature

Finding a Largest Slice

Maximum Admissible Load (MAL) (ICC 2018)

Problem: Given a set of VMs each requires a bidirectional link bandwidth of *B*, assign the maximum number of VMs in a DCN that meets the need of VM placement and their communication.

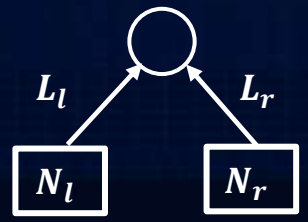
(Initially for maximum elastic computing)



A Simple Solution

Simple max load: Calculate maximum load at the root of a 2-level subtree

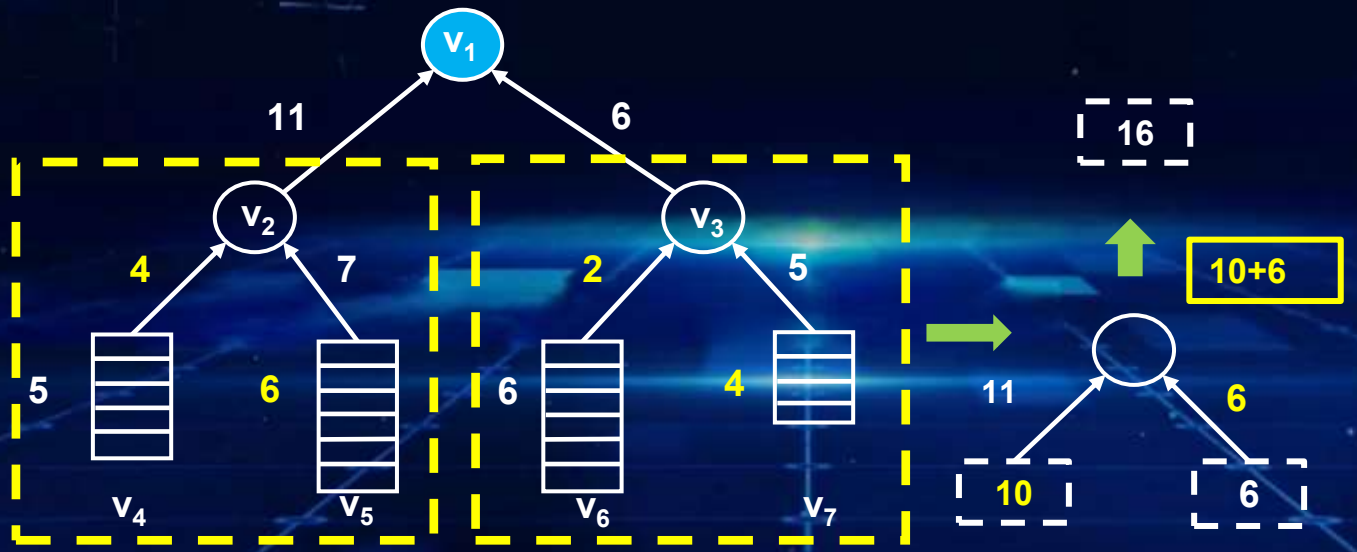
$$min\{N_l, L_l\} + min\{N_r, L_r\}$$



A proper data structure

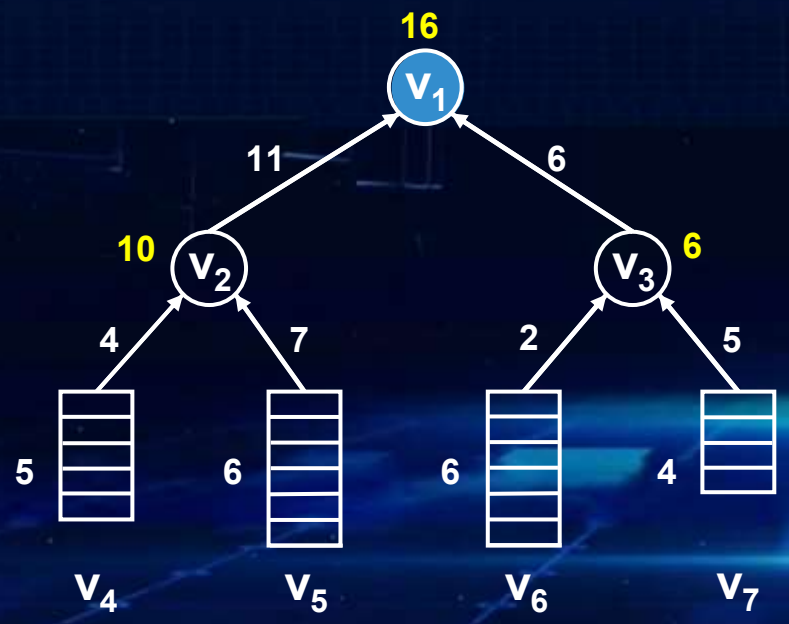
min: filtering

+: pooling

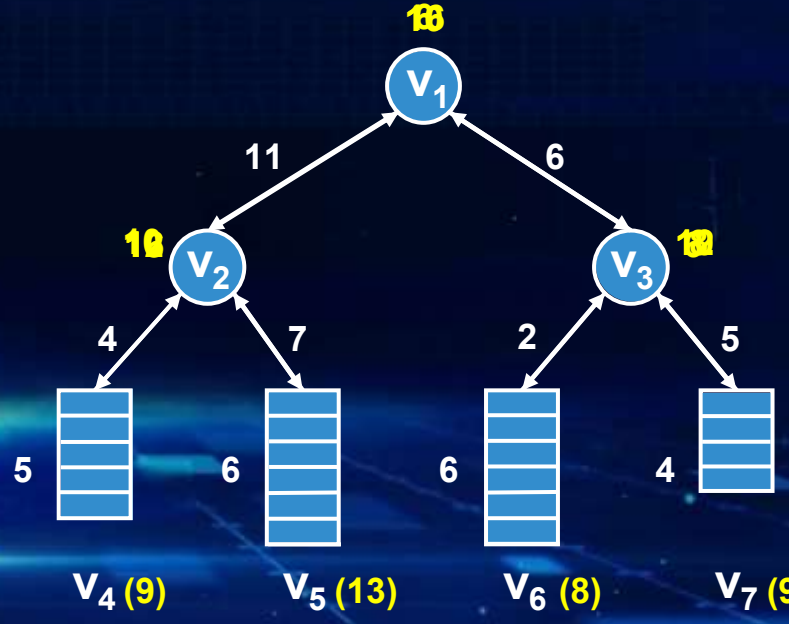


Optimal Solution

- Two-phase linear solution
  - Up-phase to find the max load at the root
  - Down-phase to find the max load at other nodes



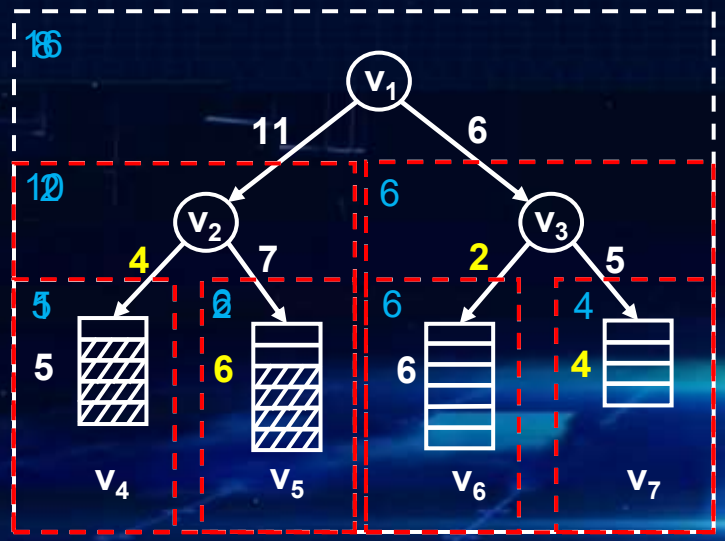
Up phase



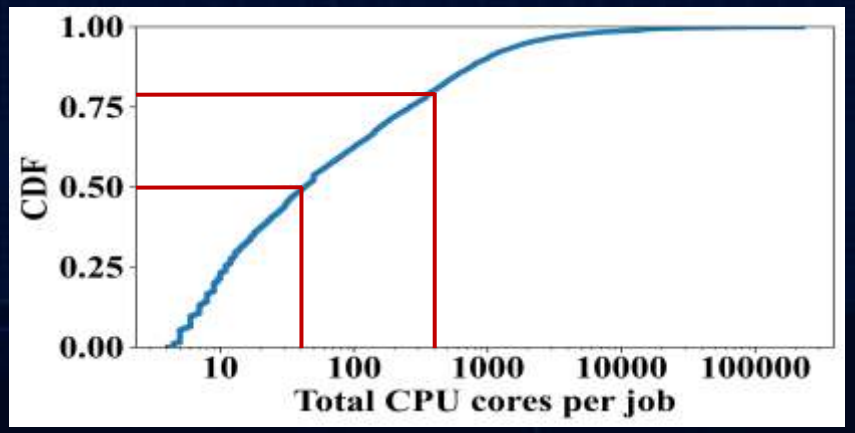
Down phase

Scheduling N VMs

- Scan from lower to upper subtree schedule
  - Best fit (smallest subtree ≥ N) and simple maintenance
  - N = 8



5. Simulation



Alibaba clusterdata v2018 trace

512 servers: each with 32 to 64 cpu cores

VM： 4 cpu cores

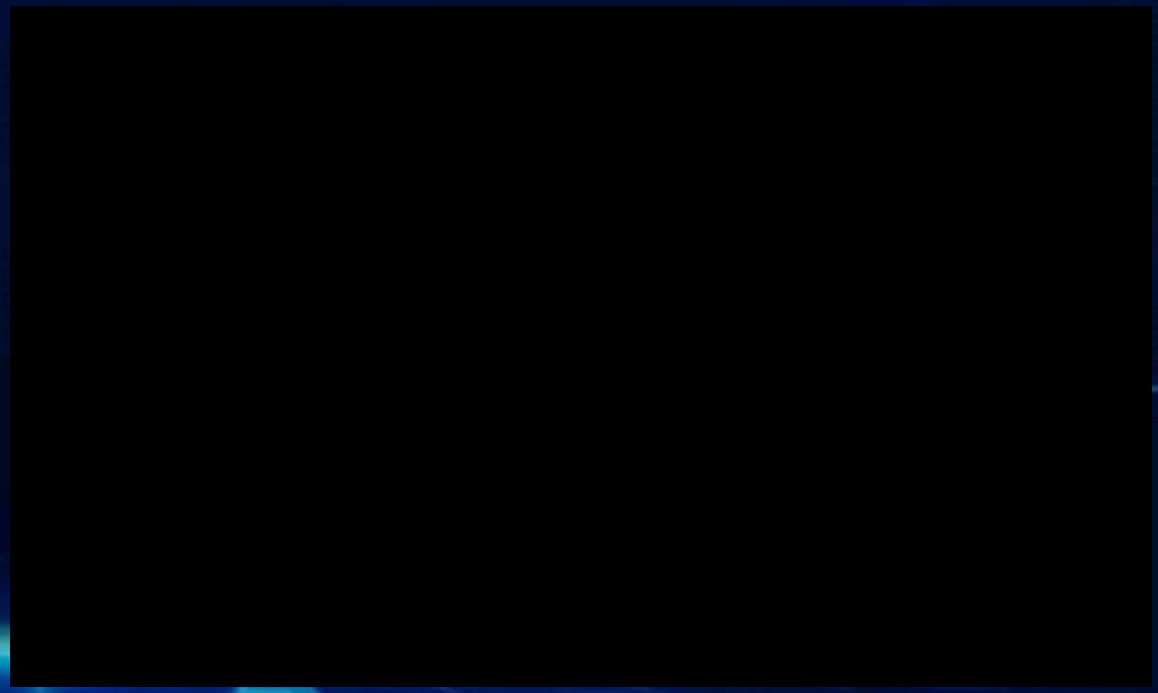
Small: 50%, jobs ≤ 10 VMs

Medium: 30%, 10 VMs < jobs ≤ 100 VMs

Large: 20%, 100 VMs < jobs

Performance Evaluation

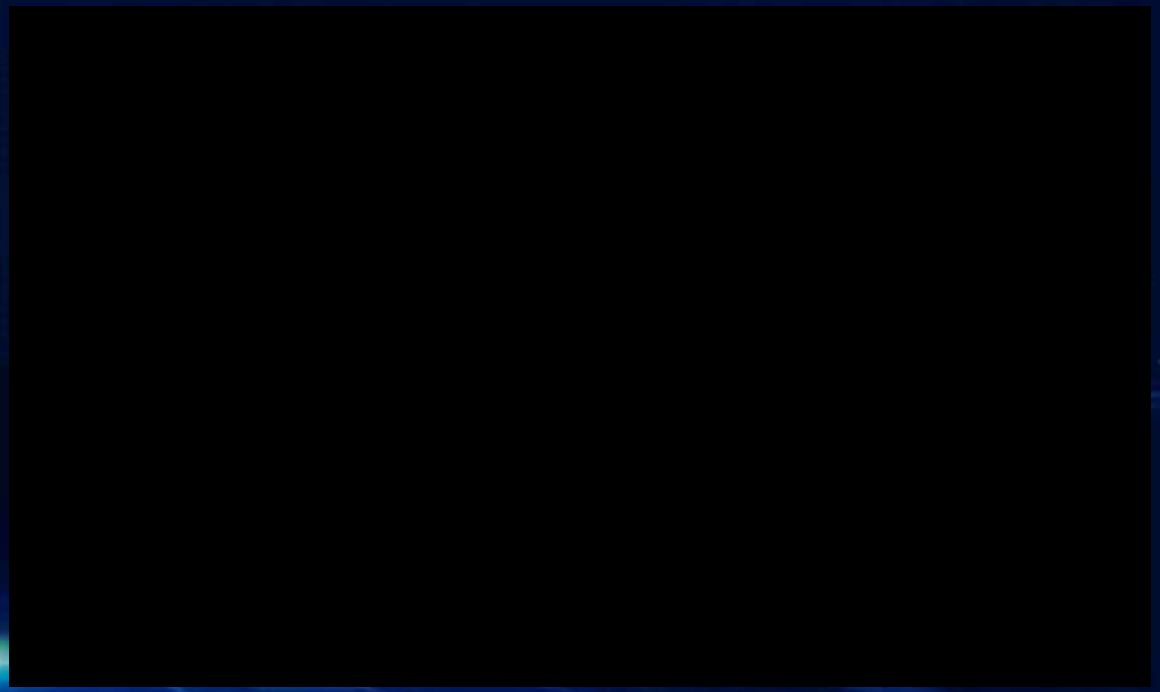
	Speed	Hop-count
○ Small	12.5%+3%	86.9%+3%
○ Medium	13.4%+7%	105%+7%
○ Large	48.3%+.5%	104%+0%
○ Mixed	13.3%+8%	101%+9%



Ratio: MAL/Oktopus, In only, dynamically drop a slice at 20-80% usage, and then garbage collection

Performance Evaluation (Cont'd)

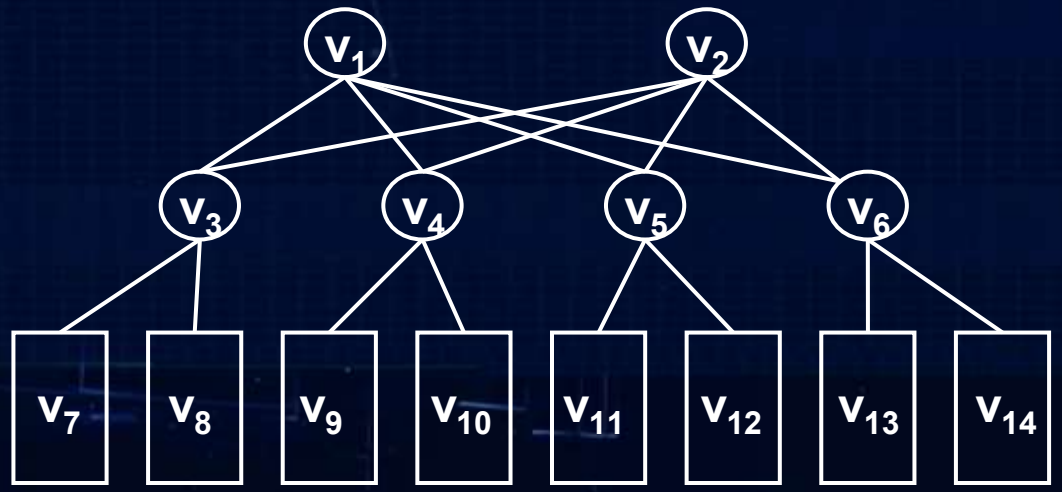
	Speed	Hop-count
○ Small	18.9%+3%	84.4%+2%
○ Medium	23.5%+5%	108.6%+3%
○ Large	52.0%+1%	106%+0%
○ Mixed	24.4%+2%	102%+2%



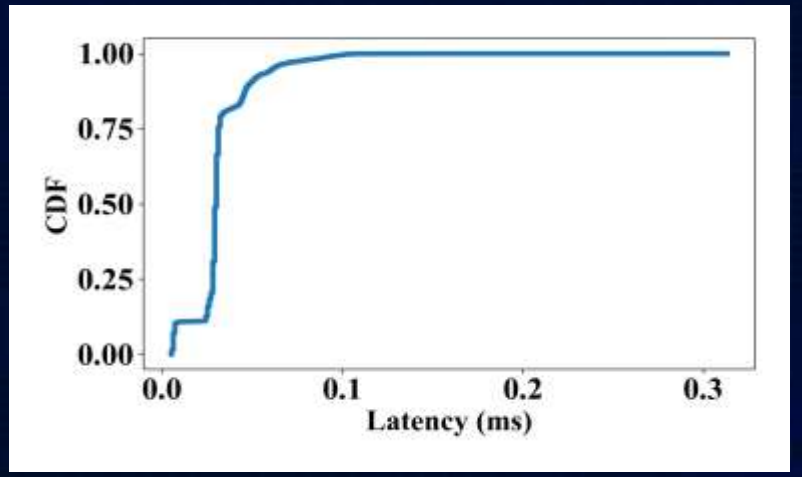
Ratio: MAL/Oktopus, In/Out traffic, dynamic load at 20-80%, drop at 20% and garbage collection

Testbed

v_1~v_2: 100G switch, v_3~v_6: 25G switch,
v_7~v_{14}: 32-core CPU, 256 GB main memory



Testbed



Distribution of end-to-end RTT

Method	Bandwidth	Hop Count	Scheduling Time	RTT
Tailor	12.53	1.45	0.025	0.028
Oktopus	12.4	1.29	0.062	0.028

Lesson Learned

- Difficulty in asking the right question in CS
- Importance of a proper (data) structure in design
  - Programming = Data Structure + Algorithm

H. Ballani et al. Towards Predictable Datacenter Networks, SIGCOMM 2011.

J. Wu et al. On Maximum Elastic Scheduling of VMs for Cloud-based DCNs, ICC 2018.