Survivable and Bandwidth-Guaranteed Embedding of Virtual Clusters in Cloud Data Centers

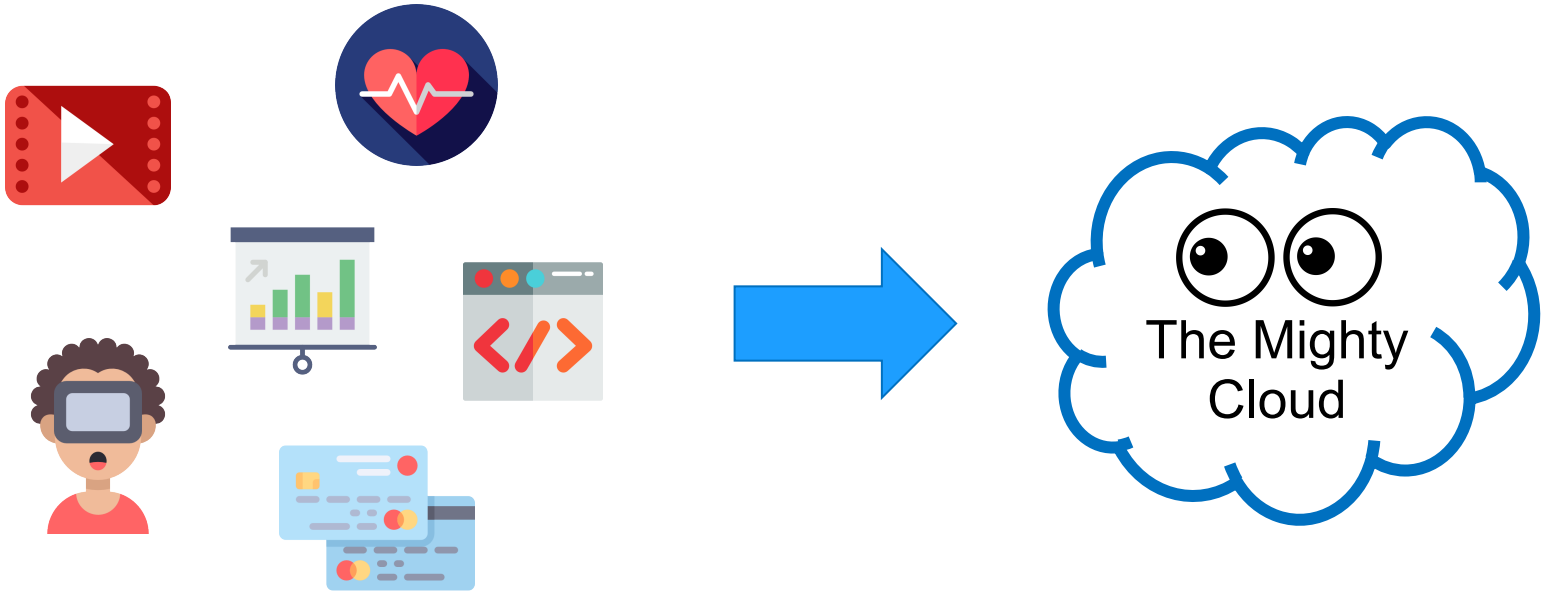
Ruozhou Yu, Guoliang Xue, and Xiang Zhang
Arizona State University
Dan Li
Tsinghua University

Outlines

- **Introduction and Motivation**
- System Model and Algorithm Design
- Performance Evaluation
- Conclusions

The Cloud Shift

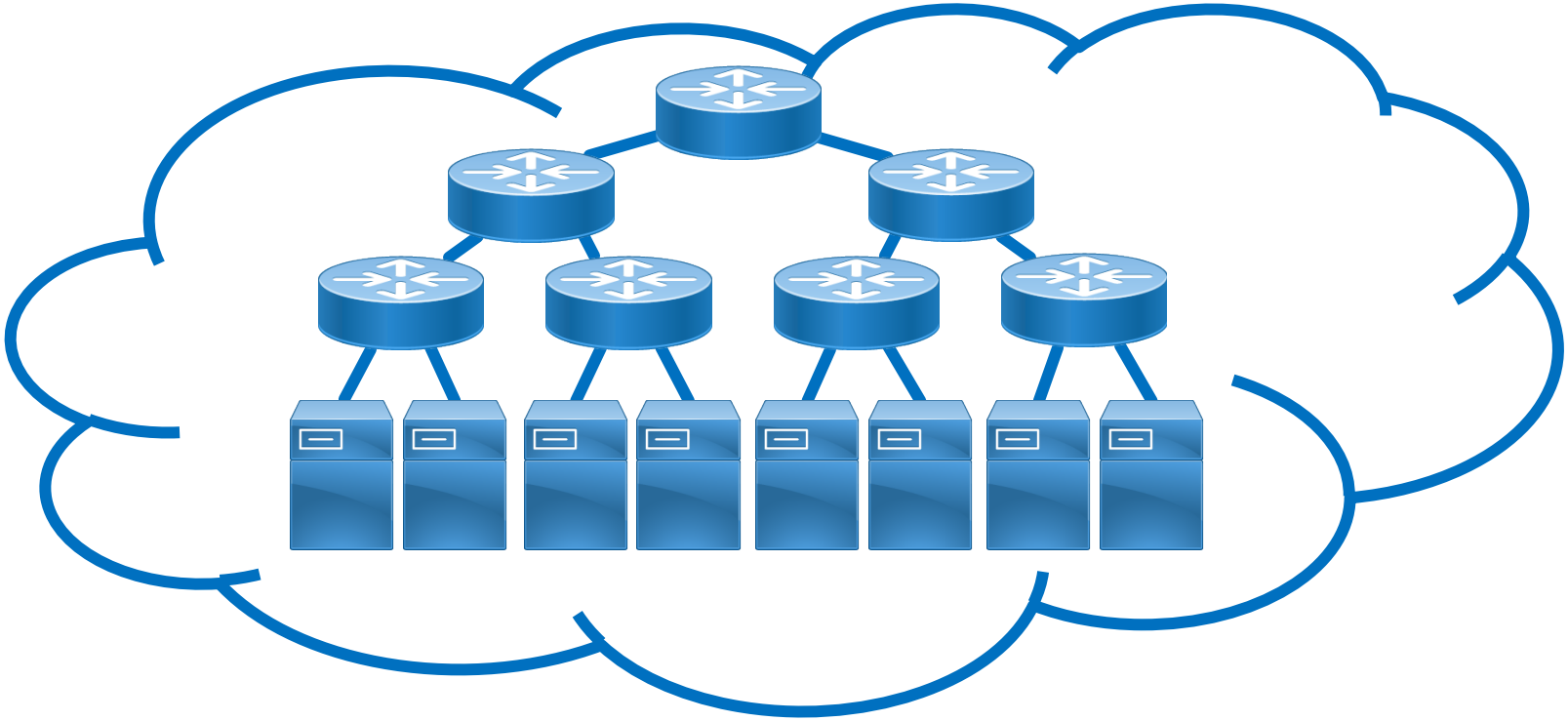
- **Cloud computing**: seems an omnipotent solution to all kinds of performance requirements



- *But is it as mighty as it seems?*

Inside the Cloud

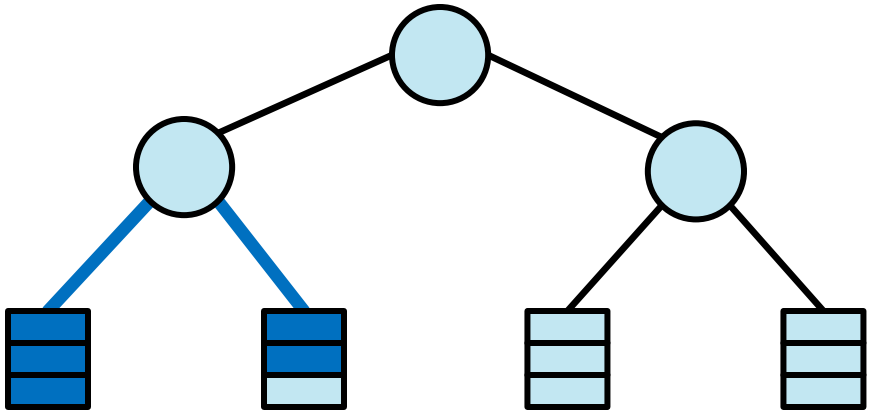
- An illusion of infinite computing resources created by large clusters of **interconnected** machines in data centers



- Performance bottleneck: **Cloud network**!

VM & Bandwidth

- Traditional approach: Network-agnostic VM allocation
- Recent advance: Bandwidth-guaranteed VM allocation
- Or **Virtual Cluster Embedding (VCE)**!



  - Existing algorithms can allocate bandwidth-guaranteed VMs with minimum bandwidth, migration costs, etc.

- But we know that Cloud machines do fail, quite often…

Survivable VCE

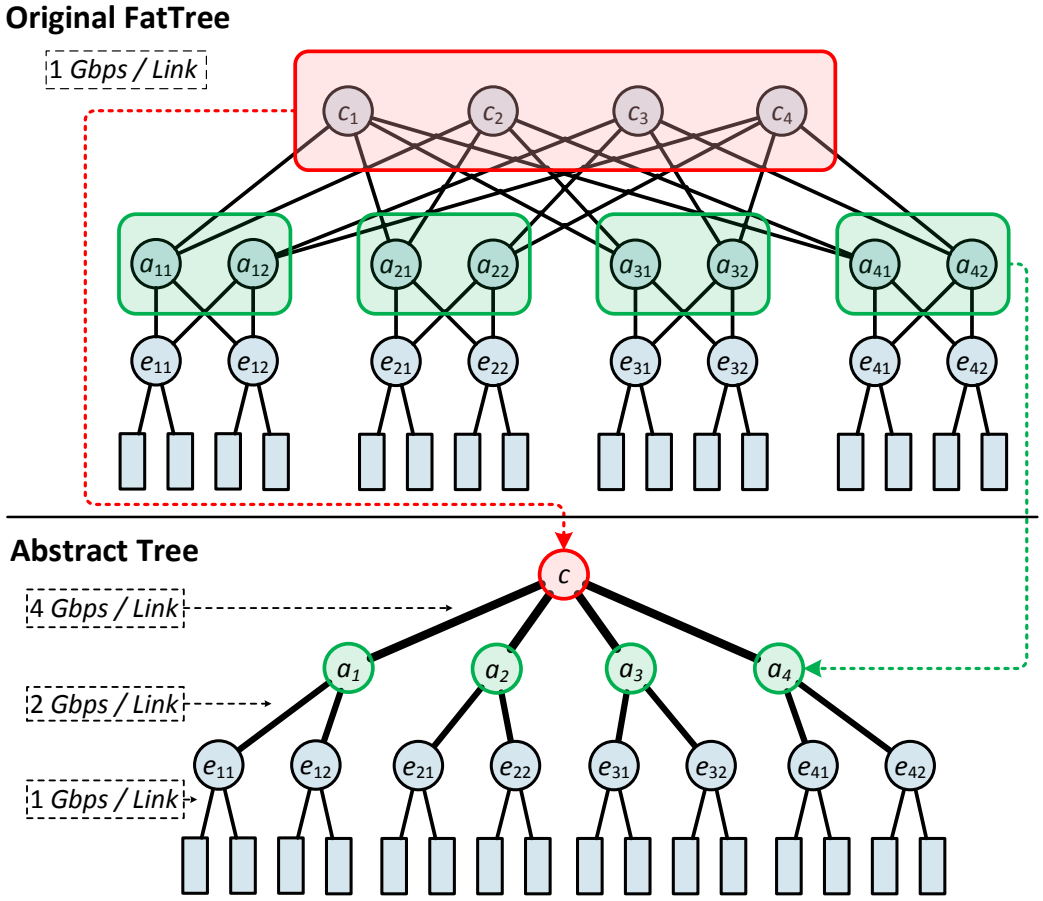
- **Question**: How can we ensure VM availability even when its host machine could fail?
- **Answer**: We prepare extra VMs and bandwidth just in case!
- **Question**: And how much will that cost us?
- **Answer**: No problem! We can minimize that!
- **Question**: How are we going to achieve that?
- **Answer**: Dynamic programming!

Outline

- Introduction and Motivation
- **System Model and Algorithm Design**
- Performance Evaluation
- Conclusions

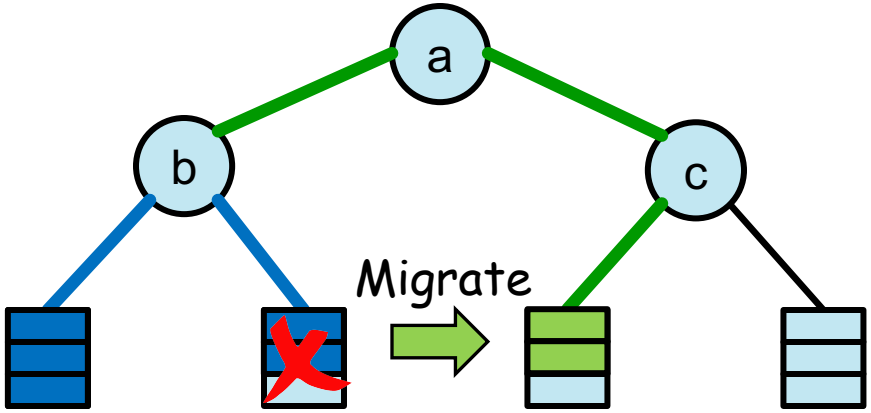
Network Topology

- **Assumption**: the DCN has a tree structure
  - Abstracts many common DCN topologies (FatTree, VL2, etc)



VM Survivability Model

- **Primary VMs**: VMs that are active during normal operations;
- **Backup VMs**: VMs in standby mode, activated when a primary VM's PM fails
  - Each backup VM synchronizes the states of multiple primary VMs



- ***Question**: Can we find a bandwidth-guaranteed allocation of both primary and backup VMs to cover an arbitrary single-PM failure, with the minimum number of backup VMs?*

Dynamic Programming for SVCE

- **Given**: topology tree T, request J = <N, B>
- **Assumption**: single PM failure
  - **Interpretation**: a failure can be either within a subtree, or outside a subtree, but cannot be both.
- **Key observation**: each subtree's ability to provide VMs is independent from the rest of the tree, both during normal operations and during an arbitrary failure
- Two layers of Dynamic Programming
  - **Outer DP**: DP for entire subtrees
  - **Inner DP**: DP for the first k sub-subtrees of each subtree

DP in Details

- **Outer DP**: $N_v[n_0, n_1]$ as the **minimum number of total VMs** needed in subtree T_v, to ensure that
  - T_v can provide at least n_0 VMs when no failure is in T_v;
  - T_v can provide at least n_1 VMs when any PM fails in T_v.

- **Inner DP**: $N_v'[n_0, n_1, k]$ as the **minimum number of total VMs** needed in the first k subtrees of v, to ensure that
  - The k subtrees can provide n_0 VMs when no failure is in them;
  - The k subtrees can provide n_1 VMs when any PM fails in them.

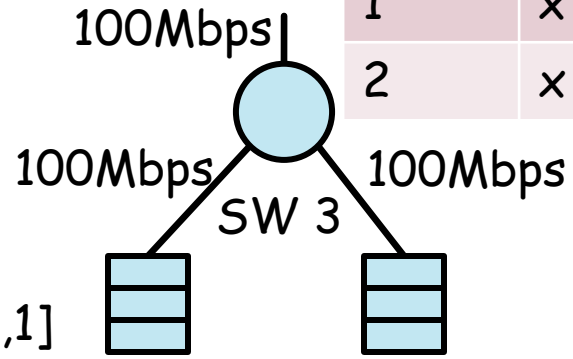
- Alternately update the two tables:
  - $N_v[n_0, n_1]$ depends on $N_v'[n_0', n_1', d_v]$ (d_v is the # subtrees under v);
  - $N_v'[n_0, n_1, k]$ depends on $N_v[n_0'', n_1'']$ of lower-layer nodes.

Work-through Example

- J = <2, 100 Mbps>

$N_3'[n_0,n_1,2] / N_3[n_0,n_1]$

$n_0 \backslash n_1$	0	1	2
0	x	x	x
1	x	x	x
2	x	x	x



$N_1[n_0,n_1] / N_3'[n_0,n_1,1]$

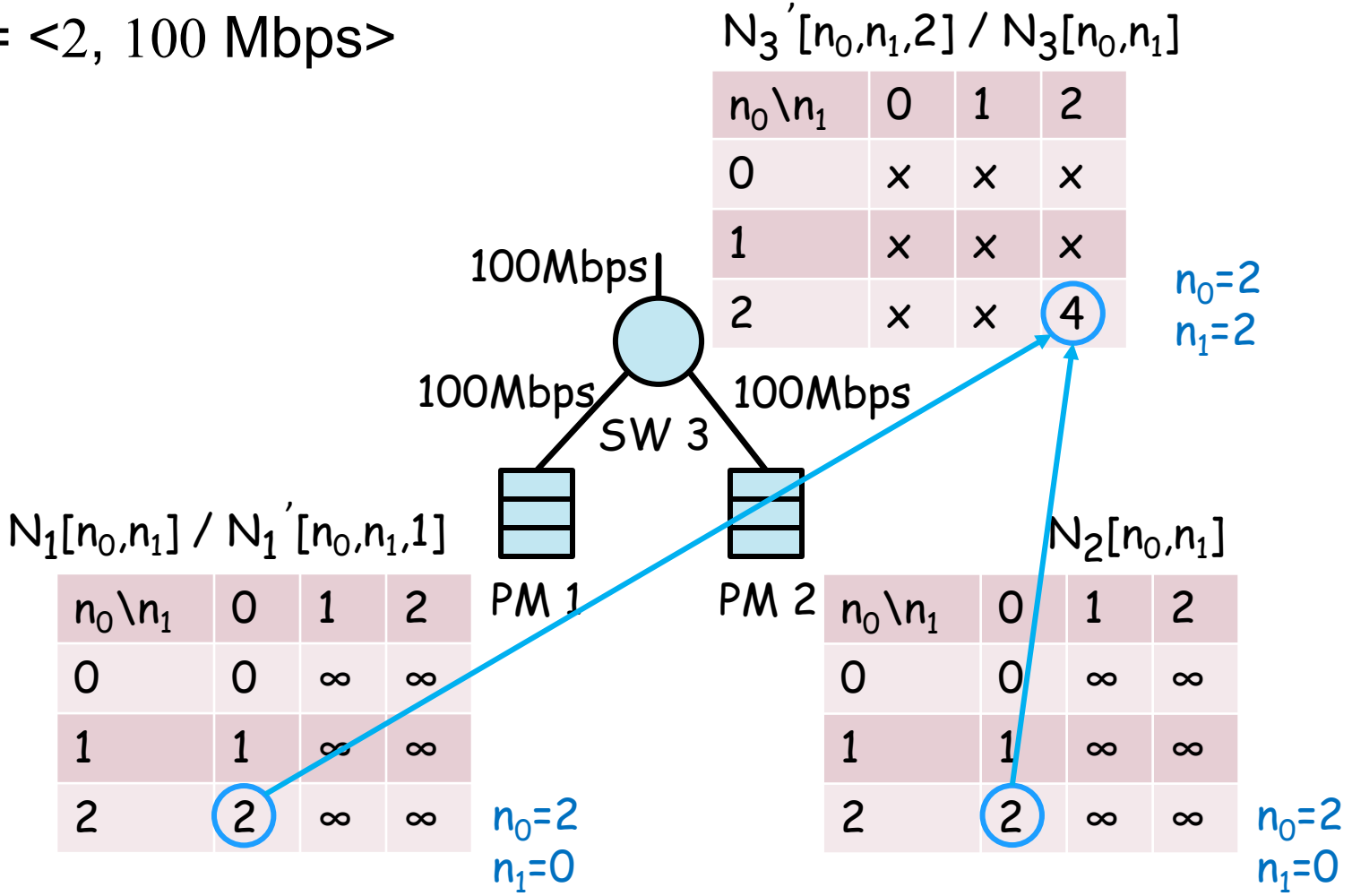
$n_0 \backslash n_1$	0	1	2
0	0	∞	∞
1	1	∞	∞
2	2	∞	∞

$N_2[n_0,n_1]$

$n_0 \backslash n_1$	0	1	2
0	0	∞	∞
1	1	∞	∞
2	2	∞	∞

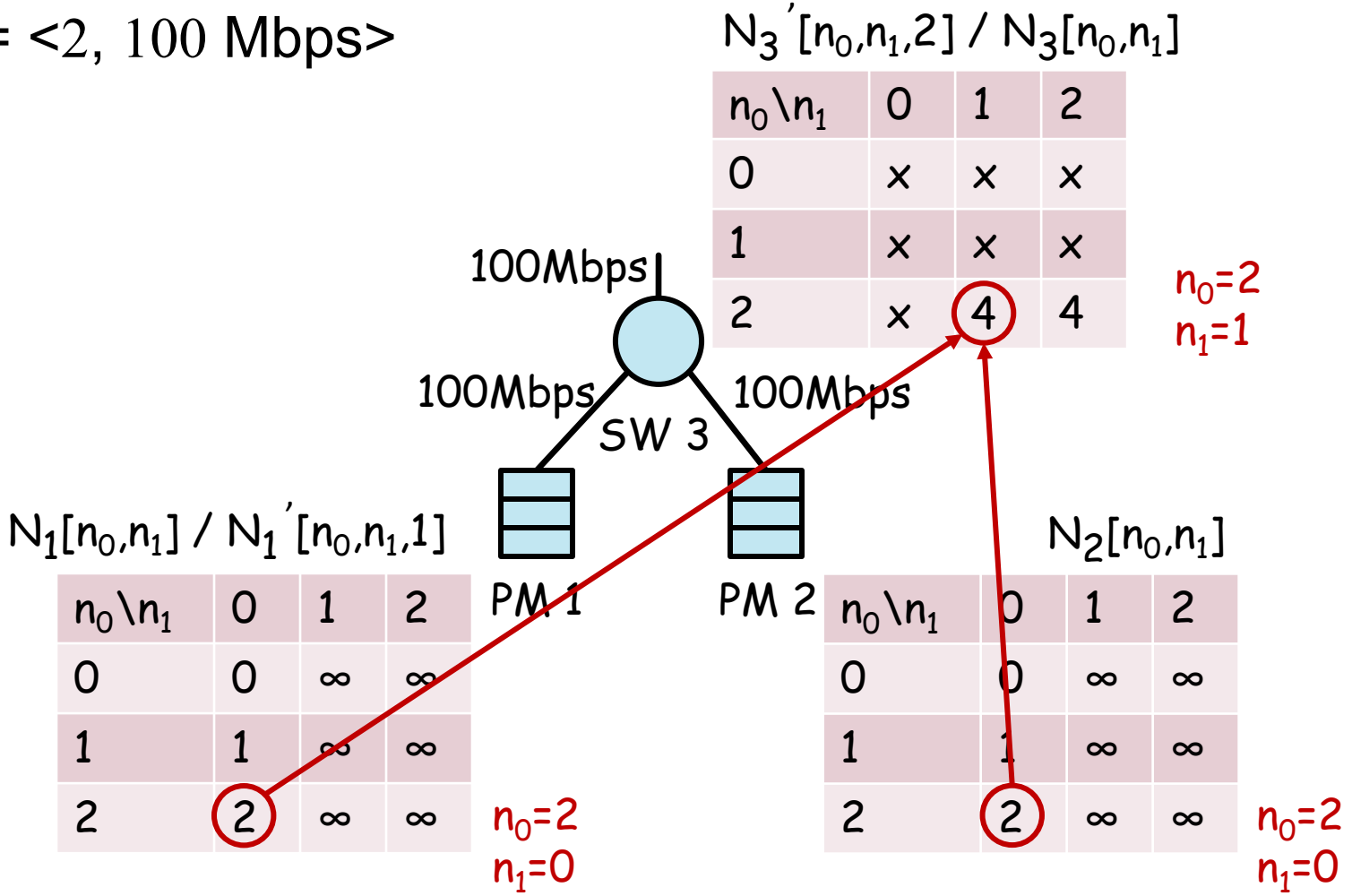
Work-through Example

- J = <2, 100 Mbps>



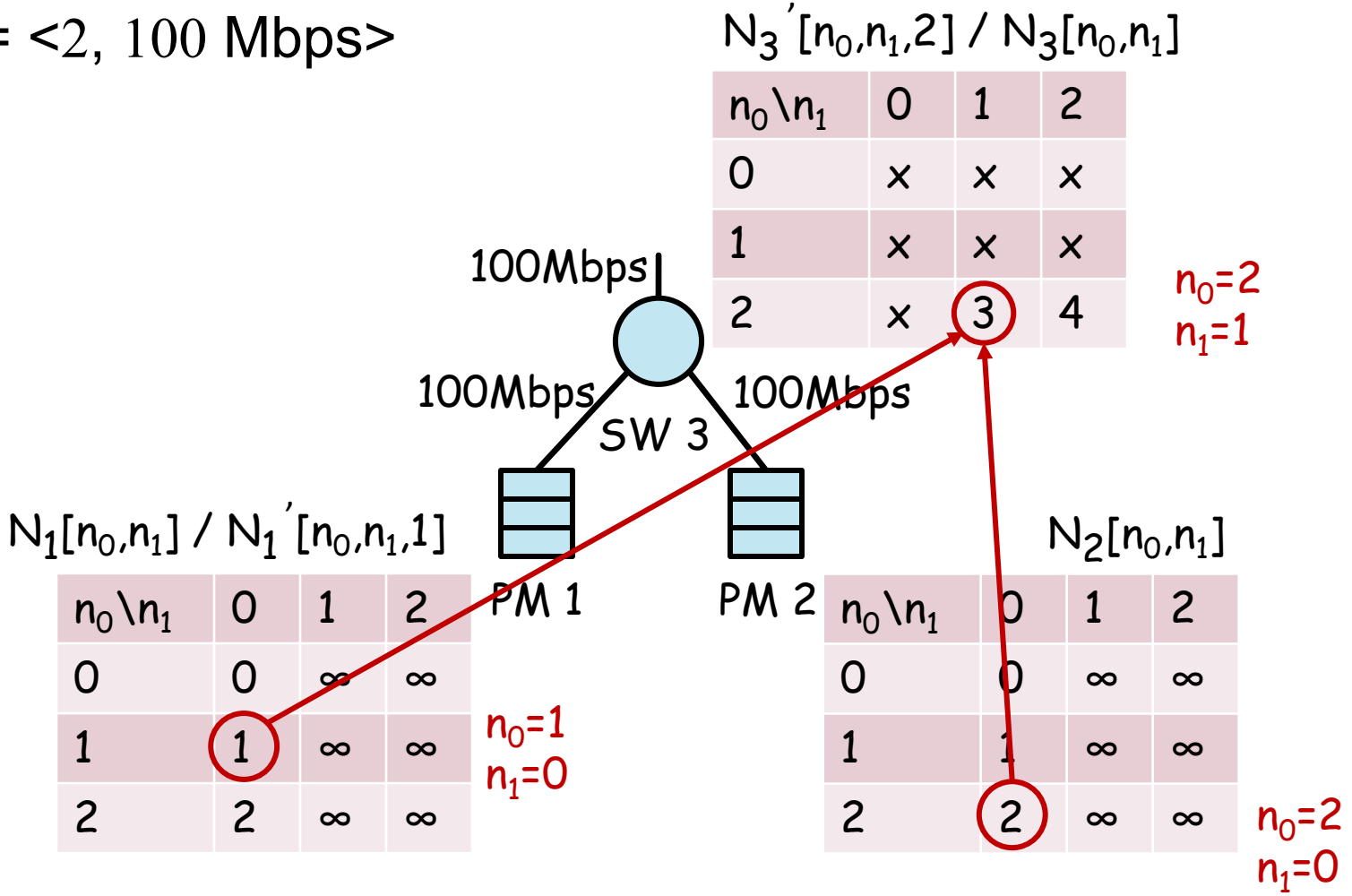
Work-through Example

- J = <2, 100 Mbps>



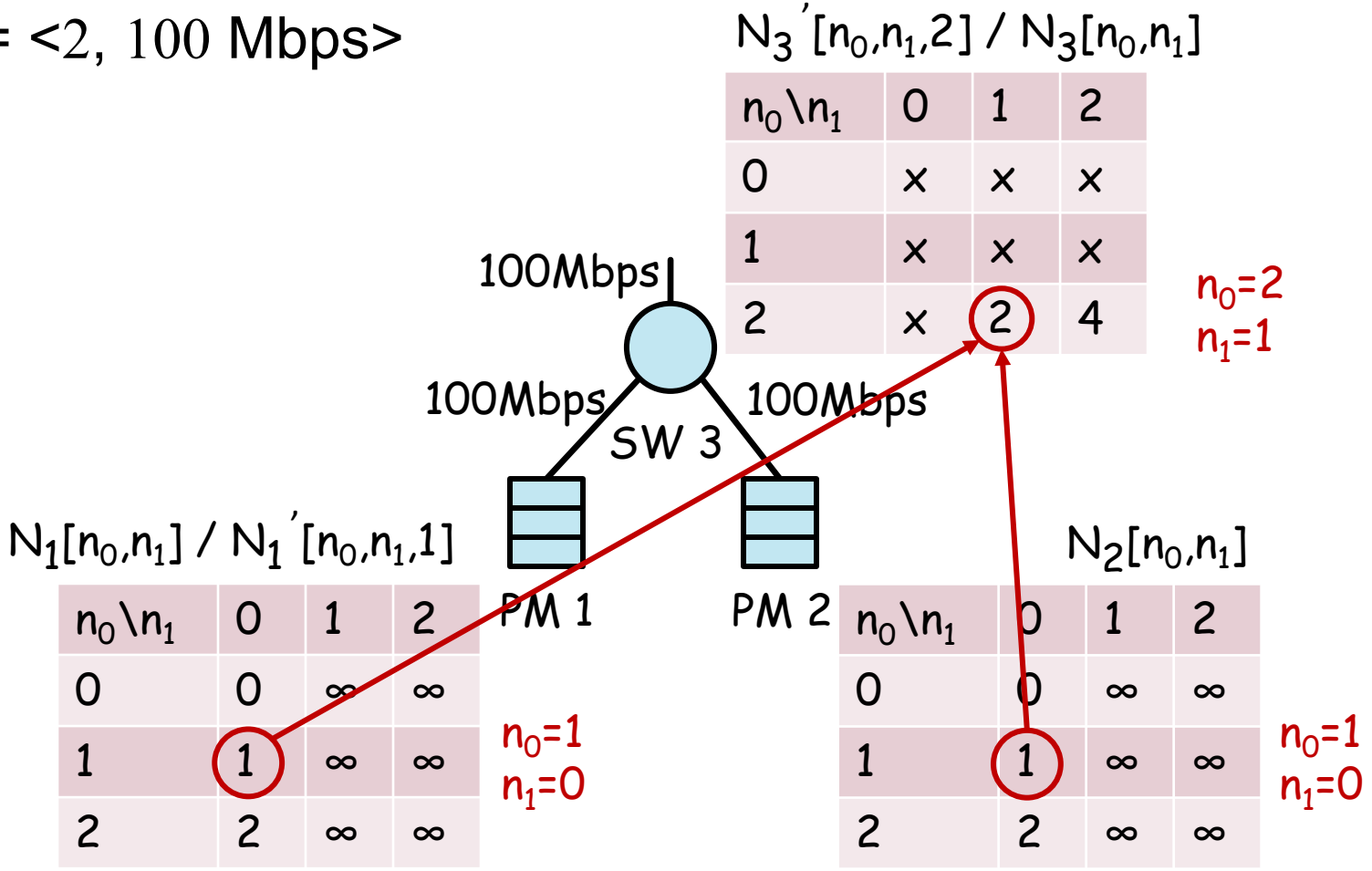
Work-through Example

- J = <2, 100 Mbps>



Work-through Example

- J = <2, 100 Mbps>



Heuristic SVCE

- **Optimal DP time complexity:** $O(|V| N^6)$
  - where $|V|$ is # tree nodes, N is # requested VMs.
- **Question**: Can we find a near-optimal solution with less time?
- **Observation**: if we find a normal VCE with $N+N'$ VMs, such that each PM hosts at most N' VMs, then we can always recover from any single PM failure.
- **Algorithm**: search from N'=1 to N, each time using an existing VCE algorithm to find a VCE with N' extra VMs, and each PM's # VMs is bounded by N'.
- **Time Complexity**: $O(N \cdot |V| \log |V|)$

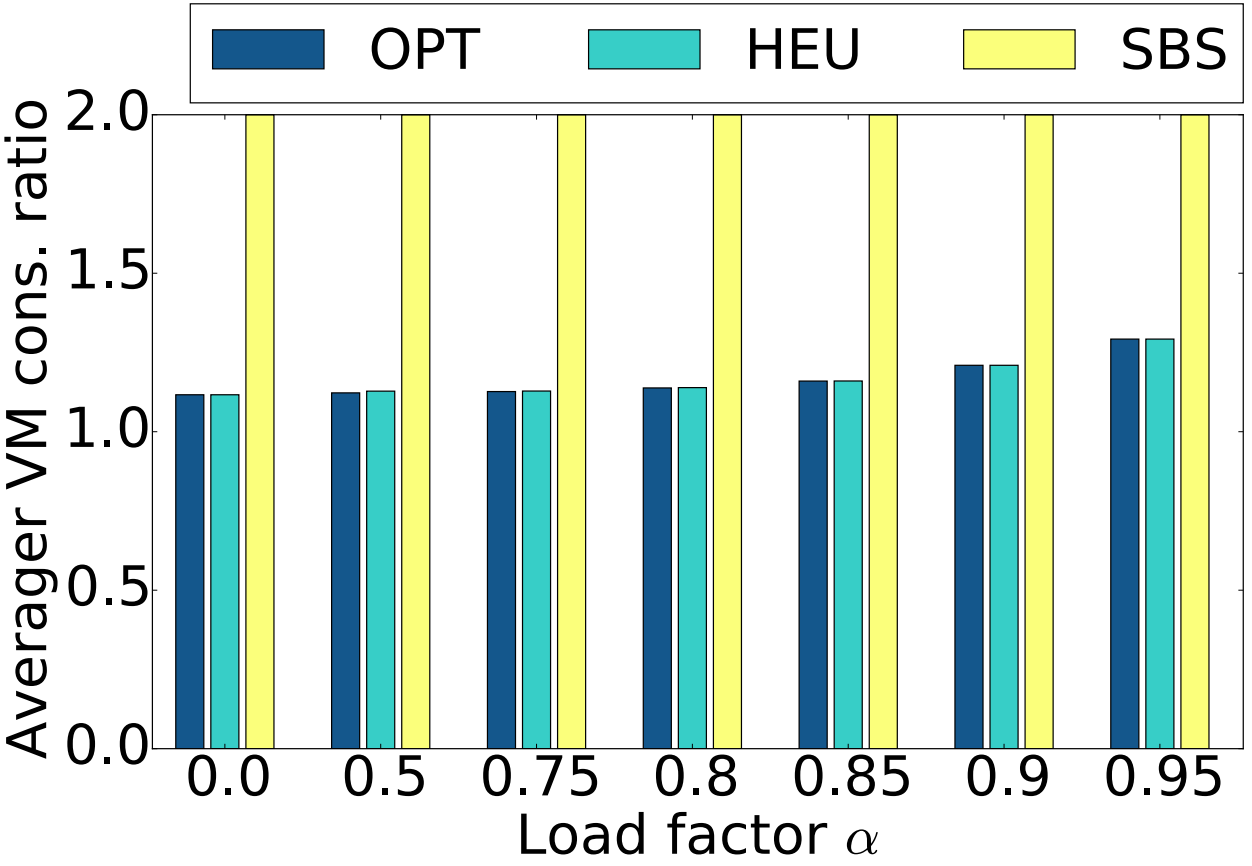
Outline

- Introduction and Motivation
- System Model and Algorithm Design
- **Performance Evaluation**
- Conclusions

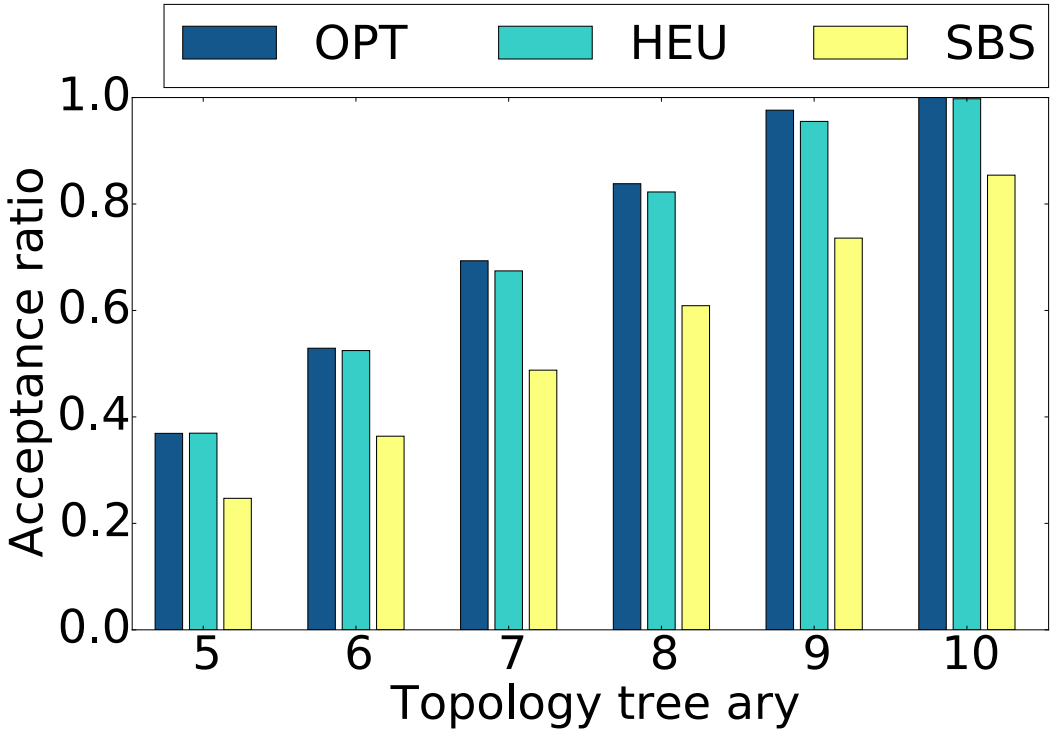
Simulation Setups

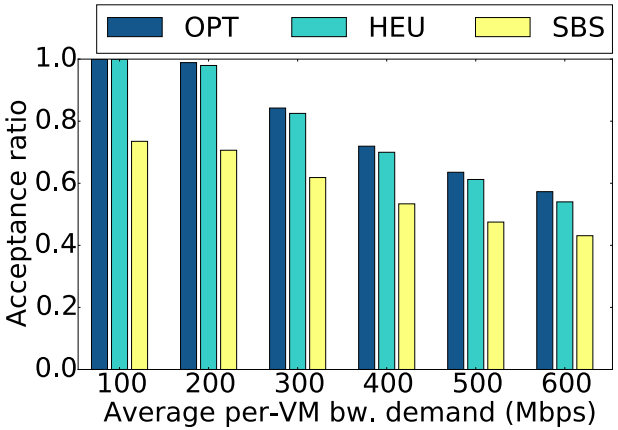
- Tree-structured DCN
  - 4-layer 8-ary (512 PMs, 73 switches)
  - 5 VM slots / PM
  - ToR bandwidth: 1 Gbps | Aggr/Core bandwidth: 10 Gbps
- Tenant VCs
  - 1000 requests
  - 15 VMs and 300 Mbps per VM, on average
  - Poisson arrivals
- Comparison:
  - OPT: Optimal DP SVCE algorithm
  - HEU: Heuristic SVCE algorithm
  - SBS: Shadow-based solution (dedicated VC backup)

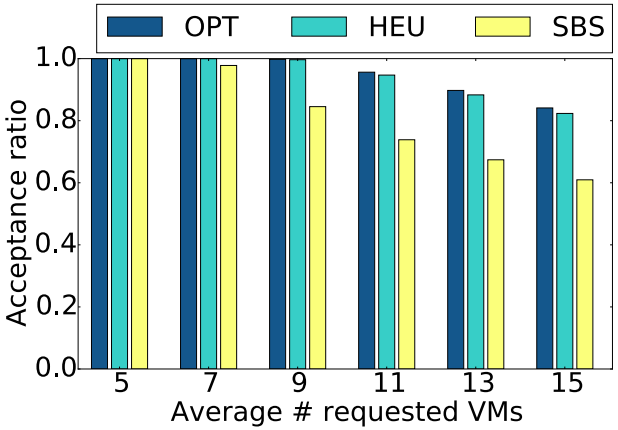
Simulation Results: Average VM Usage



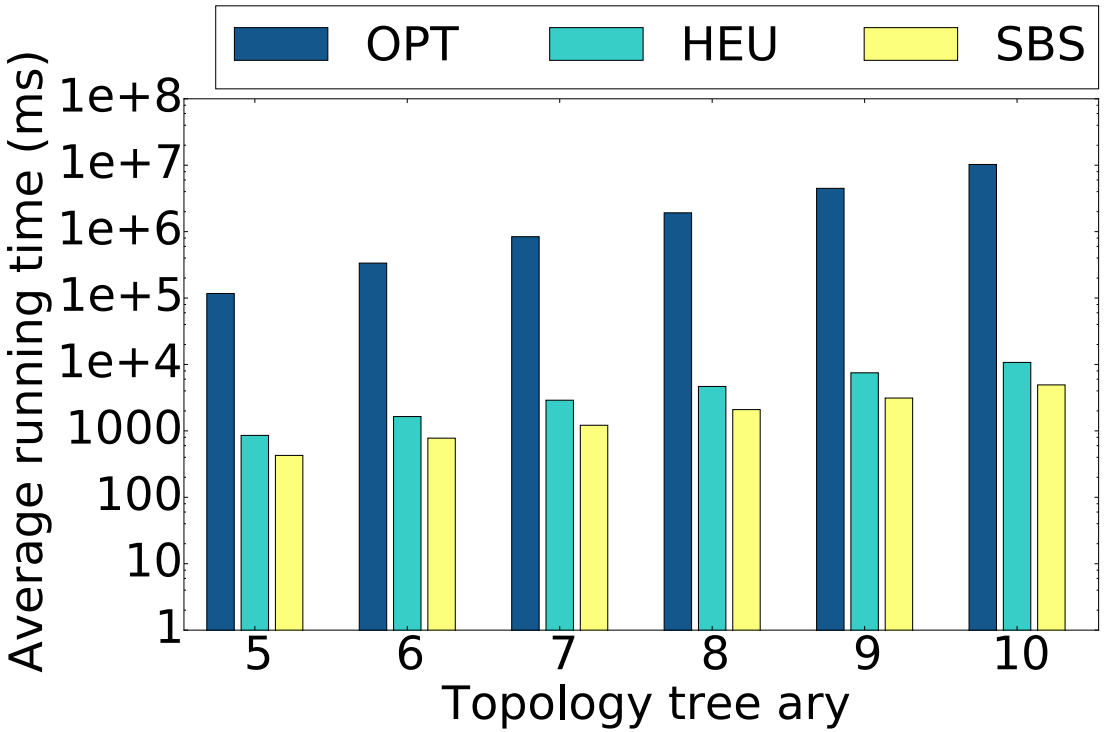
Simulation Results: Acceptance Ratio

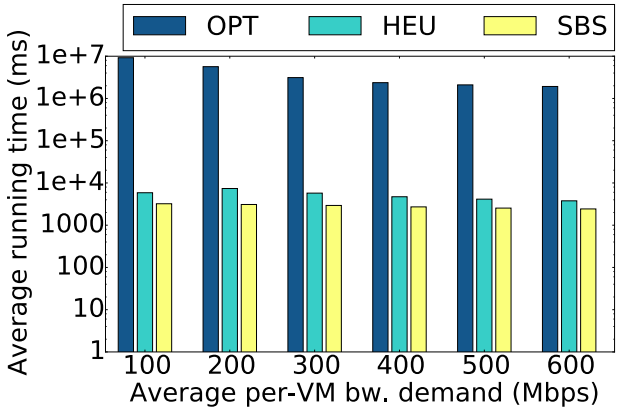


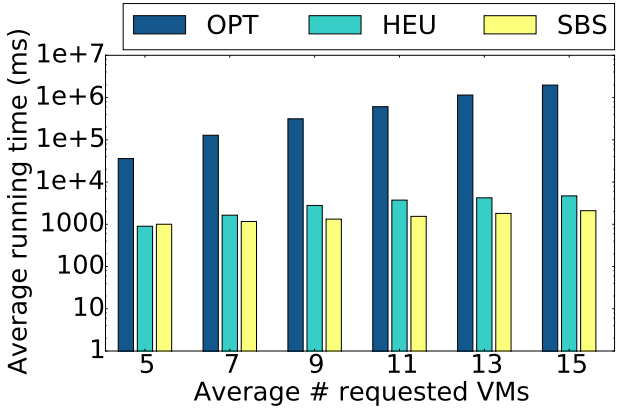




Simulation Results: Running Time







Outline

- Introduction and Motivation
- System Model and Algorithm Design
- Performance Evaluation
- **Conclusions**

Conclusions

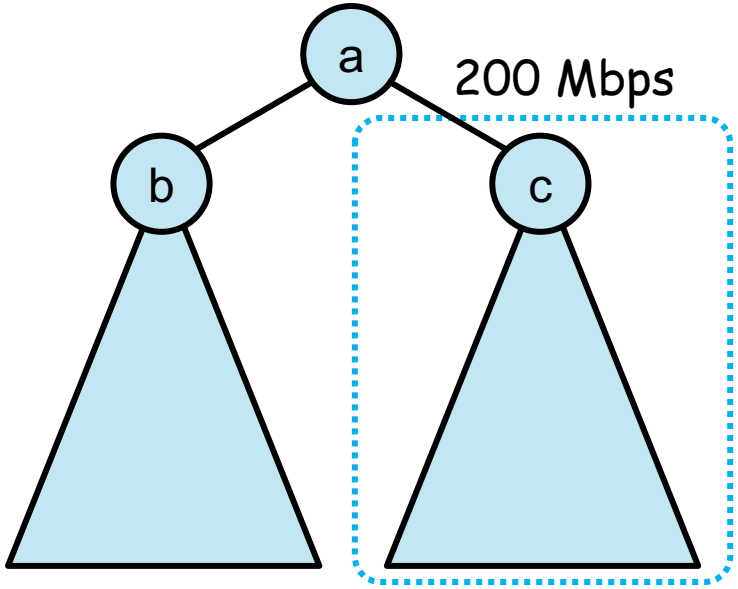
- A first study on Survivable VCE
  - A two-layer optimal DP algorithm
  - A faster near-optimal heuristic algorithm
- Discussions
  - Extension to tree-like topologies (FatTree, VL2, etc.)
  - Extension to cover a constant number of simultaneous failures
- Future work
  - SVCE on generic data center topologies (BCube, JellyFish, etc.)
  - Covering link failures in addition to PM failures

Q&A?

THANK YOU VERY MUCH!

Hose Model Bandwidth Guarantee

- Request $J = \langle N, B \rangle$
  - $N = 7, B = 100$ Mbps



Number of VMs T_c can offer (bandwidth constrained):

$n_c \in [0, 2] \cap [5, 7]$

DP in Details /2

- **Outer DP update**:
  - PM level:

$$N_h[n_0, n_1] = \begin{cases} n_0 & \text{if } n_1 = 0, n_0 \in [0, c_h] \cap \Lambda_h \\ \lambda_h & \text{if } n_1 = 0, n_0 \in \overline{\Lambda}_h, \lambda_h \le c_h \\ \infty & \text{otherwise} \end{cases} \quad (3)$$

Bandwidth feasible VMs

Lower bound of *upper bracket bw feasible VM*

Bandwidth infeasible VMs

  - Switch level:

$$N_v[n_0, n_1] = \begin{cases} N'_v[n_0, n_1, d_v] & \text{if } n_0, n_1 \in \Lambda_h \\ N'_v[\lambda_v, n_1, d_v] & \text{if } n_0 \in \overline{\Lambda}_h, n_1 \in \Lambda_h \\ N'_v[n_0, \lambda_v, d_v] & \text{if } n_0 \in \Lambda_h, n_1 \in \overline{\Lambda}_h \\ N'_v[\lambda_v, \lambda_v, d_v] & \text{if } n_0, n_1 \in \overline{\Lambda}_h \end{cases} \quad (4)$$

- **Inner DP update**:
  - No subtree:

$$N'_v[n_0, n_1, 0] = \begin{cases} 0 & \text{if } n_0 = n_1 = 0 \\ \infty & \text{otherwise} \end{cases}$$

  - k-th subtree:

$$N'_v[n_0, n_1, k] = \min_{\substack{n'_0, n''_0, \\ n'_1, n''_1}} \left\{ N'_v[n'_0, n'_1, k-1] + N_{u_k}[n''_0, n''_1] \;\middle|\; \begin{array}{l} n'_0 + n''_0 \ge n_0 \\ \min\{n'_0 + n''_1, n''_0 + n'_1\} \ge n_1 \end{array} \right\} \quad (6)$$

Work-through Example

- J = <2, 100 Mbps>