NetHint: White-Box Networking for Multi-Tenant Data Centers

Jingrong Chen, Hong Zhang, Wei Zhang, Liang Luo,
Jeffrey Chase, Ion Stoica, and Danyang Zhuo

Duke UNIVERSITY

Berkeley UNIVERSITY OF CALIFORNIA

W UNIVERSITY of WASHINGTON

Data-Intensive Applications Are Moving to The Cloud

Data Analytics

APACHE Spark™

hadoop MapReduce

Deep Learning

TensorFlow

HOROVOD

Reinforcement Learning

RAY

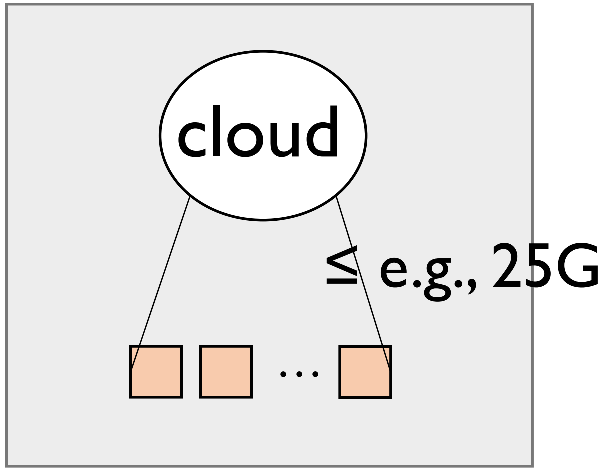
rllib

aws

Microsoft Azure

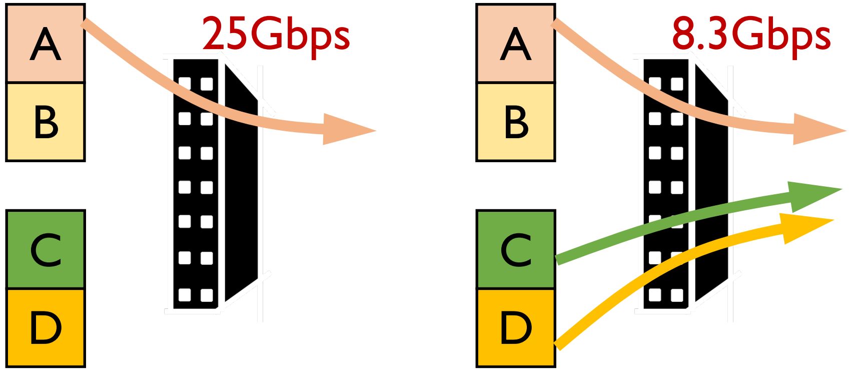
IBM Cloud

Today's Cloud Offers a "Black-Box" Abstraction

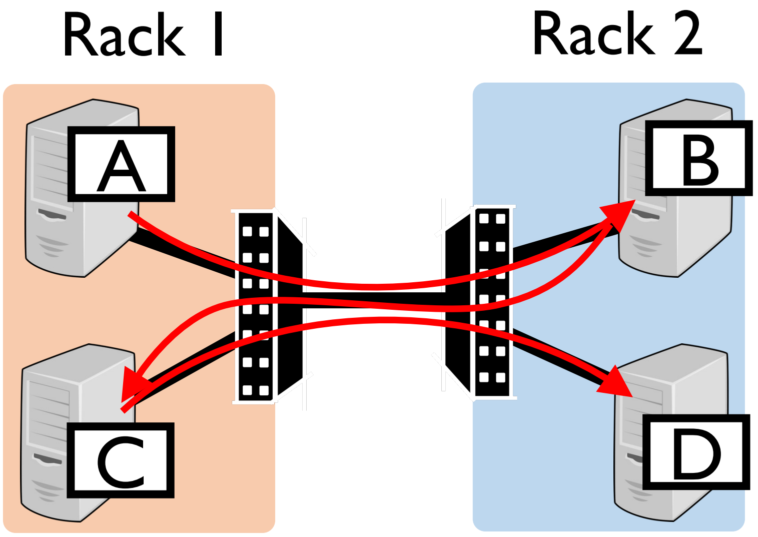


Black-Box Abstraction
for a tenant

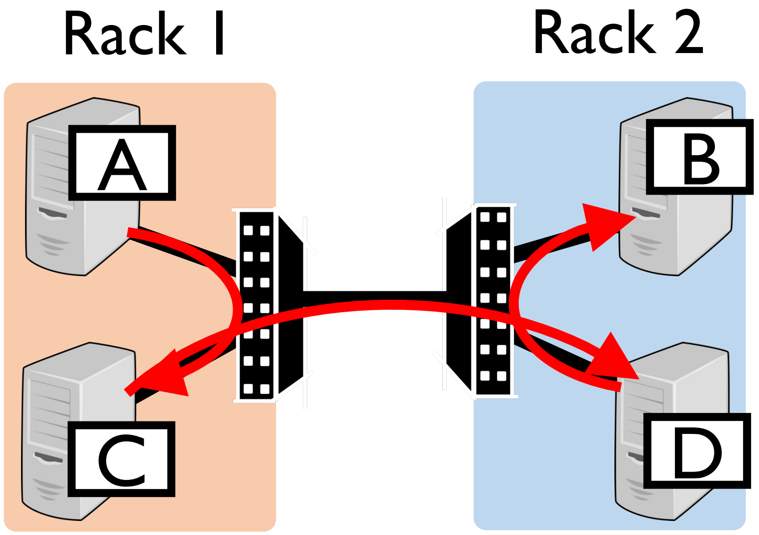
- Simple
- Tenants have minimum knowledge about the network performance
  - No link-layer topology
  - No instantaneous available bandwidth



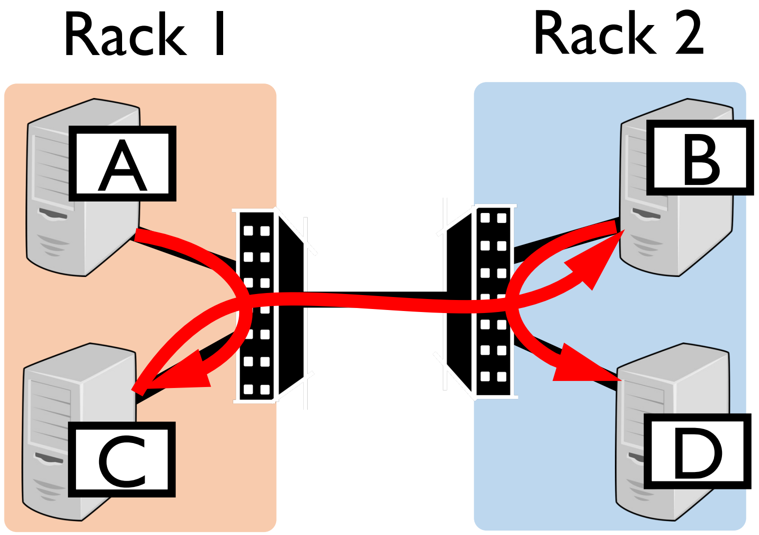
Data-Intensive Applications Can Adapt Traffic



Option 1

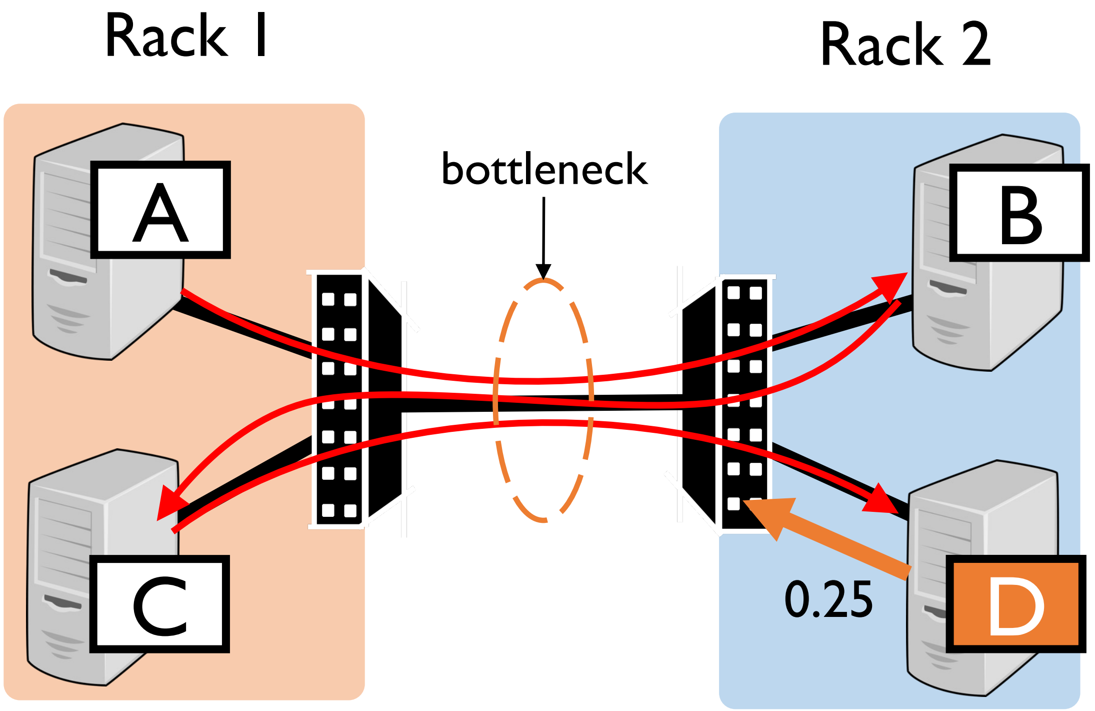


Option 2



Option 3

Data-Intensive Applications Have Incentive to Adapt Traffic

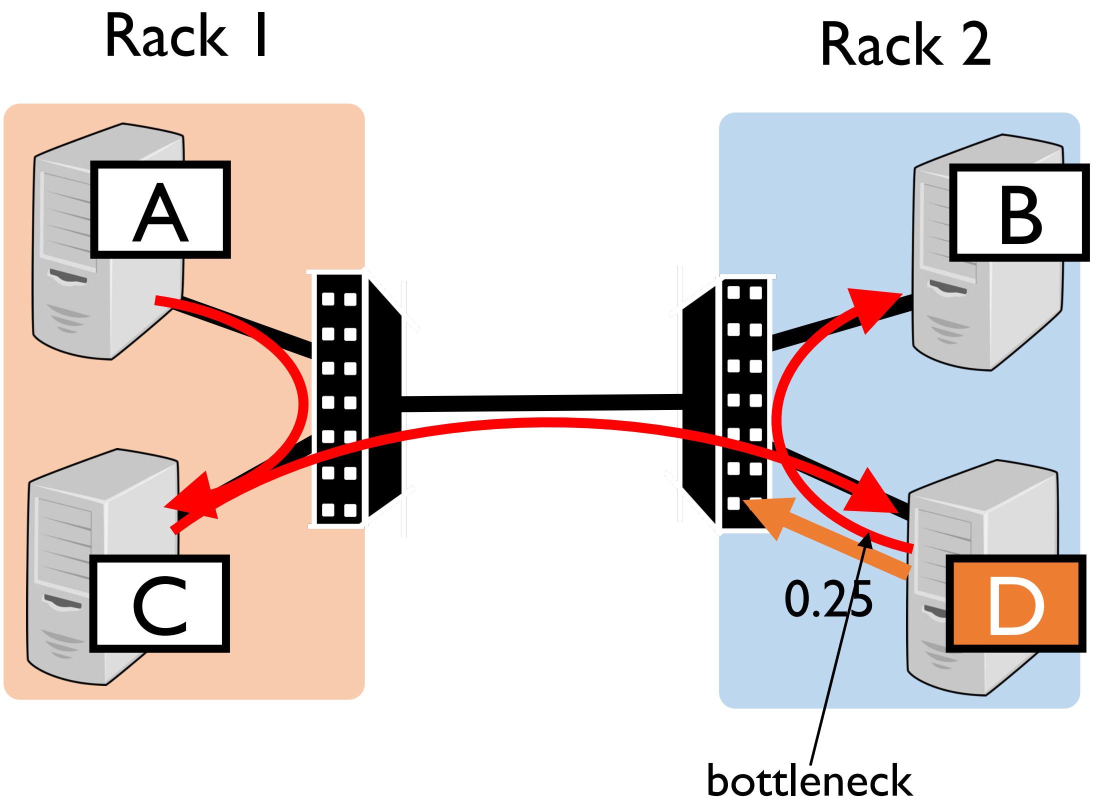


Case 1: Schedule with no information

Broadcast finish time

Case 1: 1 / 0.5 = 2

Data-Intensive Applications Have Incentive to Adapt Traffic



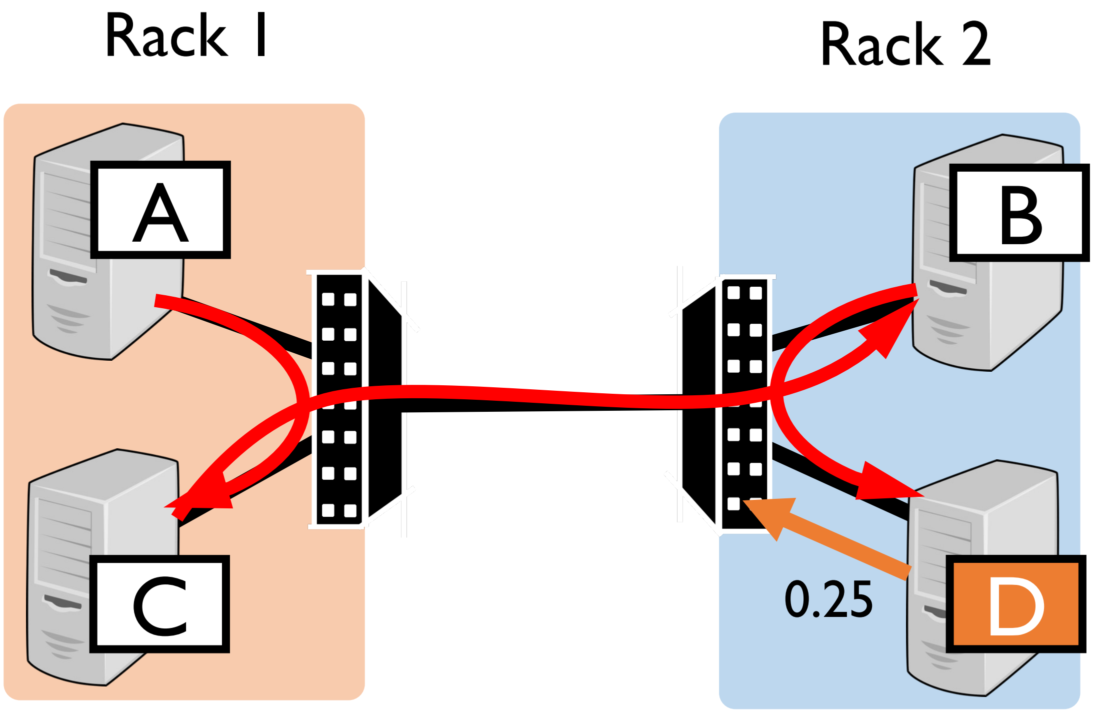
Case 2: Topology-aware schedule

Broadcast finish time

Case 1: 1 / 0.5 = 2

Case 2: 1 / 0.75 = 4/3

Data-Intensive Applications Have Incentive to Adapt Traffic



Case 3: Schedule with topology + bandwidth

Broadcast finish time

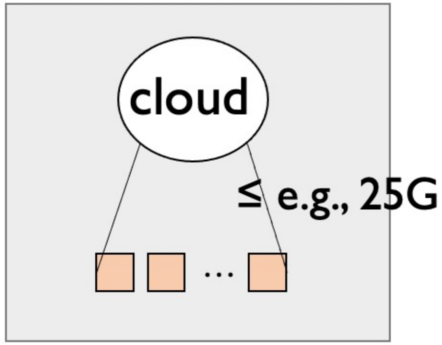
Case 1: 1 / 0.5 = 2

Case 2: 1 / 0.75 = 4/3

Case 3: 1 / 1 = 1 (optimal)

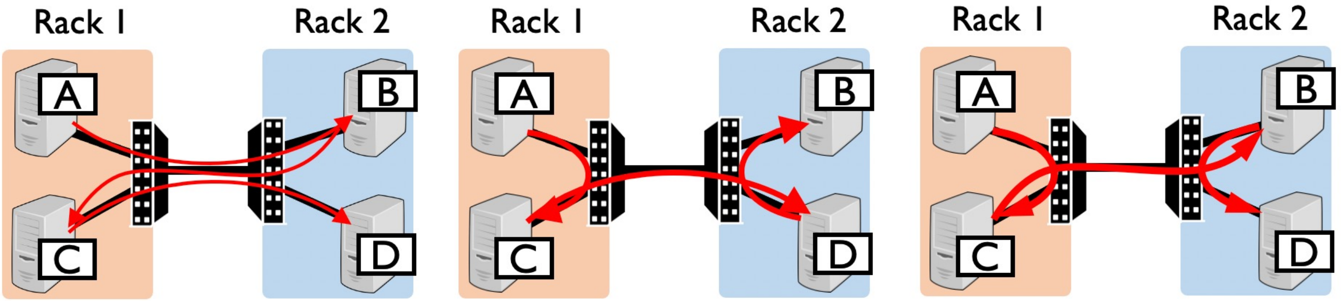
Mismatch!

- Black-Box networking abstraction does not provide *network characteristics*

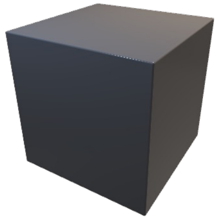


Black-Box Abstraction for a tenant

- Data-intensive applications have both the *incentive* and *ability* to **adapt** their transfer schedule based on *network characteristics.*



Can we address the mismatch without changing the black-box abstraction?



- Black-Box networking abstraction does not provide *network characteristics*

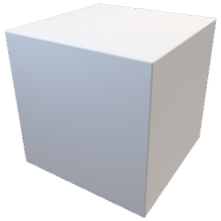
Mismatch!

User Probing

Tenants do traffic probing to profile the network performance

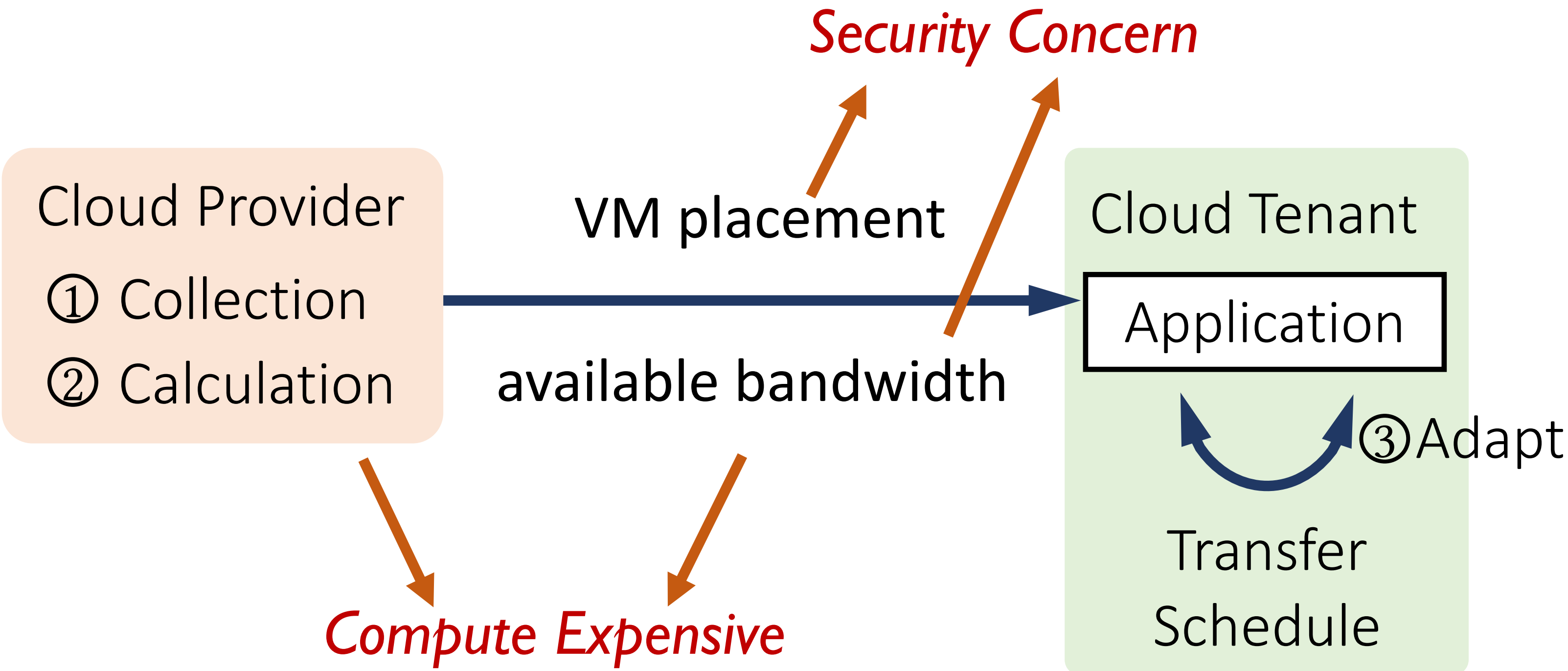
➤Costly: every app probes for itself

➤Slow: delay the start



A white-box approach to resolve this mismatch?

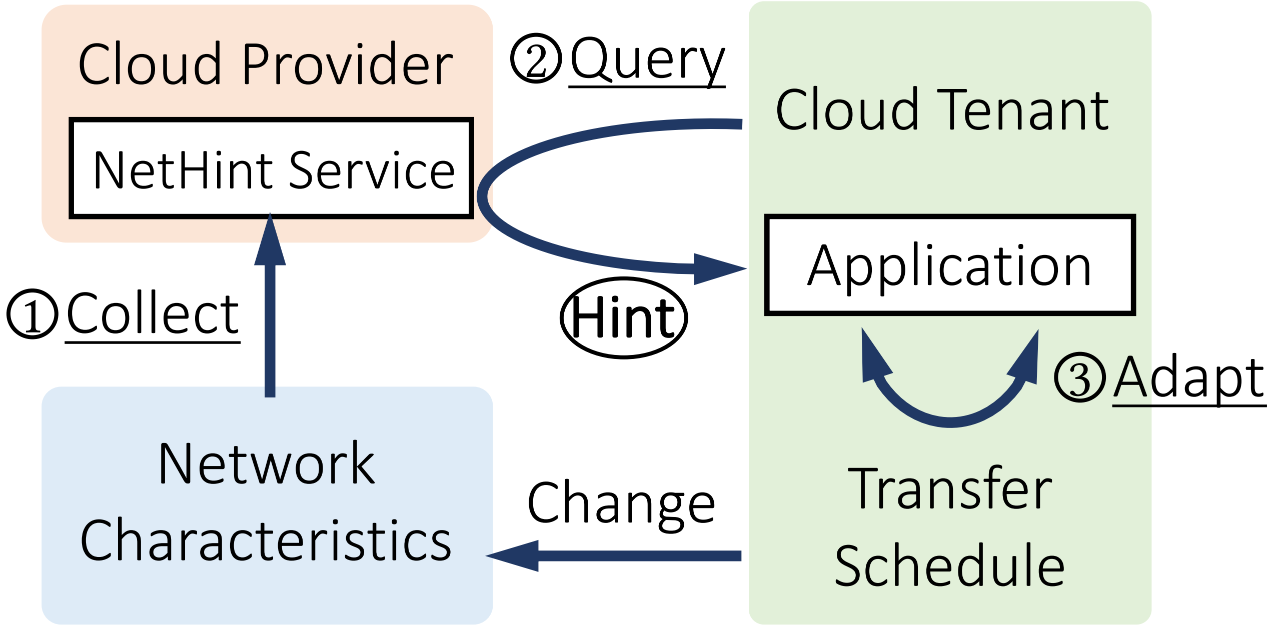
Strawman White-Box Solution



Cloud provider exposes some useful information to tenants

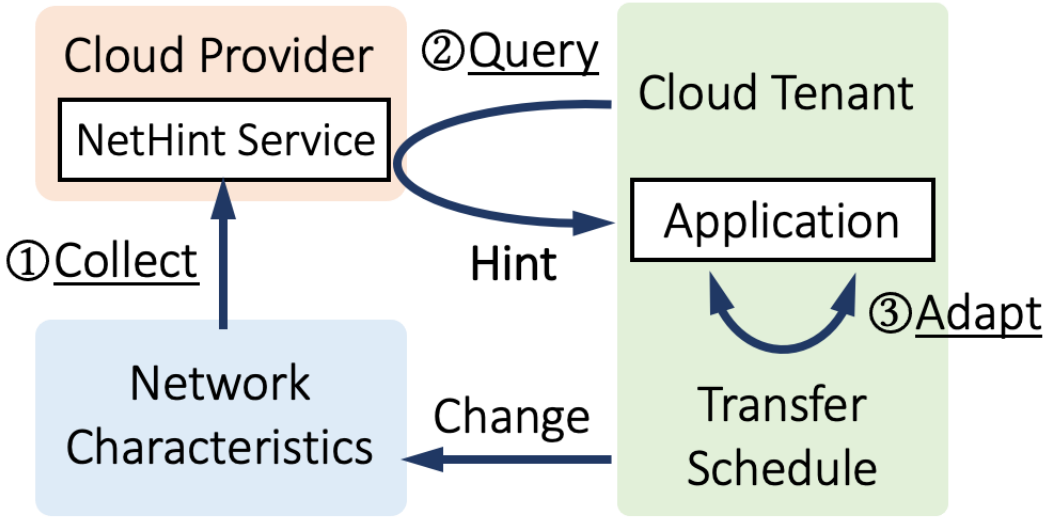
NetHint Overview

- An interactive mechanism between a cloud tenant and its provider to jointly enhance the application performance



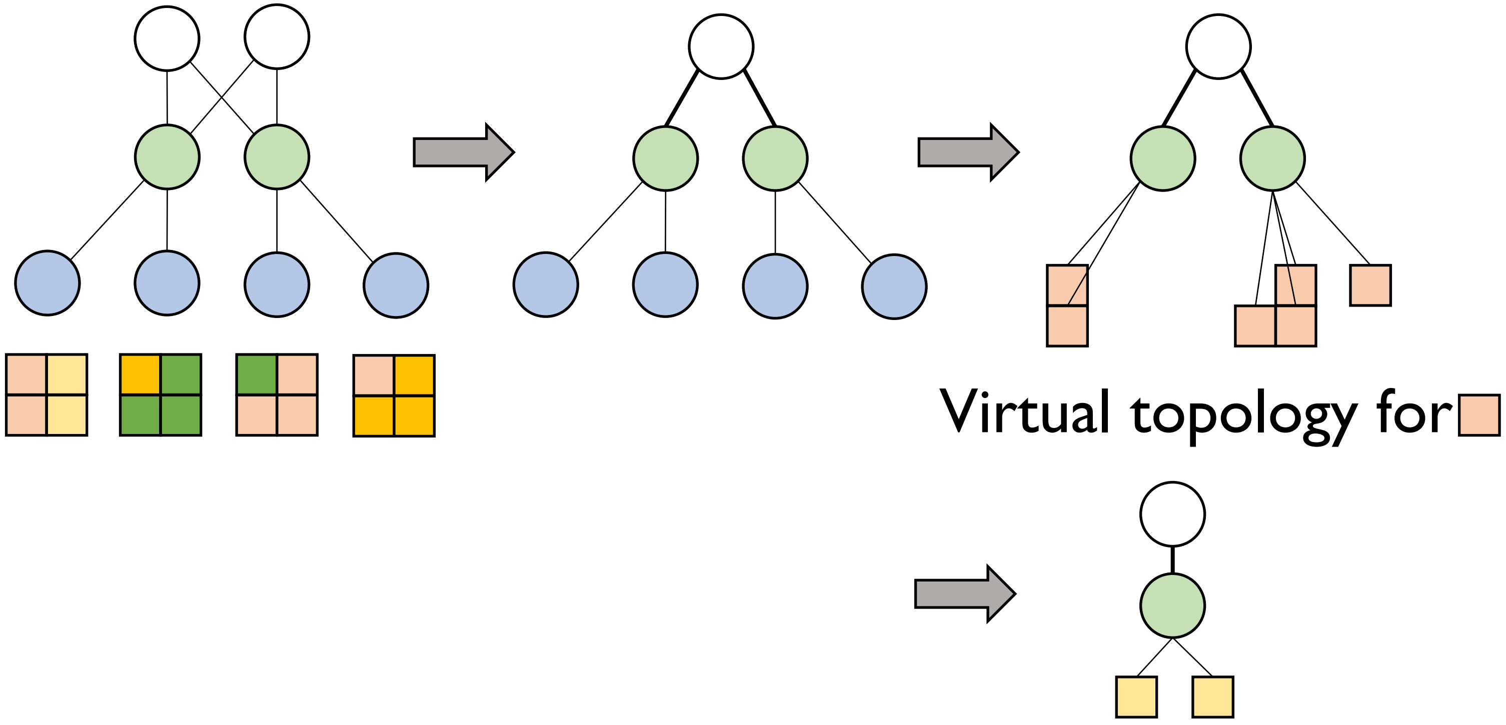
Questions to Answer

- ➢ What hints to provide?
- ➢ How to provide hints with low cost?
- ➢ How should applications adapt their traffic?



What Is in the Hint?

- Reflect locality of instances
- A hierarchical virtual topology **T** for a cloud tenant.

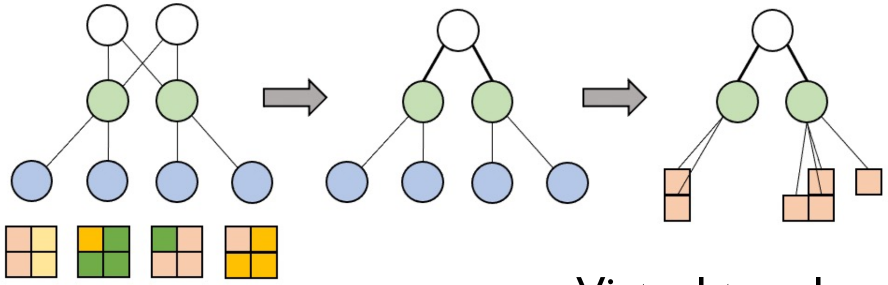


Virtual topology for 🟧

Virtual topology for 🟨

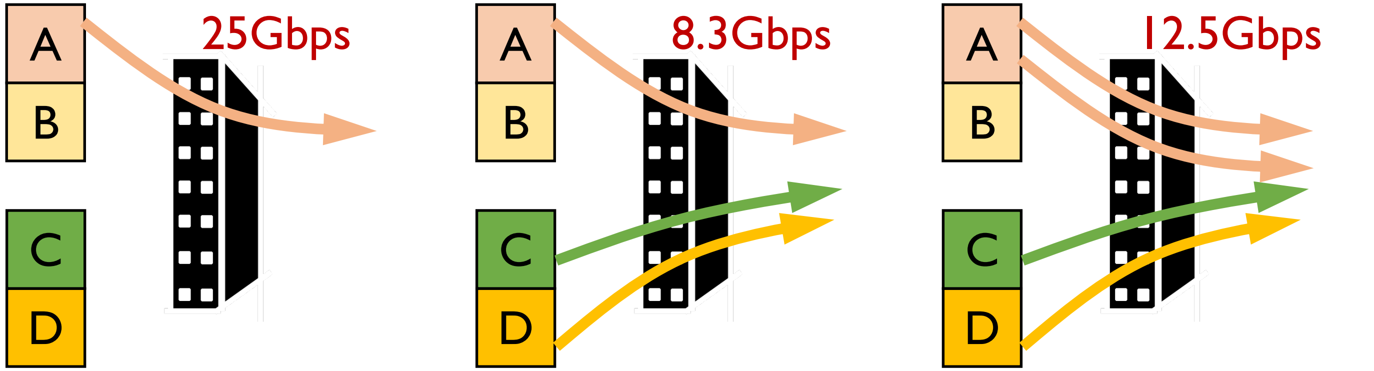
What Is in the Hint? – Cont'd

- A virtual topology $\mathbf{T}$ for a cloud tenant.
- Network utilization on each link $\boldsymbol{l}$
  - Total bandwidth $\boldsymbol{B_t}$ on link $\boldsymbol{l}$



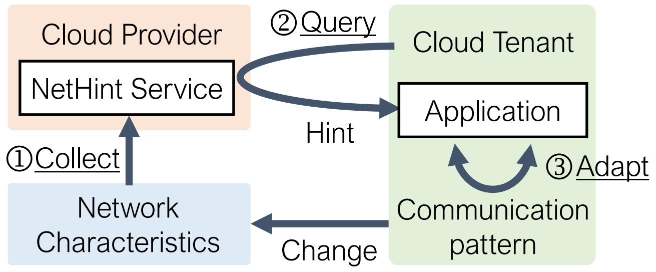
Virtual topology

1. ~~All flows~~ (security) ☹
2. Residual bandwidth $\boldsymbol{B_r}$ on link $\boldsymbol{l}$ (not accurate) 🤔
3. $\boldsymbol{B_r}$ + Number of competing flows $\boldsymbol{n}$ sharing the same link $\boldsymbol{l}$ 🙂



Timely NetHint with Low Cost

- NetHint collects network metrics periodically
- In each period, collect once for all tenants
- Hierarchical all-gather; all-to-all only among racks
- We set the information update period to 100ms



Overhead of NetHint’s Monitoring Plane

- Each CPU core emulates a rack

Allgather

# Racks	CPU Util. (%)	Memory (MB)	Latency (ms)
6	0.06	4.5	10.6
24	0.14	5.9	10.7
96	0.41	19.3	11.9
240	0.66	78	13.7

Adapting Transfer Schedules with NetHint

- Collective communication
  - Data-parallel deep learning
  - Reinforcement learning
  - Serving ensemble models

- Task placement
  - Data-analytics frameworks
  - Task-based distributed systems

Other Questions to Answer

- Applications calculation/adaptation latency?
- Highly dynamic network conditions?

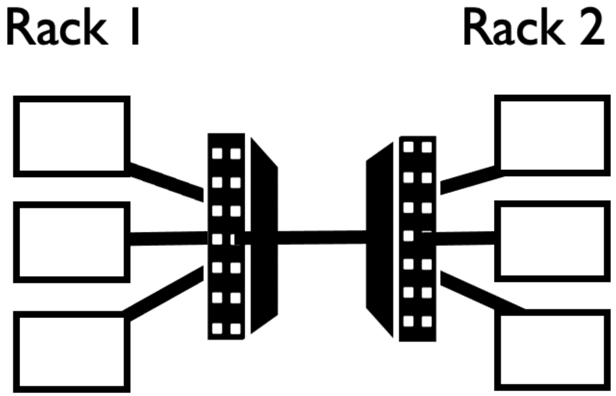
Stale Hints?

- Bandwidth estimation noises?
- Herd behavior?

How do they affect app performance?

Evaluation

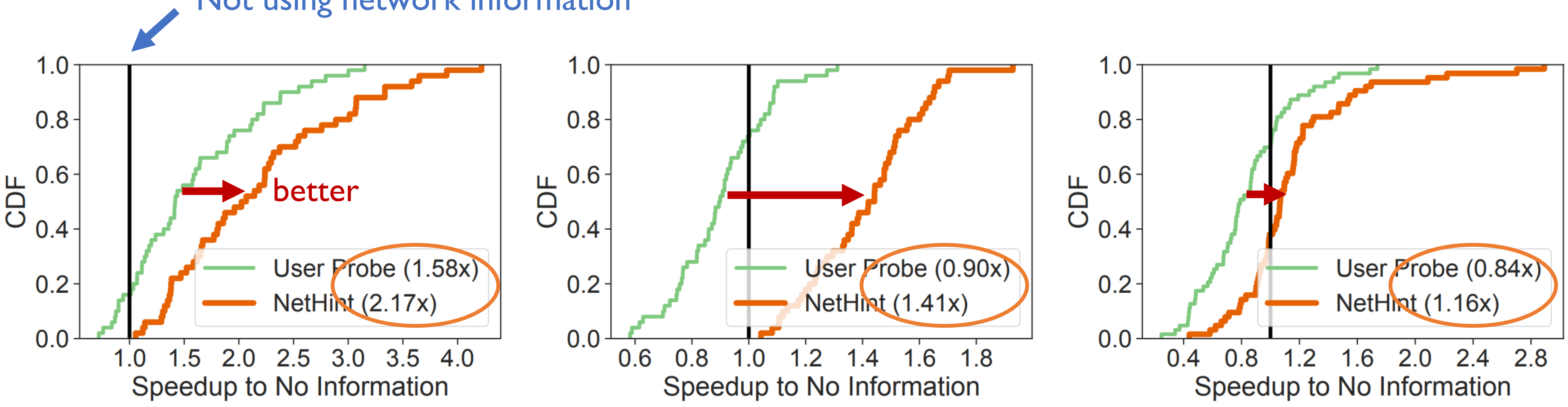
- Testbed setup
  - 6 servers, 40G
  - 2 racks, oversubscription: 3
  - Each machine run 4 VMs, 10G



- Baselines:
  - Not using network information
  - User probing
    - N hosts, N/2 rounds.
    - Each round, 10000 packets (Plink) or 1 second (Choreo), whichever is smaller

NetHint on Testbed

Not using network information



(a) Distributed deep learning

(b) Ensemble model serving

(c) MapReduce

Summary

- Black-box networking abstraction and adaptiveness of data-intensive applications create a mismatch.
- NetHint: an interactive mechanism between cloud provider and tenants to jointly optimize application performance.
  - 2.2x, 1.4x, 1.2x improvement on Deep Learning, Model Serving, and MapReduce
  - NetHint is available at https://github.com/crazyboycjr/nethint

Thank you!

Contact jingrong.chen@duke.edu

Future Directions & Discussions

Deployment

- Integration with cloud provider: CloudLab, hybrid enterprise cloud
- Integration into real applications: Spark, Ray, etc.

Algorithmic (NetHint for public cloud)

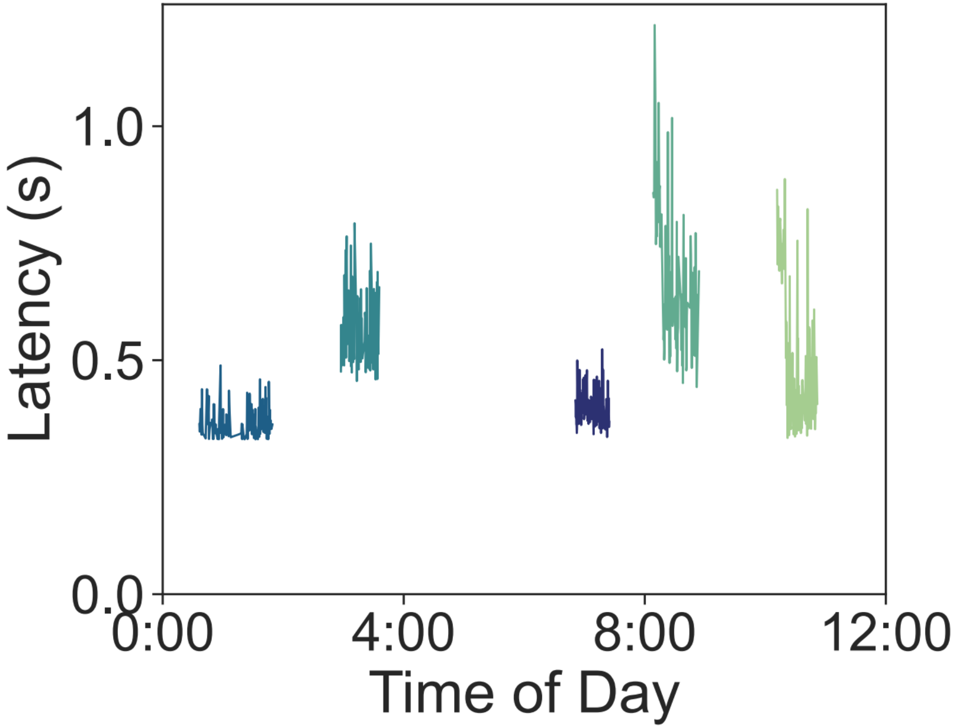
- Security & Competitive concerns
- Herd Behavior

Open-source (flow-simulator)

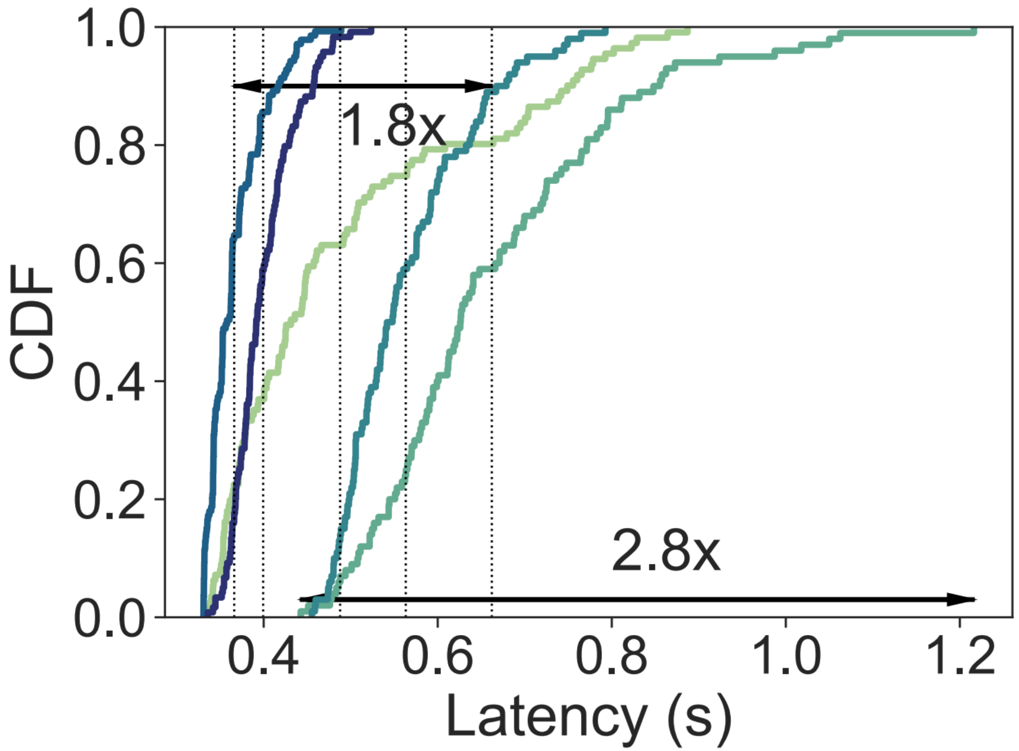
- Make it faster (utilize GPU?)
- Support more fairness models
- Support more datacenter topologies (when tree does not apply)

Backup

Allreduce latencies vary both across time and VM allocations



Allreduce latency across time

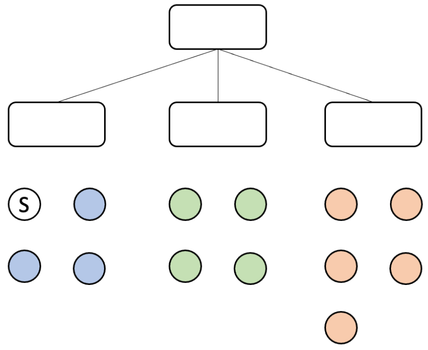


CDF

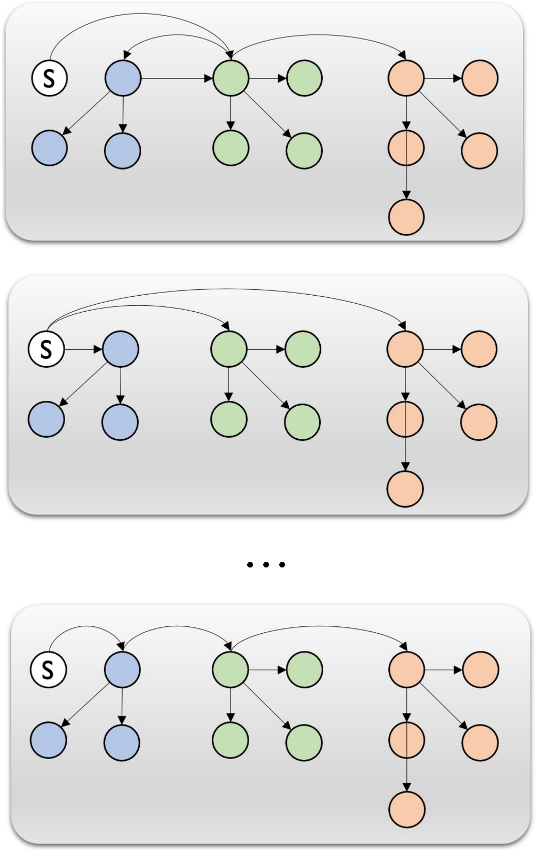
Backup

Adapting Transfer Schedules with NetHint

- Challenge #3: How should applications adapt transfer schedules?
- Broadcast



3 racks, S: source of data



- Sample a random set of broadcast tress
- Each crosses the rack only once
- Using linear programming to optimize the weight of each broadcast tree

Backup

Stale Information

- Network can be highly dynamic
- Application can choose to use a hint for a longer period

Policy

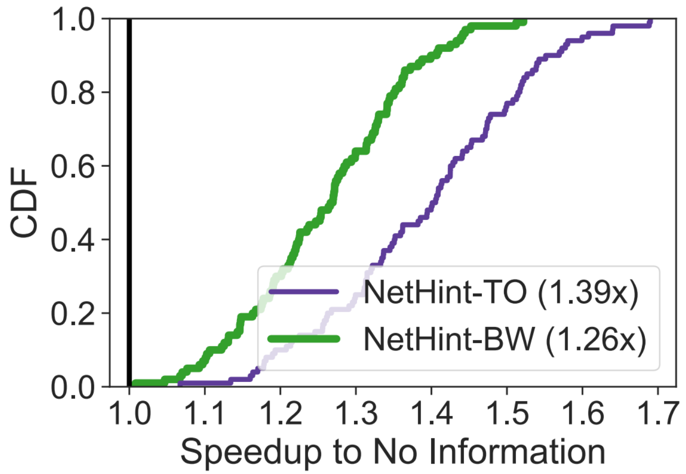
NetHint-BW	(Use both bandwidth and topology information)
NetHint-TO	(More stable Topology Only information)

Backup

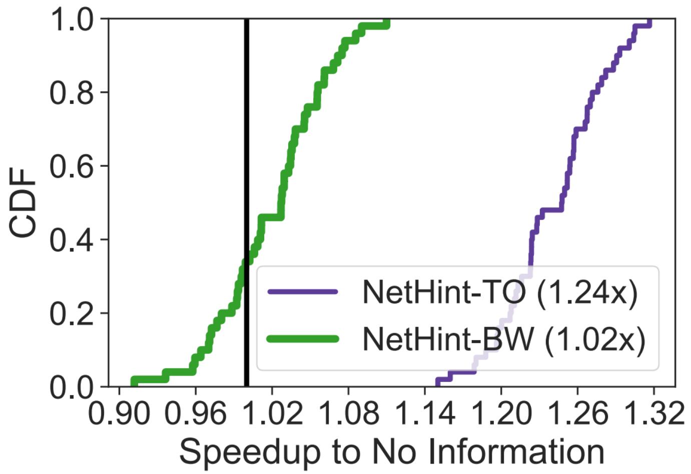
Stale Information

1) Workload granularity is large

2) Overhead of computing a transfer schedule is non-negligible



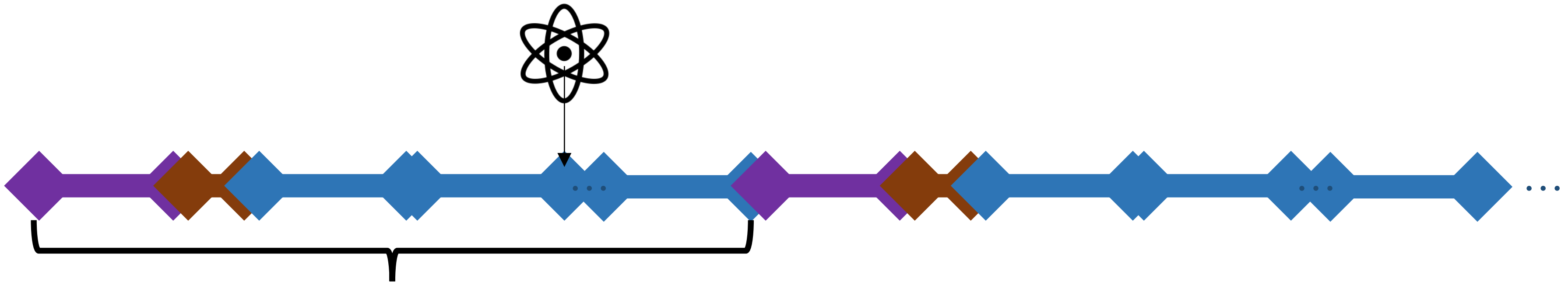
Coarse-grained workload



Non-negligible overhead to compute a transfer schedule

Backup

Stale Information



Staleness = Ta + Tu

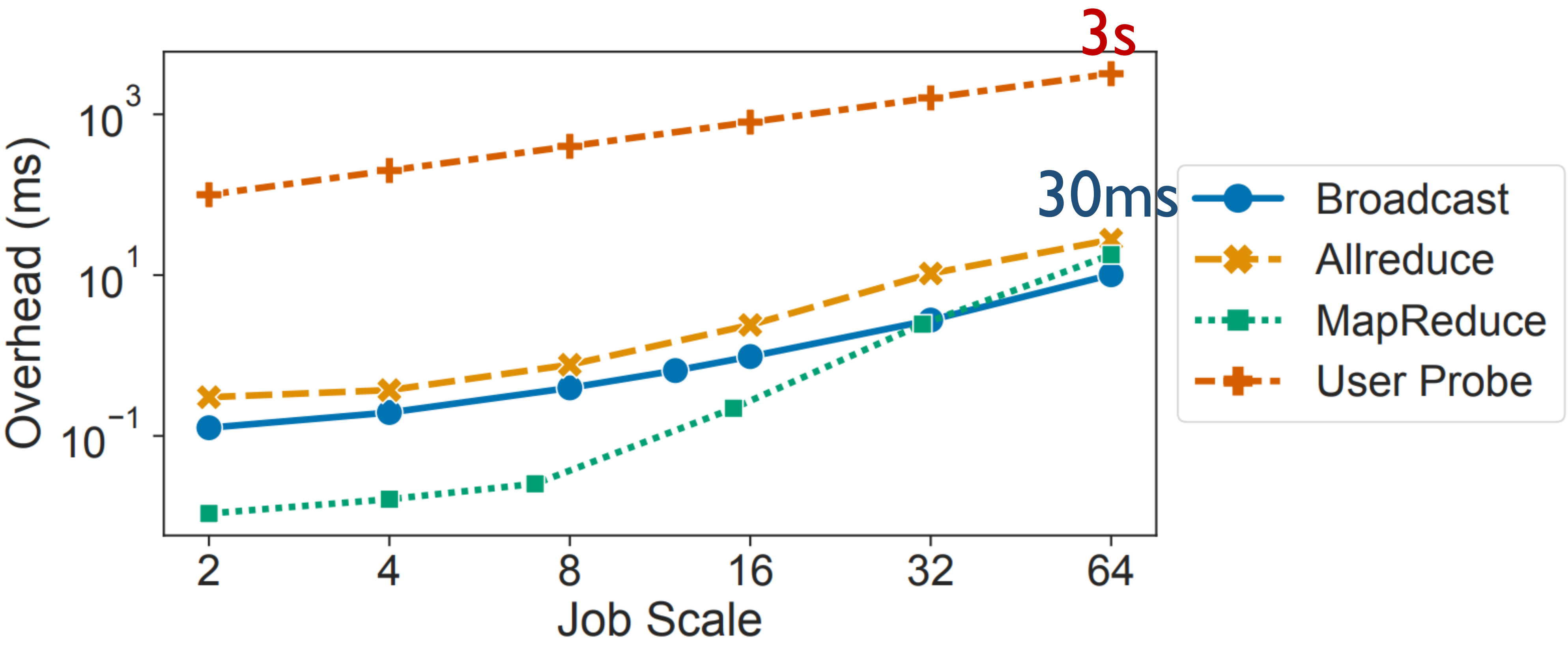
⟷ Provider collects hint
⟷ Fetch Hint & Compute a transfer schedule } Ta
⟷ A transfer schedule is used   Tu
⚛ Network condition change   Tb

Policy

Staleness < Tb → NetHint-BW (Use bandwidth Information)

Staleness ≥ Tb → NetHint-TO (More stable Topology Only information)

Latency to Compute Transfer Schedules is Low



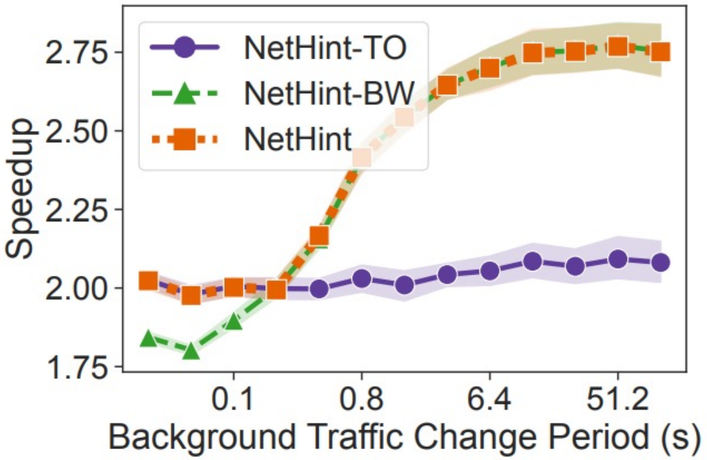
Backup

NetHint Can Choose among the Best

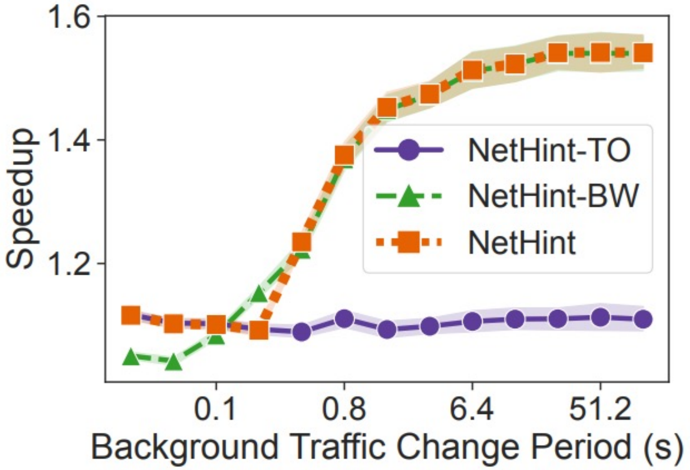
Not to use bandwidth information (NetHint-TO)

1) Workload granularity is large

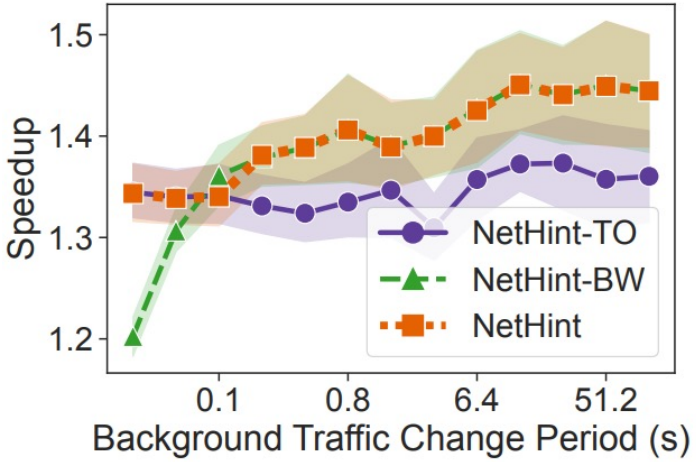
2) Overhead of computing a transfer schedule is non-negligible



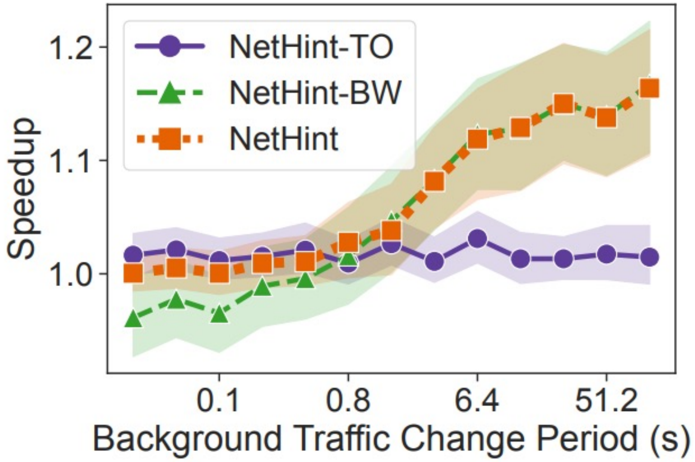
Deep Learning (oversub=3)



Deep Learning (oversub=1.5)

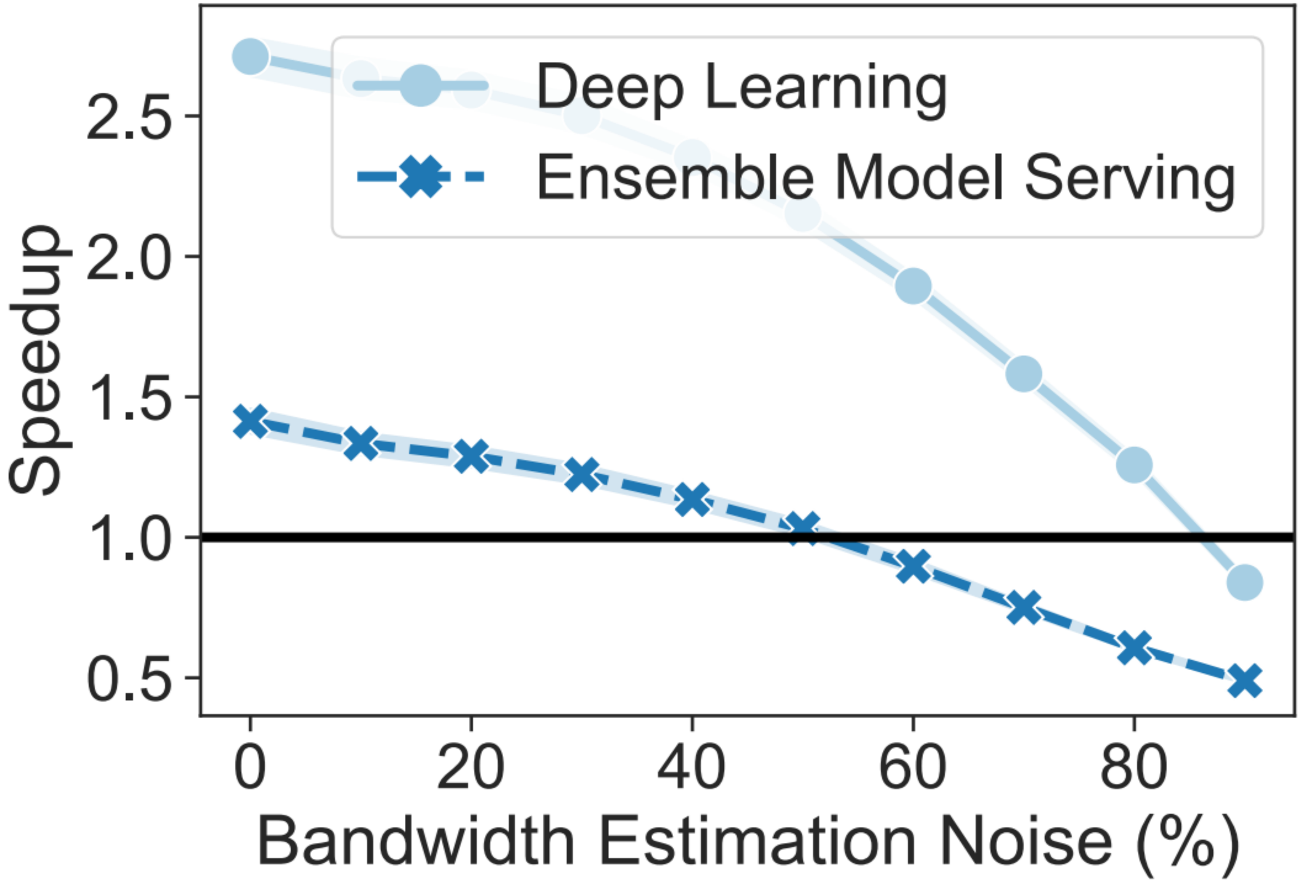


Model Serving (oversub=3)

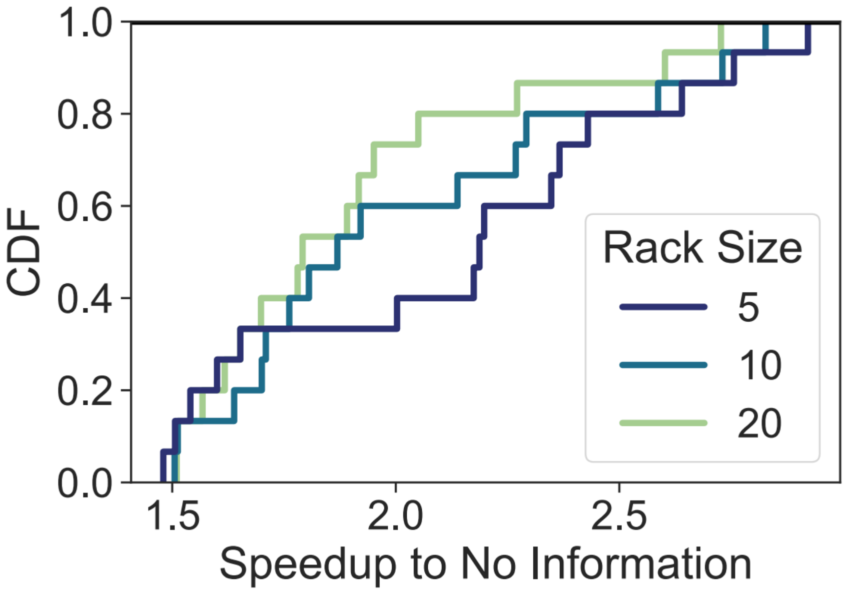


MapReduce (oversub=3)

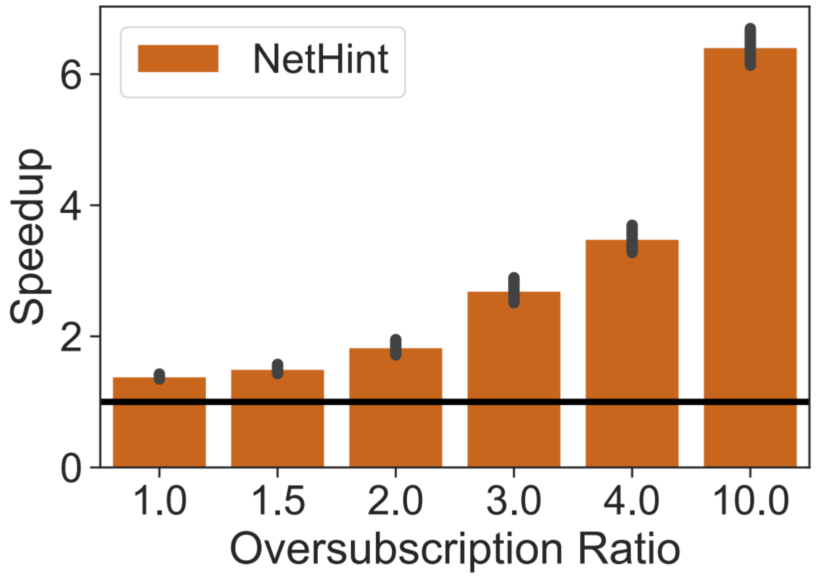
Sensitivity Analysis



Sensitivity Analysis



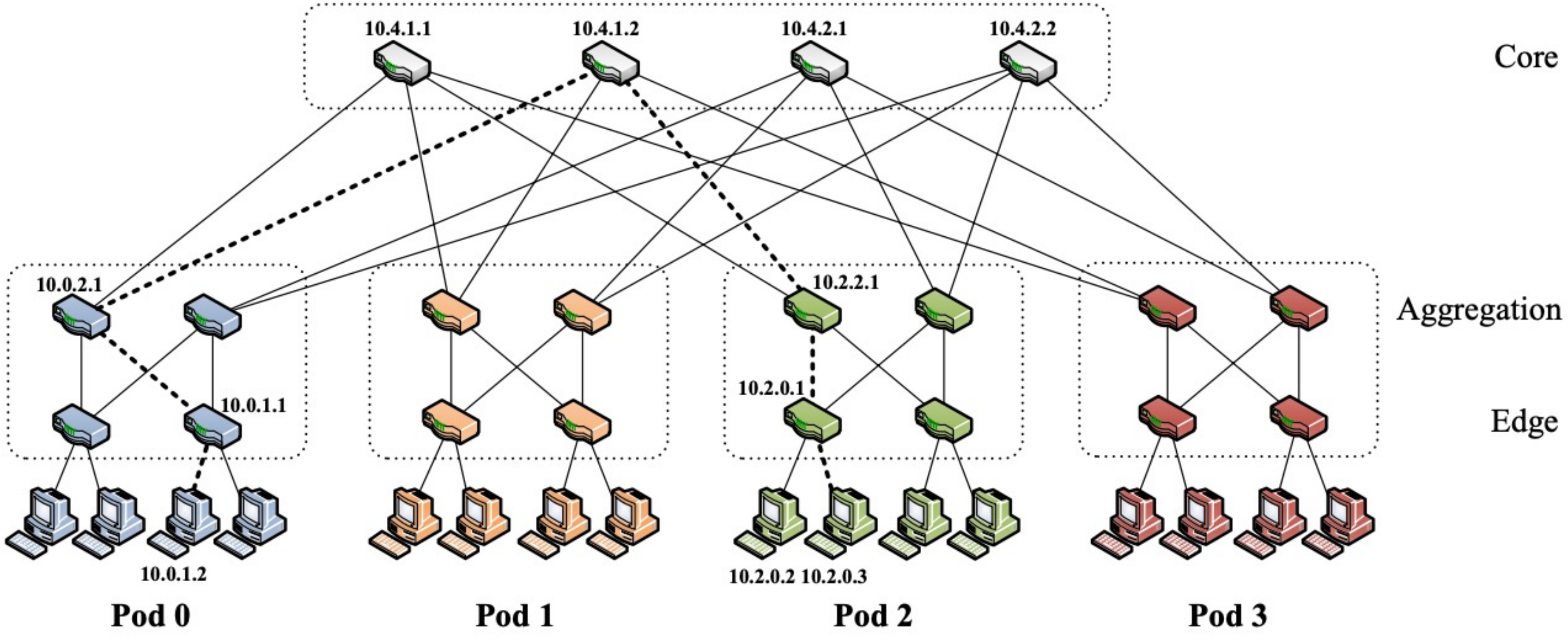
(a) Number of machines per rack



(b) Oversubscription ratios

Background knowledge

- Datacenter topology



A Scalable, Commodity Data Center Network Architecture