Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure

Qiao Zhang[1], Guo Yu[2], Chuanxiong Guo[3],

Yingnong Dang[4], Nick Swanson[4], Xinsheng Yang[4], Randolph Yao[4], Murali Chintalapati[4], Arvind Krishnamurthy[1], Tom Anderson[1]

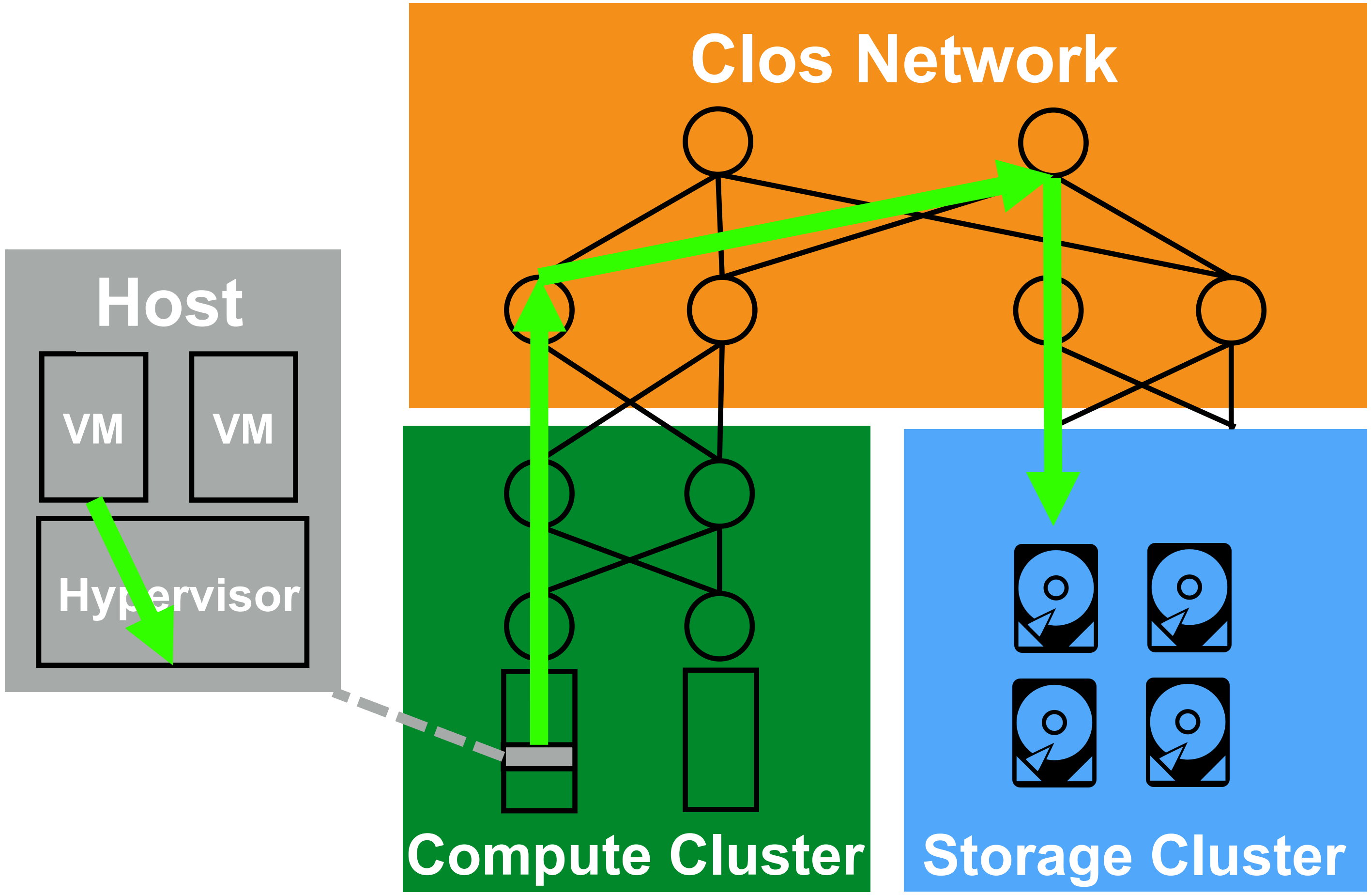
[1]University of Washington, [2]Cornell University, [3]Toutiao (Bytedance), [4]Microsoft

VM Availability

- IaaS is one of the largest cloud services today
- High VM availability is a key performance metric
- Yet, achieving 99.999% VM uptime remains a challenge

1. **What is the availability bottleneck?**
2. **How to eliminate it?**

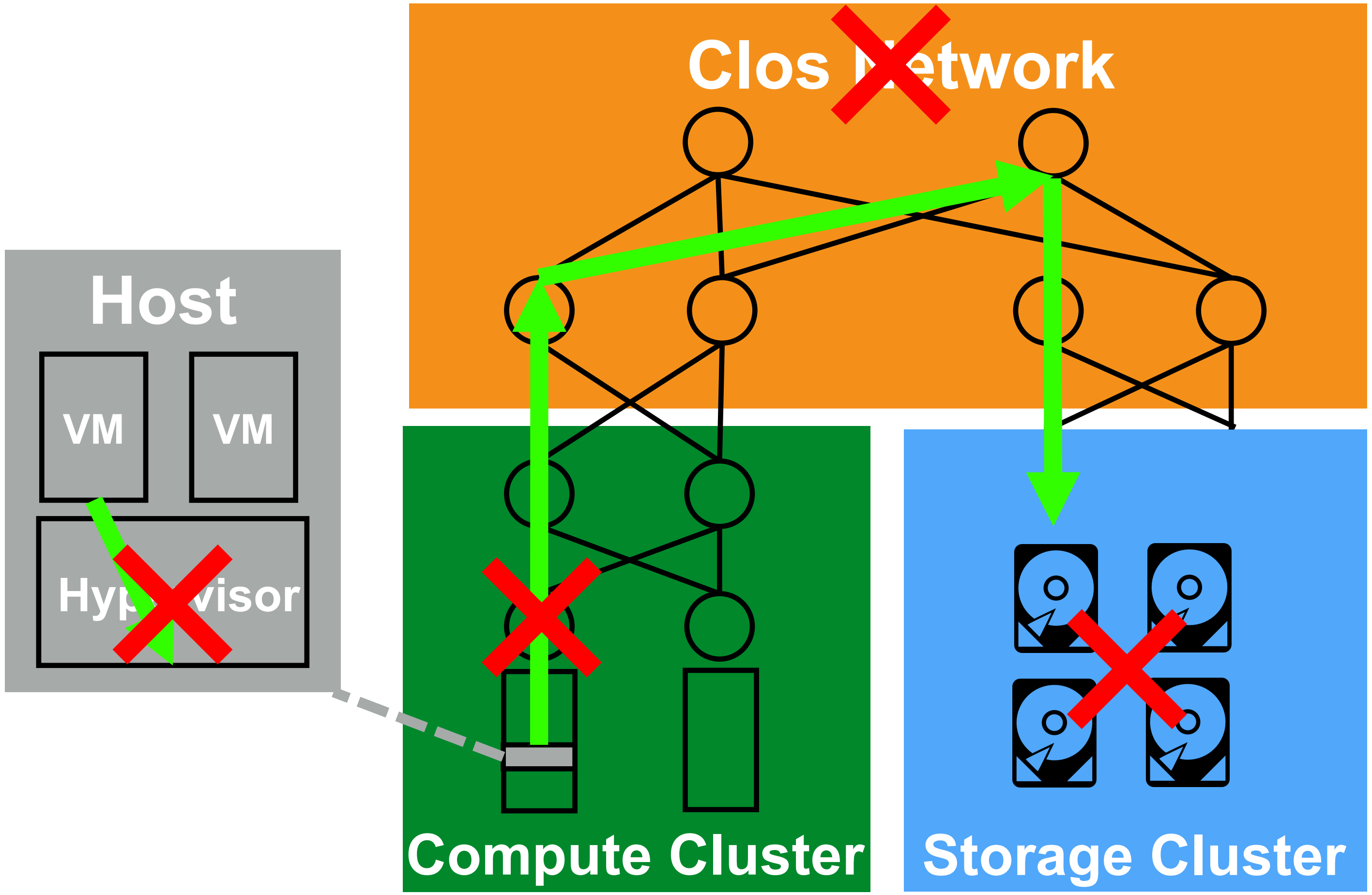
Azure IaaS Architecture



Subsystems inside a Datacenter

- Compute and storage clusters with a Clos-like network
- **Compute-storage Separation**
  - VMs and Virtual Hard Disks (VHDs) from different clusters
  - Hypervisor transparently redirects disk access
- Data survive compute rack failure

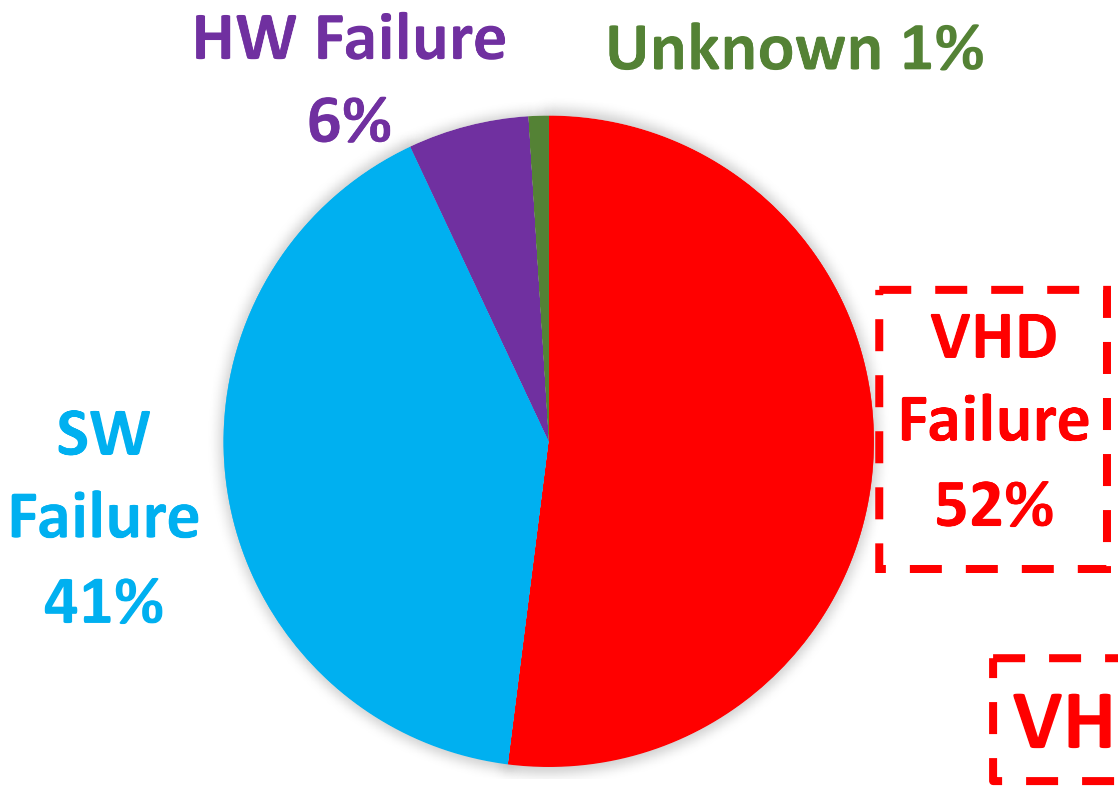
A New Type of Failure: VHD Failures



Subsystems inside a Datacenter

- Infra failures can disrupt VHD access
- Hypervisor can retry, but not indefinitely
- Hypervisor will eventually **crash the VM**
- Customers then take actions to keep their app-level SLAs

How much do VHD failures impact VM availability?



Breakdown of Unplanned VM Downtime in a Year

VHD failures:

- 52% of unplanned VM downtime
- Tens of minutes to hours to localize

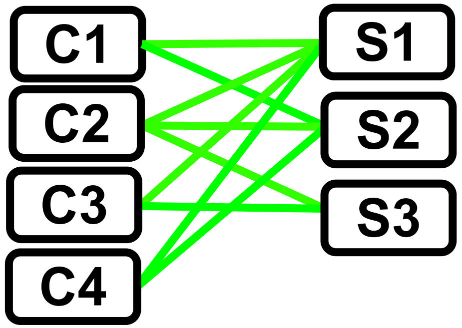
VHD failure localization is the bottleneck

Failure Triage was Slow and Inaccurate

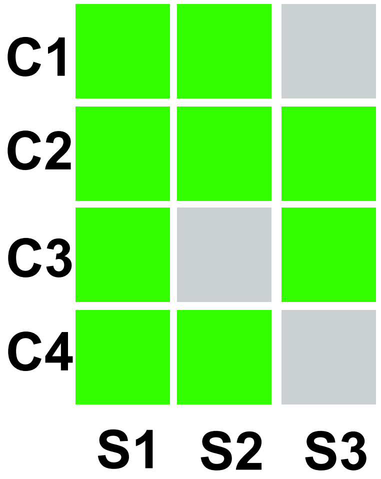
- Each team checks their subsystem for anomalies to match the incident
  - e.g., host heart-beats, storage perf-counters, link discards
- Incidents get ping-ponged due to false positives
  - Inaccurate and slow diagnosis
- Gray failures in network and storage are hard to catch
  - Troubled but not totally down
  - Only fail a subset of VHD requests
  - Can take hours to localize

Deepview Approach: Global View

- Isolate failures by examining interactions between subsystems
  - Instead of alerting every team
- Bipartite model
  - Compute Clusters (left) : Storage Clusters (right)
  - Edge if VMs from compute cluster mount VHDs from a storage cluster
  - Edge weight = VHD failure rate



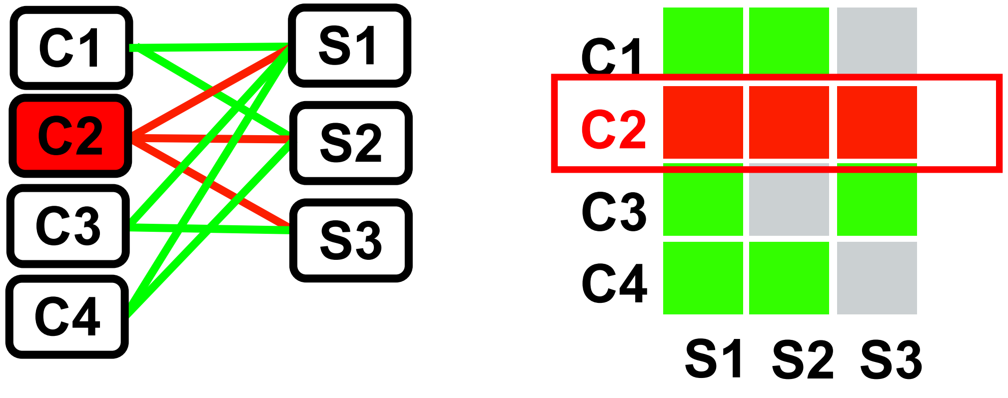
Bipartite Model



Grid View

Deepview Approach: Global View

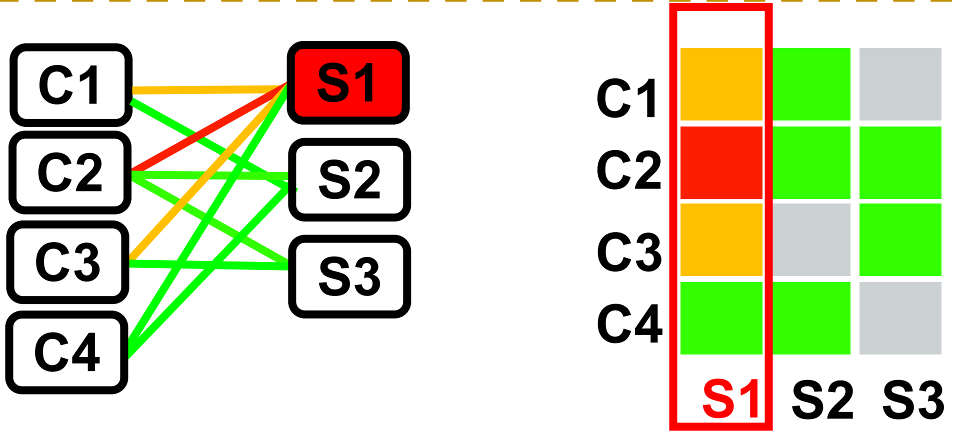
Example Compute Cluster Failure



Compute Cluster
C2 failed

C2 Failure
Grid View

Example Storage Cluster Failure



Storage Cluster
S1 Failed

S1 Gray Failure
Grid View

Azure measurements revealed many common failures patterns

Challenges

Remaining challenges:

1. Need to locate network failures → Generalized model
2. Need to handle gray failures → Lasso + Hypothesis testing
3. Need to be near-real-time → Streaming data pipeline

Summary of our goal:

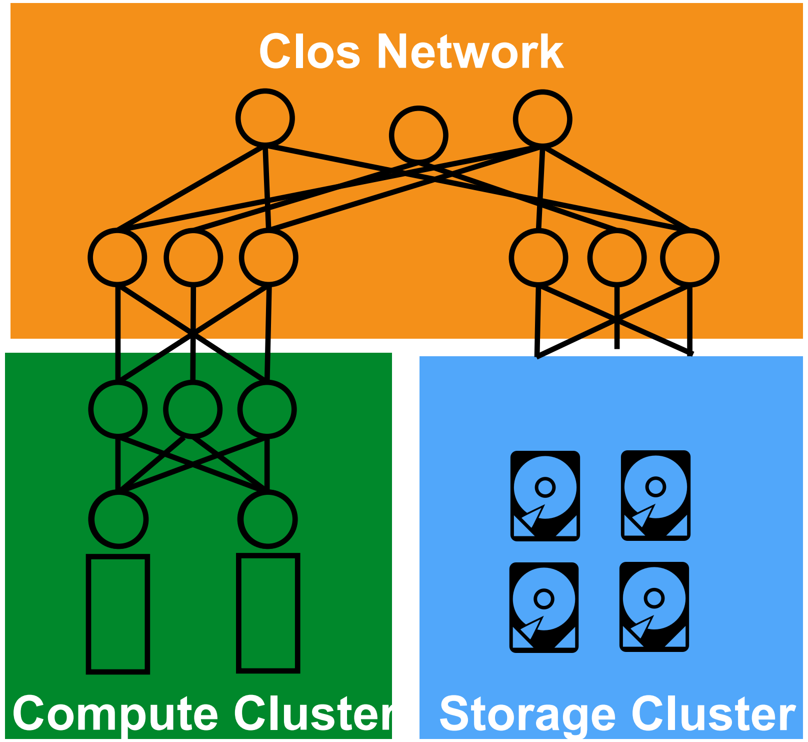
A system to localize VHD failures to underlying failures in compute, storage or network subsystems within a time budget of 15 minutes

Time budget set by production team to meet availability goals

Outline

- Global View Approach
- Model & Algorithm
- System
- Evaluation
- Architectural Lessons
- Related Work

Deepview Model: Include the Network



- Need to handle multipath & ECMP
- Simplify Clos network to a tree by aggregating network devices
- Can model at the granularity of clusters or racks

Deepview Model: Estimate Component Health

$$\mathbf{Prob(path\ i\ is\ healthy)} = \prod_{j \in path(i)} \mathbf{Prob(component\ j\ is\ healthy)}$$

Assume independent failures

Blue: observable
Red: unknown
Purple: topology

$$1 - \frac{e_i}{n_i} = \prod_{j \in path(i)} p_j$$

e_i=num of VMs crashed
n_i=num of VMs

$$\log\left(1 - \frac{e_i}{n_i}\right) = \sum_{j \in path(i)} \log p_j$$

System of Linear Equations

Component j is healthy with

$p_j = \exp(\beta_j)$

- $\beta_j = 0$, clear component j
- $\beta_j \ll 0$, may blame it

$$\boxed{y_i = \sum_{j=1}^{N} \beta_j\, x_{ij} + \varepsilon_i}$$

$y_i=\log\left(1 - \frac{e_i}{n_i}\right)$
$\beta_j=\log p_j$
ε_i=measurement noise

Deepview Algorithm: Prefer Simpler Explanation via Lasso

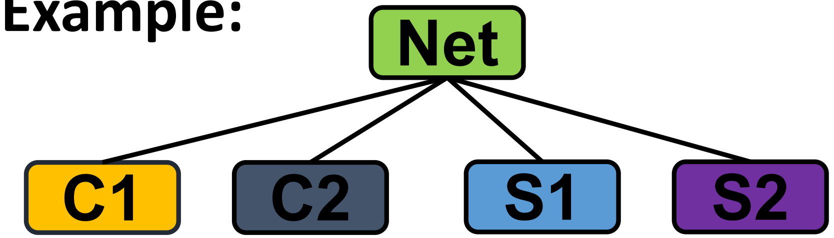
$$y_i = \sum_{j=1}^{N} \beta_j \, x_{ij} + \varepsilon_i$$

- Potentially, #unknowns > #equations
- Traditional least-square regression would fail

- But multiple simultaneous failures are rare
- Encode this domain knowledge mathematically?

- Equivalent to prefer most β_j to be zero
- **Lasso regression** can get sparse solutions efficiently

Example:



$$\begin{aligned}
y_1 &= \beta_{c1} + \beta_{net} + \beta_{s1} + \varepsilon_1 \\
y_2 &= \beta_{c1} + \beta_{net} + \beta_{s2} + \varepsilon_2 \\
y_3 &= \beta_{c2} + \beta_{net} + \beta_{s1} + \varepsilon_3 \\
y_4 &= \beta_{c2} + \beta_{net} + \beta_{s2} + \varepsilon_4
\end{aligned}$$

Lasso Objective Function:

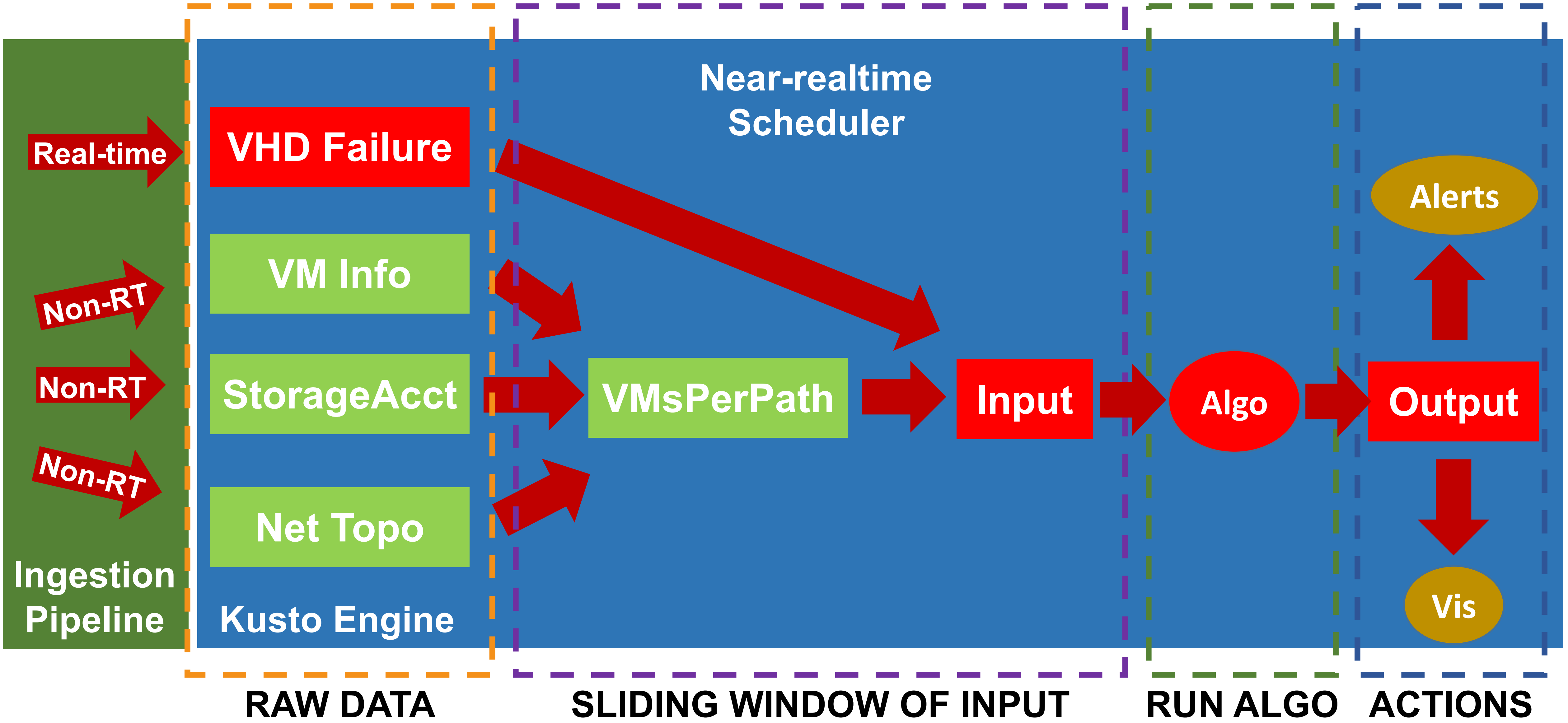
$$\hat{\beta} = \underset{\beta\in\mathbb{R}^N, \beta\le 0}{\operatorname{argmin}} \|y - X\beta\|^2 + \underbrace{\lambda \, \|\beta\|_1}_{\text{Sparsity}}$$

Deepview Algorithm: Principled Blame Decision via Hypothesis Testing

- Need a binary decision (**flag/clear**) for each component
- Ad-hoc thresholds do not work reliably
- **Can we make a principled decision?**

- *If estimated failure probability worse than average, then likely a real failure*
- **Hypothesis test**: $\mathbf{H_0(j)\colon \beta_j = \bar{\beta}}$ **vs.** $\mathbf{H_A(j)\colon \beta_j < \bar{\beta}}$
- If reject $\mathrm{H_0(j)}$, blame component j; otherwise, clear it

Deepview System Architecture: NRT Data Pipeline



Outline

- Global View Approach
- Model & Algorithm
- System
- Evaluation
- Architectural Lessons
- Related Work

Evaluation

Deepview has been deployed in production at Azure

1. How well can it localize VHD failures in production?

2. How accurate is the algorithm compared to alternatives?

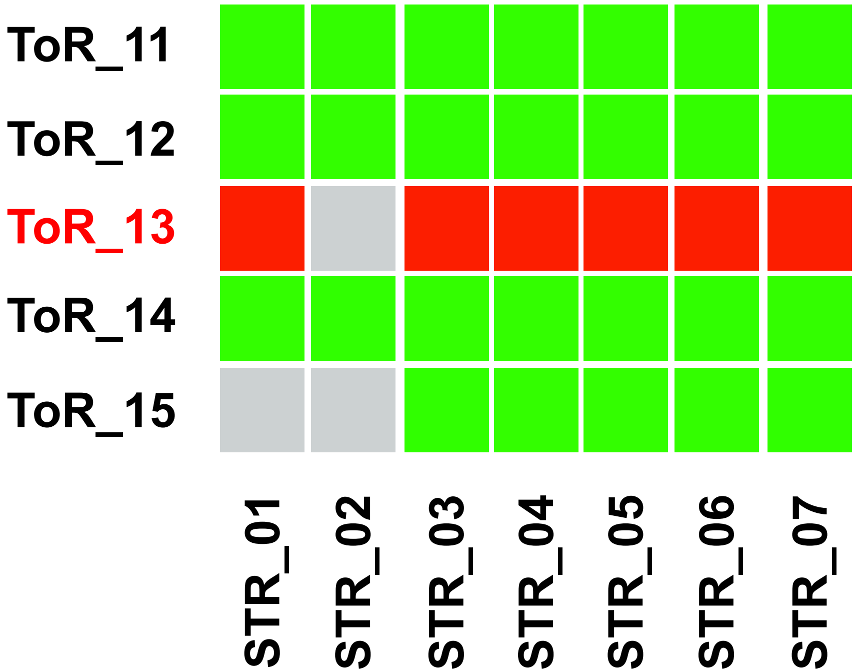
3. How fast is the system?

Some Statistics

- Analyzed Deepview results for one month
  - Daily VHD failures: hundreds to tens of thousands
- Detected 100 failures instances
  - 70 matched with existing tickets, 30 were previously undetected
- Reduced unclassified VHD failures to less than a max of 500 per day
  - Host failures or customer mistakes (e.g., expired storage accounts)

Case Study 1: Unplanned ToR Reboot

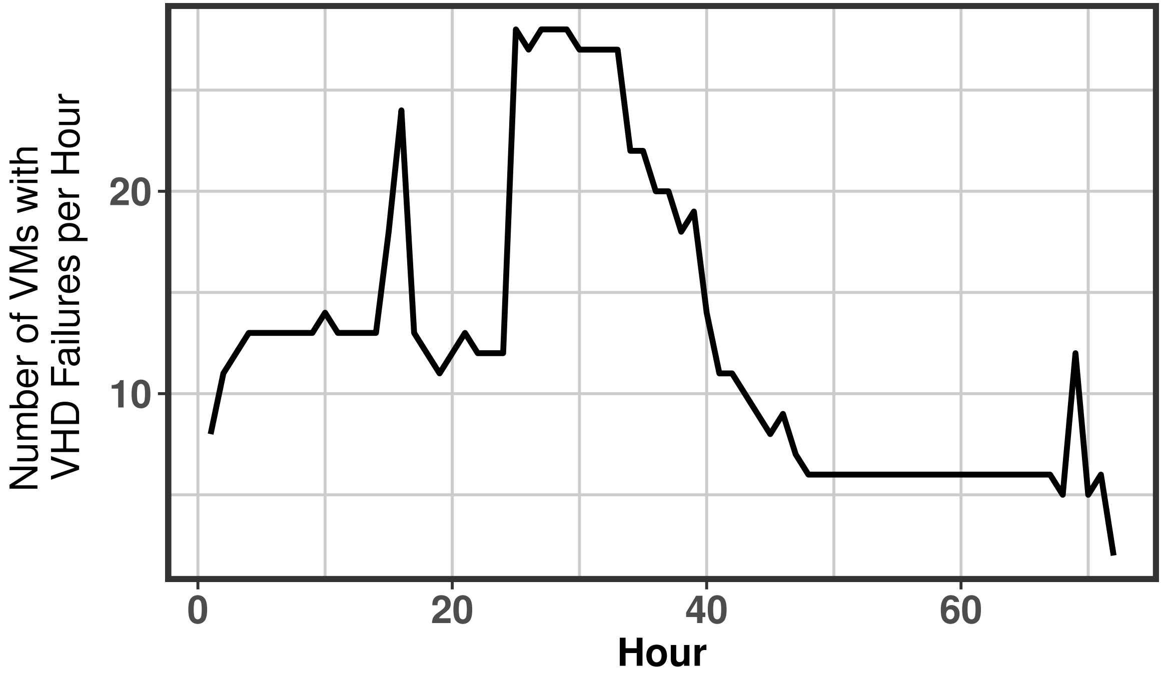
- Unplanned ToR reboot can cause VM crashes
- Know this can happen, but not where and when
- Deepview can flag those ToRs
- Associate VM downtime with ToR failures
- **Quantify the impact of ToR as a single-point-of-failure on VM availability**



Blamed the right ToR among 288 components

Case Study 2: Storage Cluster Gray Failure

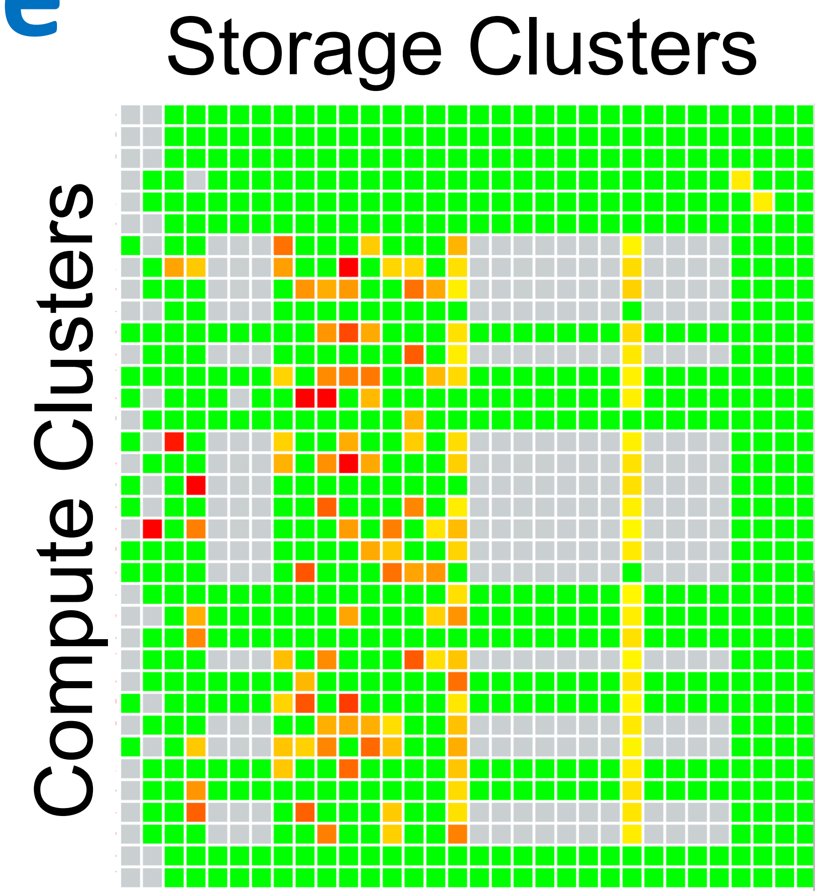
- A storage cluster was brought online with a bug that puts some VHDs in negative cache
- Deepview flagged the faulty storage cluster almost immediately while manual triage took 20+ hours



Number of VMs with VHD Failures per Hour during a Storage Cluster Gray Failure

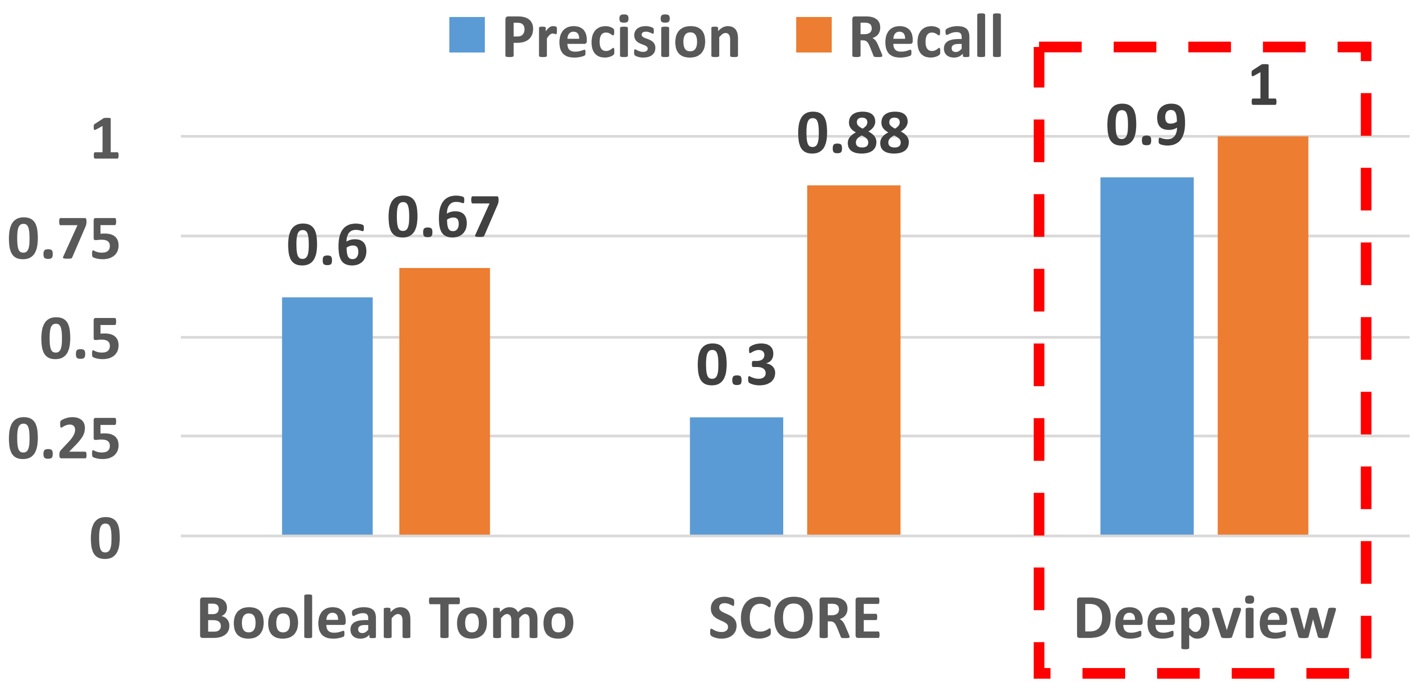
Case Study 3: Network Failure

- Network outages are rare, but do happen
- In an incident, many top tier links were mistakenly turned off, causing large capacity loss
- When storage replication traffic hit, it caused huge packet losses and many VMs to crash
- Deepview pinpointed the misbehaving aggregate switches



A Network Failure due to Top Tier Link Capacity Loss

Algorithm Accuracy Comparison



- Two other tomography algorithms: **Boolean-Tomo** and **SCORE**
  - Greedy heuristics to find minimum set of failures
- Use production trace from 42 incidents
  - 16 Compute, 14 Storage, 10 ToR, 2 Net

Deepview Time to Detection

- **Time to detection (TTD)**
  - Time from incident start to failure localized
  - Estimate start time from VHD failure event timestamp

- Deepview's TTD is under 10 min
  - Data ingestion takes ~3.5 min
  - ~5 minutes sliding window delay
  - Worst-case 18 sec algorithm running time
- Meets the target TTD of 15 min
  - Can be made faster but mitigation time is on human time scale

Outline

- Global View Approach
- Model & Algorithm
- System
- Evaluation
- Architectural Lessons
- Related Work

ToR as a Single Point of Failure

- **Reduced Network Cost vs. Availability cost for using a single ToR per rack**
- Soft failures (recoverable by reboot) vs. hard failures

ToR Availability

$$= 1 - \frac{(\%\ \mathbf{soft} * \mathbf{soft\ dur.} + \%\ \mathbf{hard} * \mathbf{hard\ dur.}) * \mathbf{frac.\ rebooted\ ToRs\ per\ month}}{\mathbf{total\ time\ in\ a\ month}}$$

$$= 1 - \frac{(\mathbf{90\%} * \mathbf{20\ min} + \mathbf{10\%} * \mathbf{120\ min}) * \mathbf{0.1\%}}{\mathbf{30} * \mathbf{24} * \mathbf{60\ min}}$$

$$= \mathbf{99.99993}\%$$

- Dependent services (ToRs) need to provide one extra nine to target service (VMs)

ToRs not on critical path for VMs to achieve five-nines availability

VMs and their Storage Co-location

- For load balancing, VMs can mount VHDs from any storage cluster in the same region
- Some VMs have storage that are further away
- **Can longer network paths impact VM availability? And by how much?**
- At Azure, 52% two-hop, 41% four-hop
- Compute daily VHD failure rates: r_0 (two-hop), r_1 (four-hop)
- Average over 3-months, $\overline{\mathrm{r}_0}$ and $\overline{\mathrm{r}_1}$
- $(\overline{\mathrm{r}_1} - \overline{\mathrm{r}_0})/\overline{\mathrm{r}_0} = 11.4\%$ increase

Longer network path do lead to higher (11.4%) VHD failure rate

Related Work

- **NetPoirot [SIGCOMM '16]**
  - A single-node solution to failure localization using TCP statistics
  - Complementary if TCP statistics from customer VMs are available
- **Binary Tomography**
  - Deepview achieves higher precision/recall than those greedy heuristics
- **(Approximate) Bayesian Network**
  - Too slow for our problem
  - Future work to compare accuracy experimentally

Conclusion

- Identified VHD failures as the availability bottleneck at Azure
- Deepview reduced unclassified daily VHD failures from 10,000s to 100s
- Revealed new failures, e.g., unplanned ToR reboots, storage gray failures
- Quantified the impact of several architectural decisions on VM availability

Thank you! Questions?