Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure

Qiao Zhang¹, Guo Yu², Chuanxiong Guo³,
Yingnong Dang⁴, Nick Swanson⁴, Xinsheng Yang⁴, Randolph Yao⁴,
Murali Chintalapati⁴, Arvind Krishnamurthy¹, Tom Anderson¹

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

- 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

Subsystems inside a Datacenter
A New Type of Failure: VHD Failures

- 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?

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

VHD failure localization is the bottleneck

Breakdown of Unplanned VM Downtime in a Year

- SW Failure 41%
- HW Failure 6%
- Unknown 1%
- VHD Failure 52%
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
2. Need to handle gray failures
3. Need to be near-real-time

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

\[ \text{Prob(path i is healthy)} = \prod_{j \in \text{path}(i)} p_j \]

\[ \text{Prob(component j is healthy)} \]

Assume independent failures

\[ e_i = \text{num of VMs crashed} \]
\[ n_i = \text{num of VMs} \]

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

System of Linear Equations

\[ y_i = \sum_{j=1}^{N} \beta_j x_{ij} + \epsilon_i \]
\[ \beta_j = \log p_j \]
\[ \epsilon_i = \text{measurement noise} \]

Blue: observable
Red: unknown
Purple: topology

Component j is healthy with
\[ p_j = \exp(\beta_j) \]

- \( \beta_j = 0 \), clear component j
- \( \beta_j \ll 0 \), may blame it
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 \( \beta_j \) to be zero
- **Lasso regression** can get sparse solutions efficiently

Lasso Objective Function:
\[
\hat{\beta} = \text{argmin}_{\beta \in \mathbb{R}^N, \beta \leq 0} ||y - X\beta||^2 + \lambda \|\beta\|_1
\]

Example:
\[
\begin{align*}
\text{Net} \\
C1 & \quad C2 & \quad S1 & \quad S2 \\
\hat{y}_1 &= \beta_{c1} + \beta_{\text{net}} + \beta_{s1} + \varepsilon_1 \\
\hat{y}_2 &= \beta_{c1} + \beta_{\text{net}} + \beta_{s2} + \varepsilon_2 \\
\hat{y}_3 &= \beta_{c2} + \beta_{\text{net}} + \beta_{s1} + \varepsilon_3 \\
\hat{y}_4 &= \beta_{c2} + \beta_{\text{net}} + \beta_{s2} + \varepsilon_4
\end{align*}
\]
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: $H_0(j): \beta_j = \bar{\beta}$ vs. $H_A(j): \beta_j < \bar{\beta}$
• If reject $H_0(j)$, blame component $j$; otherwise, clear it
Deepview System Architecture: NRT Data Pipeline

- **Ingestion Pipeline**
  - Real-time: VHD Failure
  - Non-RT: VM Info, StorageAcct, Net Topo

- **Ingestion Pipeline**
  - RAW DATA

- **SLIDING WINDOW OF INPUT**
  - VMsPerPath

- **RUN ALGO**
  - Algo

- **ACTIONS**
  - Alerts, Vis

- **Real-time**
  - Kusto Engine

- **Non-RT**
  - VM Info, StorageAcct, Net Topo

- **Input**

- **Output**
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
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
• 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

\[= 1 - \frac{(% \text{soft} \times \text{soft dur.} + % \text{hard} \times \text{hard dur.}) \times \text{frac. rebooted ToRs per month}}{\text{total time in a month}}\]

\[= 1 - \frac{(90\% \times 20 \text{ min} + 10\% \times 120 \text{ min}) \times 0.1\%}{30 \times 24 \times 60 \text{ min}}\]

\[= 99.99993\%\]

• 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, $\overline{r_0}$ and $\overline{r_1}$
- $\frac{(\overline{r_1} - \overline{r_0})}{\overline{r_0}} = 11.4\%$ 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?