# Deepview: Virtual Disk Failure Diagnosis and Pattern Detection for Azure

By Qiao Zhang, Guo Yu, Chuanxiong Guo, Yingnong Dang, Nick Swanson, Xinsheng Yang, Randolph Yao, Murali Chintalapati, Arvind Krishnamurthy, and Thomas Anderson

Presented by: Andrew Yoo

# Compute-Storage Separation

- Virtual hard disks (VHDs) and VMs on different physical clusters
- Easy VM migration when VMs are unavailable
  - Can create VM on different host/cluster
  - Simply attach VHD
- Load balancing through many-to-many relationship
  - Many VMs can use the same VHD
  - One VM can use multiple VHDs
- VHD driver in hypervisor level send RPCs to storage service

# VHD Failure

- In Azure, shutdown guest OS if unresponsive for 2 minutes
  - Mainly to protect data integrity
  - Notify the customers of VHD access failure
  - Maintain customers' SLAs

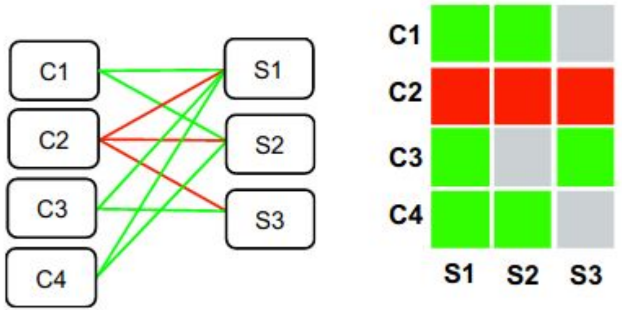
| VHD Failure | SW Failure | HW Failure | Unknown |
|---|---|---|---|
| 52% | 41% | 6% | 1% |

Table 1: Breakdown of the causes of VM downtime. VHD failures cause the majority of VM downtime.

# Previous Approach

- SREs examined different parts of the system
    - Compute team, storage team, and network team
- 10s of minutes or sometimes hours to localize the failure
- One network failure led to 363 incidents
    - Incident was moved from one team to another
- In short, not accurate and not efficient

# Bipartite Model




(a) Bipartite model (b) Matrix view

Figure 3: The bipartite model and the corresponding matrix view of a downtime event.

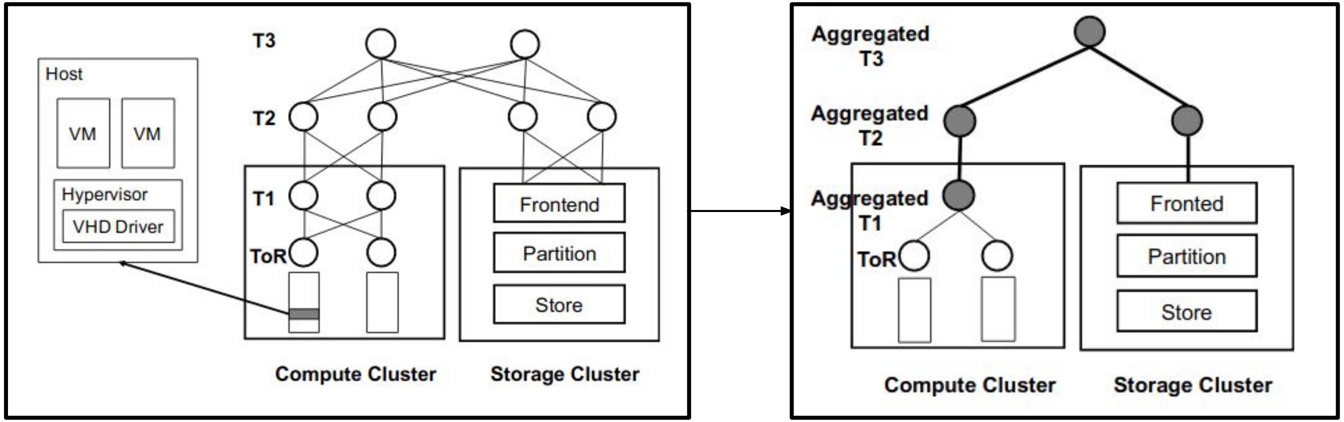
# Goals of the Paper

- Stronger granularity
  - Simple bipartite model does not work for multi-tier networks
- Detection of Gray failures
- Fast detection
  - Aim for 15 minutes (availability objective)

# Model - Network

- Represent Clos topology as tree
    - Start from Top-of-Rack switches
    - Group every Tier 1 switch that connects to ToR
    - Group every Tier 2 switch that connects to T1 switch
    - And so on....
    - Finally, determine midpoint by collecting shortest paths
- Discussion: Valid approach?

# Model - Network



# Derivation of Probabilities

$$\mathbb{P}(\text{path i is fine}) = \prod_{j \in \text{path}(i)} \mathbb{P}(\text{component j is fine})$$

# Derivation of Probabilities

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

$n_i$ is the number of VMs
$e_i$ is the number of VMs with VHD failures for a given period
$p_j$ is the probability that the path is fine

Next step: Model approximation with noise and create system of equations

# Derivation of Probabilities

$$y_i = \sum_{j=1}^{N} \beta_j x_{ij} + \varepsilon_i, \qquad \varepsilon_i \overset{i.i.d.}{\sim} N(0, \sigma^2)$$

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

# Interpretation of Equations

- A significantly negative $B_j$ usually suggests to blame that component
- Necessity to modify data
  - Failures are normally independent from each other
  - Array of *B* should theoretically be mostly 0's
  - Apply Lasso estimate with a parameter ƛ (black-box)
    - Tradeoff: goodness-of-fit versus sparsity

$$\hat{\beta} = \underset{\beta \in \mathbb{R}^N, \beta \leq \mathbf{0}}{\arg\min} \ ||\mathbf{y} - \mathbf{X}\beta||_2^2 + \lambda \, ||\beta||_1 \, .$$

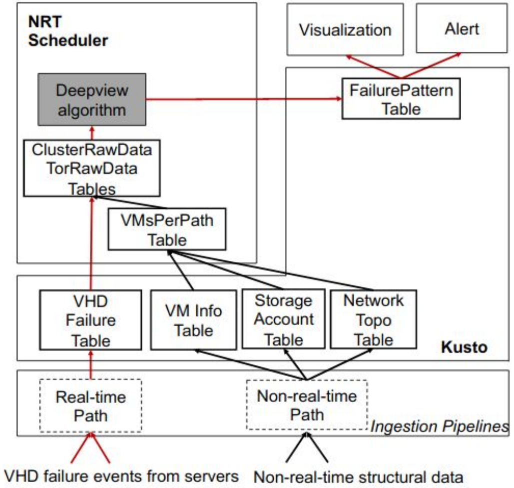
# Hypothesis Testing

1. Define a z-score from the mean and standard deviation of a value
2. Compute the p-value
   - In this paper, assume Gaussian distribution
   - Is this a safe assumption?
3. Make decision based on the p-value
   - If the p-value is less than 1%, blame the component

# Deepview System

- Non-real-time information
  - Network topology, account information, compute-storage
  - Periodic snapshots every few hours
    - Sufficient?
- Real-time information
  - VHD failures send signals
  - Implement a streaming system

# Deepview System




| Table Name | Schema |
|---|---|
| VHDFailure | (ts, vm_id, vhd, str_account) |
| VMInfo | (ts, vm_id, comp_cluster, tor) |
| StorageAccount | (ts, str_account, str_cluster) |
| NetworkTopo | (ts, cluster, tor_list, t1_list, t2_list, t3_list) |
| VMsPerPath | (tstart, tend, num_vms) |
| ClusterRawData | (tstart, tend, comp_cluster, str_cluster, num_vms, num_failed_vms) |
| TorRawData | (tstart, tend, comp_cluser, tor, str_cluster, num_vms, num_failed_vms) |
| FailurePattern | (tstart, tend, region, type, loc, pval, visual_url) |

# Deepview System

- Computation DAG
- NRT Scheduler
- Sparse Matrix/Region Filtering
  - Used to reduce runtime
- Coordinate Descent
  - One of the fastest ways to solve Lasso regression
- Cross-validation for $\lambda$

# Evaluation: ToR Reboot

- All VMs communicate with a single ToR
  - Failure means lack of communication with VHD
- Deepview predicts and verifies prediction
- Chose the correct ToR out of 288 components
  - p-value much less than 0.01

# Evaluation: Storage Gray Failure

- At hour 0, three components with failure probability > 0
  - Storage cluster S0 - 0.34
  - Compute cluster C0 - 0.002
  - Compute cluster C1 - 0.047
- Only S0 corresponded to p-value less than 1%
- Similar case for network failures

# Other Algorithms

- Boolean-Tomo and SCORE
  - Greedy algorithms
  - Goal: Identify bad paths based on threshold
  - Iterative discovery or computation
- Approximate Bayesian Network
  - Exponential runtime
  - Lack of meaningful results

# Precision and Recall

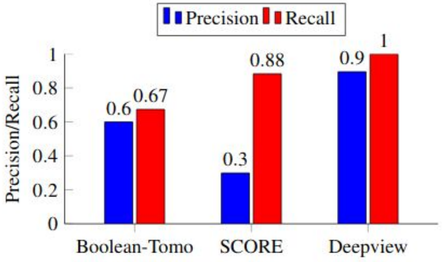



Figure 10: Precision/Recall comparison.

| | Compute | Storage | Net | ToR |
|---|---|---|---|---|
| Precision | 0.85 | 0.875 | 1.0 | 1.0 |
| Recall | 1.0 | 1.0 | 1.0 | 1.0 |

Table 3: Precision/Recall by failure type for Deepview.

# How well does it perform?

- Lasso regression
  - Impossible to find universally optimal Lambda
- Hypothesis testing
  - Even with low failure probability, hypothesis testing works
  - Example: Storage cluster gray failure case
- Runtime of Deepview
  - Worst-case: 18.3 seconds
- TTD: under 10 minutes

# ToR - Single-point-of-Failure

- Questions about ToR as bottleneck for availability
- Deepview detects ToR failure
  - More data!
- First, <0.1% of switches experience ToR reboots
- Second, 90% of reboots are soft

$$1 - \frac{0.9 \times 20 + 0.1 \times 120}{1000 \times 30 \times 24 \times 60} = 99.99993\%$$

# Network Path

- Distribution of network path
  - 51.4% go up to T2
  - 41.0% go up to T3
  - Rest go beyond
- Leverage Deepview data
- 11.4% increase in VHD failure rate with T3 or above
- Maybe beneficial to keep VHD and VM close

# Machine Learning

- Can a ML approach work in this situation?
- Possible weakness: Need richer signals
- Paper mentions NetPoirot
  - Features: TCP statistics
  - Labels: failure locations
  - Complementary
- BUT can a ML approach replace or outperform Deepview?

# Pros and Cons

Pros:

- Application of simple statistical ideas in a real-world system problem
- Presentation of valuable lessons learned through the data acquired in Deepview

Cons:

- Some parts of the model have to be inferred through reading different parts (could have been more clear)
- Lack of discussion on more examples of cases

# References

## Figures, equations, and content used from:

Qiao Zhang, Guo Yu, Chuanxiong Guo, Yingnong Dang, Nick Swanson, Xinsheng Yang, Randolph Yao, Murali Chintalapati, Arvind Krishnamurthy, and Thomas Anderson. 2018. Deepview: virtual disk failure diagnosis and pattern detection for azure. In *Proceedings of the 15th USENIX Conference on Networked Systems Design and Implementation* (*NSDI'18*). USENIX Association, USA, 519–532.