Check before You Change:
Preventing Correlated Failures in Service Updates
Background

• Cloud services ensure reliability by redundancy:
  - Storing data redundantly
  - Replicating service states across multiple nodes

• Examples:
  - Amazon AWS, AliCloud, Google Cloud, etc. replicate their data and service states
However, cloud outages still occur

Why redundancy does not help?
An AWS Outage in 2018

AWS outage killed some cloudy servers, recovery time is uncertain

‘Power event’ blamed, hit subset of kit in US-EAST-1

By Simon Sharwood 1 Jun 2018 at 00:48

Updated Parts of Amazon Web Services' US-East-1 region have experienced about half an hour of downtime, but some customers' instances and data can't be restored because the hardware running them appears to have experienced complete failure.

The cloud colossus' status page reports an investigation of “connectivity issues affecting some instances in a single Availability Zone in the US-EAST-1 Region" as of 3:13 PM PDT on Thursday, May 31.

A 3:42 PM update confirmed “an issue in one of the datacenters that makes up one of US-EAST-1 Availability Zones. This was a result of a power event impacting a small percentage of the physical servers in that datacenter as well as some of the networking devices.”
Elastic Compute Cloud (EC2)

Elastic Block Store (EBS)
Correlated failures resulting from deep dependencies
Correlated Failures

- Correlated failures are harmful and epidemic:
  - Propagated to all the redundant instances
  - Undermine redundancy and fault tolerance efforts
Correlated failures are prevalent
State of the Art

Service initialization

Service Runtime
State of the Art

Post-Failure Forensics

1. Diagnosis (e.g., Sherlock [SIGCOMM'07])
2. Accountability (e.g., AVM [OSDI'10])
3. Provenance (e.g., DiffProv [SIGCOMM'16])
4. ... ...
State of the Art

Proactive Auditing
1. INDaas [OSDI’14]
2. reCloud [CoNEXT’16]
3. RepAudit [OOPSLA’17]
4. … …

Post-Failure Forensics
1. Diagnosis (e.g., Sherlock [SIGCOMM’07])
2. Accountability (e.g., AVM [OSDI’10])
3. Provenance (e.g., DiffProv [SIGCOMM’16])
4. … …

Service initialization

Service Runtime
Proactive Auditing

• They did pre-deployment recommendations:
Proactive Auditing

- They did pre-deployment recommendations:
  - Step1: Automatically collecting dependency data
Proactive Auditing

• They did pre-deployment recommendations:
  - Step1: Automatically collecting dependency data
  - Step2: Modeling system stack in fault graph
Proactive Auditing

• They did pre-deployment recommendations:
  - Step1: Automatically collecting dependency data
  - Step2: Modeling system stack in fault graph
  - Step3: Evaluating alternative deployments’ independence
Redundancy configuration fails
Redundancy configuration fails

AND gate: all the sublayer nodes fail, the upper layer node fails
OR gate: one of the sublayer nodes fails, the upper layer node fails
Redundancy configuration fails

- Server 1 fails
  - HW fails
  - Net fails
  - SW fails
    - Path1
    - Path2
    - Agg1
    - Core1
    - Agg2

- Server 2 fails
  - Net fails
  - SW fails
  - HW fails
    - HBase
    - HDFS

... ...
... ...
... ...
Redundancy configuration fails

Server 1 fails
  - HW fails
  - Net fails
  - SW fails
    - Path1
      - Agg1
    - Path2
      - Agg2

Server 2 fails
  - HW fails
  - Net fails
  - SW fails
    - HBase
    - HDFS
State of the Art

Proactive Auditing
1. INDaas [OSDI’14]
2. reCloud [CoNEXT’16]
3. RepAudit [OOPSLA’17]
4. ...

Post-Failure Forensics
1. Diagnosis (e.g., Sherlock [SIGCOMM’07])
2. Accountability (e.g., AVM [OSDI’10])
3. Provenance (e.g., DiffProv [SIGCOMM’16])
4. ...

Service initialization

Service Runtime
Benjamin Treynor Sloss, Google's VP of engineering, explained that the root cause of last Sunday's outage was a configuration change for a small group of servers in one region being wrongly applied to a larger number of servers across several neighboring regions.
Problem 1: Inefficient Auditing in Updates

Proactive Auditing
1. INDaaS [OSDI’14]
2. reCloud [CoNEXT’16]
3. RepAudit [OOPSLA’17]
4. ... ...

O(50) hours per auditing V.S. One update every 3 hours

Service initialization
Changing network paths
Upgrading software components

Service Runtime

Post-Failure Forensics
1. Diagnosis (e.g., Sherlock [SIGCOMM’07])
2. Accountability (e.g., AVM [OSDI’10])
3. Provenance (e.g., DiffProv [SIGCOMM’16])
4. ... ...
Problem 2: Lack of fixing risks

Proactive Auditing

1. INDaas [OSDI’14]
2. reCloud [CoNEXT’16]
3. RepAudit [OOPSLA’17]
4. ...

Changing network paths

Fix ?

Upgrading software components

---

Service Runtime

Post-Failure Forensics

1. Diagnosis (e.g., Sherlock [SIGCOMM’07])
2. Accountability (e.g., AVM [OSDI’10])
3. Provenance (e.g., DiffProv [SIGCOMM’16])
4. ...
Our Contribution

Proactive Auditing
1. INDaas [OSDI’14]
2. reCloud [CoNEXT’16]
3. RepAudit [OOPSLA’17]
4. … …

Post-Failure Forensics
1. Diagnosis (e.g., Sherlock [SIGCOMM’07])
2. Accountability (e.g., AVM [OSDI’10])
3. Provenance (e.g., DiffProv [SIGCOMM’16])
4. … …

Service initialization
Changing network paths
Upgrading software components

Service Runtime

CloudCanary

Fast Audit & Fix
CloudCanary’s Workflow

Updated Service Snapshot

Operator
CloudCanary’s Workflow

Updated Service Snapshot

Dependency acquisition and Fault graph generator

Operator

CloudCanary’s Work
CloudCanary’s Workflow

Updated Service Snapshot → Dependency acquisition and Fault graph generator → Fault Graph

Operator
CloudCanary’s Workflow

- Updated Service Snapshot
- Dependency acquisition and Fault graph generator
- Fault Graph
- SnapAudit

1. {CoreRouter-1}
2. {Agg1, Agg2}
   ...

- Challenge 1: SnapAudit

- Operator

CloudCanary’s Work
CloudCanary’s Workflow

- Challenge 1: SnapAudit
- Challenge 2: DepBooster
CloudCanary’s Workflow

- Challenge 1: SnapAudit
- Challenge 2: DepBooster
A Fault Graph

Redundancy Deployment

E1   E1

A1  A2  A3
Risk Groups in Fault Graphs

- A risk group means a set of leaf nodes whose simultaneous failures lead to the failure of root node.
Risk Groups in Fault Graphs

- A risk group means a set of leaf nodes whose simultaneous failures lead to the failure of root node

{A2} and {A1, A3} are risk groups
{A1} or {A3} is not risk group
Risk Groups in Fault Graphs

Identifying correlated failure risks can be reduced to the problem of finding risk groups in the fault graph.

However, analyzing risk groups is \textbf{NP-complete problem}.

Simultaneous failures lead to the failure of root node:

\{A2\} and \{A1, A3\} are risk groups

\{A1\} or \{A3\} is not risk group
CloudCanary’s Workflow

- Challenge 1: SnapAudit
- Challenge 2: DepBooster
The Insight of SnapAudit

$S \xrightarrow{\Delta \text{ is small}} S' \xrightarrow{\Delta' \text{ is small}} S'' \xrightarrow{\Delta'' \text{ is small}} \ldots$

- Service initialization
- Changing network paths
- Upgrading software components

Service Runtime
The Insight of SnapAudit

FirstAudit

$S \rightarrow \Delta \text{ is small} \rightarrow S' \rightarrow \Delta' \text{ is small} \rightarrow S'' \rightarrow \Delta'' \text{ is small}$

Service initialization

Changing network paths

Upgrading software components

Service Runtime
The Insight of SnapAudit

FirstAudit

- Service initialization
- Changing network paths
- Upgrading software components

CloudCanary

IncAudit

- $\Delta$ is small
- $\Delta'$ is small
- $\Delta''$ is small
SnapAudit: FirstAudit & IncAudit

FirstAudit

IncAudit

IncAudit

IncAudit

Δ is small

Δ′ is small

Δ″ is small

Service initialization

Changing network paths

Upgrading software components

Service Runtime
FirstAudit Primitive

```
R
 +
  |
  +
  |
F
 +
 |
D
 +
 |
A
 +
 |
B
 +
 |
B
 +
 |
C
 +
 |
X
 +
 |
Y
 +
 |
...
...
```
FirstAudit Primitive

**H(D)=43cd**

**H(E)=a4vo**

**H(F)=x31g**

**H(R)=aed8**

**H(Z)=lktd**

**H(X)=xbn7**

**H(Y)=bbk9**
FirstAudit Primitive

H(D)=43cd

H(F)=x31g

H(E)=a4vo

H(R)=aed8

H(Z)=lktd

H(X)=xbn7

H(Y)=bbk9
FirstAudit Primitive

- \( H(D) = 43cd \)
  - \{A\}
  - \{B\}

- \( H(F) = x31g \)

- \( H(E) = a4vo \)
  - \{B\}
  - \{C\}

- \( H(X) = xbn7 \)

- \( H(Y) = bbk9 \)

- \( H(R) = aed8 \)
FirstAudit Primitive

- \( H(D) = 43cd \)
  - \{A\}
  - \{B\}

- \( H(F) = x31g \)
  - \{A, B\}
  - \{A, C\}
  - \{B, B\}
  - \{B, C\}

- \( H(E) = a4vo \)
  - \{B\}
  - \{C\}

- \( H(R) = aed8 \)
- \( H(Z) = lktd \)
- \( H(X) = xbn7 \)
- \( H(Y) = bbk9 \)
**FirstAudit Primitive**

- **H(D) = 43cd**
  - \{A\}
  - \{B\}

- **H(F) = x31g**
  - \{A, B\}
  - \{A, C\}
  - \{B, B\} = \{B\}
  - \{B, C\}

- **H(E) = a4vo**
  - \{B\}
  - \{C\}

- **H(X) = xbn7**

- **H(Y) = bbk9**

- **H(Z) = lktd**

- **H(R) = aed8**
FirstAudit Primitive

- \( H(F) = \text{31g} \)
  - \{A, B\}
  - \{A, C\}
  - \{B\}
  - \{B, C\}

- \( H(D) = \text{43cd} \)
  - \{A\}
  - \{B\}

- \( H(E) = \text{a4vo} \)
  - \{B\}
  - \{C\}

- \( H(X) = \text{xbn7} \)
  - \{A\}
  - \{S, T\}

- \( H(Y) = \text{bbk9} \)
  - \{K\}
  - \{A, S\}

- \( H(Z) = \text{lktd} \)
  - \{A\}
  - \{K\}
  - \{S, T\}

- \( H(R) = \text{aed8} \)

\( H(R) - \{A\} - \{K\} - \{S, T\} \)

\( H(Z) - \{A\} - \{K\} - \{S, T\} \)

\( H(X) - \{A\} - \{S, T\} \)

\( H(Y) - \{K\} - \{A, S\} \)

\( H(F) - \{A, B\} - \{A, C\} - \{B\} - \{B, C\} \)

\( H(D) - \{A\} - \{B\} \)
FirstAudit Primitive

H(D) = 43cd
- {A}
- {B}

H(E) = a4vo
- {B}
- {C}

H(F) = x31g
- {A, B}
- {A, C}
- {B}
- {B, C}

H(R) = aed8

H(X) = xbn7
- {A}
- {S, T}

H(Y) = bbk9
- {K}
- {A, S}

H(Z) = lktd
- {A}
- {K}
- {S, T}

H(R) = aed8
FirstAudit Primitive

<table>
<thead>
<tr>
<th>Node</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>H(D)=43cd</td>
</tr>
<tr>
<td>F</td>
<td>H(F)=x31g</td>
</tr>
<tr>
<td>E</td>
<td>H(E)=a4vo</td>
</tr>
<tr>
<td>X</td>
<td>H(X)=xbn7</td>
</tr>
<tr>
<td>R</td>
<td>H(R)=aed8</td>
</tr>
<tr>
<td>Z</td>
<td>H(Z)=lktd</td>
</tr>
<tr>
<td>Y</td>
<td>H(Y)=bbk9</td>
</tr>
</tbody>
</table>

- {A}
- {B}
- {A, C}
- {B}
- {B}
- {B}
- {A}
- {S, T}
- {K}
- {A, S}
- {A}
- {K}
- {S, T}
FirstAudit Primitive

- \( H(D) = 43cd \)
  - \{A\}
  - \{B\}

- \( H(E) = a4vo \)
  - \{B\}
  - \{C\}

- \( H(F) = x31g \)
  - \{B\}
  - \{A, C\}

- \( H(R) = aed8 \)

- \( H(X) = xbn7 \)

- \( H(Z) = lktd \)

- \( H(Y) = bbk9 \)
FirstAudit Primitive

H(D) = 43cd
- {A}
- {B}

H(E) = a4vo
- {B}
- {C}

H(F) = x31g
- {B}
- {A, C}

H(R) = aed8
- {A}
- {K}
- {B}
- {S, T}

H(Z) = lktd
- {A}
- {K}
- {S, T}

H(Y) = bbk9
- {K}
- {A, S}

H(X) = xbn7
- {A}
- {S, T}
SnapAudit: FirstAudit & IncAudit

FirstAudit

IncAudit IncAudit IncAudit

Service initialization Changing network paths Upgrading software components

Δ is small Δ′ is small Δ″ is small
Our Insight

• Algorithm sketch:
  - Finding all the border nodes (black nodes)
  - Computing their risk groups
  - Merging these risk groups towards root
Updated Deployment

H(D) = 43cd
- {A}
- {B}

H(E) = a4vo
- {B}
- {C}

H(F) = x31g
- {B}
- {A, C}

H(X) = xbn7
- {A}
- {S, T}

H(Y) = bbk9
- {K}
- {A, S}

H(Z) = lktd
- {A}
- {K}
- {S, T}

H(R) = aed8
- {A}
- {K}
- {B}
- {S, T}
Updated Deployment

H(D) = 43cd
- {A}
- {B}

H(F) = x31g
- {B}
- {A, C}

H(E) = a4vo
- {B}
- {C}

H(R) = aed8
- {A}
- {K}
- {B}
- {S, T}

H(X) = xbn7
- {A}
- {S, T}

H(Y) = bbk9
- {K}
- {A, S}

H(Z) = 2xzb
- {A}
- {K}
- {S, T}
Updated Deployment

H(D)=43cd
- {A}
- {B}

H(F)=x31g
- {B}
- {A, C}

H(E)=a4vo
- {B}
- {C}

H(R)=45zc
- {A}
- {K}
- {B}
- {S, T}

H(Z)=2xzb
- {A}
- {K}
- {S, T}

H(X)=xbn7
- {A}
- {S, T}

H(Y)=bbk9
- {K}
- {A, S}

H(R)=45zc
- {A}
- {K}
- {B}
- {S, T}
Step 1: Find Border Nodes

<table>
<thead>
<tr>
<th>Node</th>
<th>Label</th>
<th>Adjacent Sets</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>H(D)=43cd</td>
<td>{A}, {B}, {A,C}</td>
</tr>
<tr>
<td>E</td>
<td>H(E)=a4vo</td>
<td>{B}, {C}</td>
</tr>
<tr>
<td>F</td>
<td>H(F)=x31g</td>
<td>{B}, {A,C}</td>
</tr>
<tr>
<td>R</td>
<td>H(R)=45zc</td>
<td>{A}, {K}, {B}, {S,T}</td>
</tr>
<tr>
<td>Z</td>
<td>H(Z)=2xzb</td>
<td>{A}, {K}, {S,T}</td>
</tr>
<tr>
<td>X</td>
<td>H(X)=xbn7</td>
<td>{A}, {S,T}</td>
</tr>
<tr>
<td>Q</td>
<td>H(Y)=bbk9</td>
<td>{K}, {A,S}</td>
</tr>
</tbody>
</table>
Step 2: Q's Risk Groups

Border Nodes

H(R)=45zc
- {A}
- {K}
- {B}
- {S, T}

H(Z)=2xz
- {A}
- {K}
- {S, T}

H(F)=x31g
- {B}
- {A, C}

H(D)=43cd
- {A}
- {B}

H(E)=a4vo
- {B}
- {C}

H(X)=xbn7
- {A}
- {S, T}

H(Y)=bbk9
- {K}
- {A, S}

Step 2: Q's Risk Groups
Boolean formula

= E₁ ∧ E₂

= (A₁ ∨ A₂) ∧ (A₂ ∨ A₃)
Our Insight

Boolean formula

\[ = E_1 \land E_2 \]

\[ = (A_1 \lor A_2) \land (A_2 \lor A_3) \]

Satisfying assignment:

\{A_1=1, A_2=0, A_3=1\}

SAT solver
Our Insight

- Problem:
  - Standard SAT solver outputs an arbitrary satisfying assignment
Our Insight

- Problem:
  - Standard SAT solver outputs an arbitrary satisfying assignment
  - What we want is top-k critical (minimal) risk groups
Identifying Risk Groups

- Using MinCostSAT solver
  - Satisfiable assignment with the least weights
  - Obtain the least $C = \sum c_i \cdot w_i$
  - Very fast with 100% accuracy
Identifying Risk Groups

• Using MinCostSAT solver
  - Satisfiable assignment with the least weights
  - Obtain the least \( C = \sum c_i \cdot w_i \)
  - Very fast with 100% accuracy

We set the values of all the leaf nodes (i.e., \( W_i \)) as 1
Identifying Risk Groups

• Using MinCostSAT solver
  - Satisfiable assignment with the least weights
  - Obtain the least $C = \sum c_i \cdot w_i$
  - Very fast with 100% accuracy

<table>
<thead>
<tr>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Identifying Risk Groups

- Using MinCostSAT solver
  - Satisfiable assignment with the least weights
  - Obtain the least $C = \sum c_i \cdot w_i$
  - Very fast with 100% accuracy

<table>
<thead>
<tr>
<th></th>
<th>A1</th>
<th>A2</th>
<th>A3</th>
<th>Weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
</tr>
</tbody>
</table>
Identifying Risk Groups

- Find out the top-k critical risk groups
  - Use $\land$ to connect the current formula and the negation of the resulting assignment

$$(A_1 \lor A_2) \land (A_2 \lor A_3) \land \neg(\neg A_1 \land A_2 \land \neg A_3)$$
Step 2: Q’s Risk Groups

H(F) = x31g
- {B}
- {A, C}

H(D) = 43cd
- {A}
- {B}

H(E) = a4vo
- {B}
- {C}

H(X) = xbn7
- {A}
- {S, T}

H(Y) = bbk9
- {K}
- {A, S}

H(Z) = 2xzb
- {A}
- {K}
- {S, T}

H(R) = 45zc
- {A}
- {K}
- {B}
- {S, T}
Step 2: Q’s Risk Groups

- **H(D)=43cd**
  - {A}
  - {B}

- **H(E)=a4vo**
  - {B}
  - {C}

- **H(F)=x31g**
  - {B}
  - {A, C}

- **H(R)=45zc**
  - {A}
  - {K}
  - {B}
  - {S, T}

- **H(Z)=2xzb**
  - {A}
  - {K}
  - {S, T}

- **H(X)=xbn7**
  - {A}
  - {S, T}

- **H(Q)=x1r7**
  - {A}
  - {S, T}

- **H(Y)=bbk9**
  - {K}
  - {A, S}
Step 3: Merging Changed Caches
Step 3: Merging Changed Caches

- \( H(D) = 43cd \)
  - \{A\}
  - \{B\}

- \( H(E) = a4vo \)
  - \{B\}
  - \{C\}

- \( H(F) = x31g \)
  - \{B\}
  - \{A, C\}

- \( H(R) = 45zc \)
  - \{A\}
  - \{K\}
  - \{B\}
  - \{S, T\}

- \( H(X) = xbn7 \)
  - \{A\}
  - \{S, T\}

- \( H(Y) = bbk9 \)
  - \{K\}
  - \{A, S\}

- \( H(Z) = 2xzb \)
  - \{A\}
  - \{K\}
  - \{S\}
Step 3: Merging Changed Caches

- **R**
  - \( H(R)=45zc \)
  - {A}
  - {K}
  - {B}
  - {S, T}

- **F**
  - \( H(F)=x31g \)
  - {B}
  - {A, C}

- **D**
  - \( H(D)=43cd \)
  - {A}
  - {B}

- **E**
  - \( H(E)=a4vo \)
  - {B}
  - {C}

- **X**
  - \( H(X)=xbn7 \)
  - {A}
  - {S, T}

- **Q**
  - \( H(Q)=x1r7 \)
  - {S}
  - {K}

- **Y**
  - \( H(Y)=bbk9 \)
  - {K}
  - {A, S}

- **Z**
  - \( H(Z)=2xzb \)
  - {A}
  - {K}
  - {S}

- **A**
- **B**
- **C**
Step 3: Merging Changed Caches

- H(F) = x31g
  - {B}
  - {A, C}

- H(D) = 43cd
  - {A}
  - {B}

- H(E) = a4vo
  - {B}
  - {C}

- H(X) = xbn7
  - {A}
  - {S, T}

- H(Y) = bbk9
  - {K}
  - {A, S}

- H(Z) = 2xzb
  - {A}
  - {K}
  - {S}

- H(R) = 45zc
  - {A}
  - {K}
  - {B}
  - {S}
CloudCanary’s Workflows

• Challenge 1: SnapAudit
• Challenge 2: DepBooster
Correlated Failure Risk Repairing
Correlated Failure Risk Repairing

Specification:
$Server \rightarrow 172.28.228.21, 172.28.228.22$
$\text{goal}(\text{failProb}(ft)<0.08 \mid \text{ChNode} \mid \text{Agg3})$
Correlated Failure Risk Repairing

$Server \rightarrow 172.28.228.21, 172.28.228.22$

goal(failProb(ft)<0.08 \mid ChNode \mid Agg3)$
Correlated Failure Risk Repairing

Specification:
$$\text{Server} \rightarrow 172.28.228.21, 172.28.228.22$$
$$\text{goal} (\text{failProb}(ft) < 0.08 \mid \text{ChNode} \mid \text{Agg3})$$

Plan 1: Move replica from S1 \rightarrow S4
Plan 2: Move replica from S2 \rightarrow S4
Correlated Failure Risk Repairing

**Specification:**

\[
\text{Server} \rightarrow 172.28.228.21, 172.28.228.22 \\
goal(\text{failProb}(ft)<0.08 \mid \text{ChNode} \mid \text{Agg3})
\]

**Synthesis**

Plan 1: Move replica from S1 \rightarrow S4  
Plan 2: Move replica from S2 \rightarrow S4
Correlated Failure Risk Repairing

Specification:
$Server \rightarrow 172.28.228.21, 172.28.228.22$
goal(failProb(ft)<0.08 | ChNode | Agg3)

Plan 1: Move replica from S1 -> S4
Plan 2: Move replica from S2 -> S4
CloudCanary’s Workflow

- Challenge 1: SnapAudit
- Challenge 2: DepBooster
Evaluation

• Comparing CloudCanary with the state of the art

• Evaluating CloudCanary’s practicality via real dataset
## Evaluation

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Efficiency</th>
<th>Improvement</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>INDaaS [OSDI’14]</strong></td>
<td>✔️</td>
<td>✘</td>
<td>✘</td>
</tr>
<tr>
<td><strong>ProbINDaaS [OSDI’14]</strong></td>
<td>✘</td>
<td>✔️</td>
<td>✘</td>
</tr>
<tr>
<td><strong>reCloud [CoNEXT’16]</strong></td>
<td>✘</td>
<td>✔️</td>
<td>✘</td>
</tr>
<tr>
<td><strong>RepAudit [OOPSLA’17]</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✘</td>
</tr>
<tr>
<td><strong>CloudCanary</strong></td>
<td>✔️</td>
<td>✔️</td>
<td>✔️</td>
</tr>
</tbody>
</table>
Efficiency Comparison

- INDaaS
- ProbINDaaS \((10^7)\)
- ReCloud \((10^7)\)
- RepAudit
- CloudCanary

Auditing Time (hours)

- INDaaS: \(~8\) hours
- ProbINDaaS: \(>16\) hours
- ReCloud: \(>16\) hours
- RepAudit: \(>16\) hours
- CloudCanary: \(~8\) hours

Update Snapshots
Accuracy V.S. Efficiency

- 20,608 switches; 524,288 servers; 638,592 software components
- Auditing a random update affecting 20% components
Our approach is 200x faster than state-of-the-arts, and offers 100% accurate results.
We evaluated CloudCanary via real update trace:

<table>
<thead>
<tr>
<th></th>
<th>Detected Num</th>
<th>Confirmed</th>
<th>Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Microservices</strong></td>
<td>50+</td>
<td>96%</td>
<td>Authentication and access control systems introduce most risk groups</td>
</tr>
<tr>
<td><strong>Power Sources</strong></td>
<td>10+</td>
<td>100%</td>
<td>Primary and backup power sources are carelessly assigned to multiple racks hosting a critical service</td>
</tr>
<tr>
<td><strong>Network</strong></td>
<td>30+</td>
<td>100%</td>
<td>Aggregation and ToR switches are easily updated to be risk groups</td>
</tr>
</tbody>
</table>
Conclusion

• CloudCanary is the first system for real-time auditing
  - SnapAudit primitive: Quickly auditing update snapshot
  - DepBooster: Quickly generating improvement plans

• We evaluated CloudCanary with real trace and large-scale emulations
Thanks, questions?

• CloudCanary is the first system for real-time auditing
  - SnapAudit primitive: Quickly auditing update snapshot
  - DepBooster: Quickly generating improvement plans

• We evaluated CloudCanary with real trace and large-scale emulations