Rapid and Robust Impact Assessment of Software Changes in Large Internet-based Services

Shenglin Zhang, Ying Liu, Dan Pei
Yu Chen, Xianping Qu, Shimin Tao, Zhi Zang
Internet-based Services

- Search
  - Google
  - bing
  - Baidu

- Shopping
  - Amazon
  - eBay
  - Taobao

- Social
  - Facebook
  - Twitter
  - Weibo

- Portal
  - Yahoo!
  - AOL
  - Sina

- Video
  - YouTube
  - Blip
  - Youku
Software Change: **Software Upgrade or Configuration Change**

- **Software upgrade**

  - Introduce new feature
  - Improve performance
  - Fix bugs
Software Change: Software Upgrade or Configuration Change

• Software upgrade

  Introduce new feature
  Improve performance
  Fix bugs

• Configuration change
  • e.g., traffic switching for load balancing reasons
Software Change: **Software Upgrade or Configuration Change**

- **Software upgrade**
  - Introduce new feature
  - Improve performance
  - Fix bugs

- **Configuration change**
  - e.g., traffic switching for load balancing reasons

- **Occurs frequently**
  - 10K+ per day in Baidu
Impact of Erroneous Software Upgrades

2012.10, Google

An update to Google's load balancing software
• Poor performance to Gmail for 18 minutes

Google Apps Incident Report
Gmail Partial Outage - December 10, 2012
Prepared for Google Apps customers

The following is the full incident report for the Gmail partial outage on December 10, 2012. We understand this event caused inconvenience, and apologize to all those affected.

Issue:
For 18 minutes, some Gmail users experienced poor performance.

The notification from users indicated issues with Gmail and the Chrome Sync.
The root cause was a software issue.

Actions and Root Cause:
Background: The load balancing software is responsible for routing users' requests to Google data centers around the world for processing and serving content, such as search results and email.

Between 8:45 AM PT and 9:13 AM PT, a routine update to Google's load balancing software was rolled out to production. A bug in the software update caused it to incorrectly interpret a portion of Google data centers as being unavailable. The Google load balancers have a fail-safe mechanism to prevent this type of failure from causing Google-wide service degradation, and they continued to route user traffic. As a result, most Google services, such as Google Search, Maps, and AdWords, were unaffected. However, some services, including Gmail, that require specific data center information to efficiently route users' requests, experienced a partial outage.
Impact of Erroneous Software Upgrades

2012.10, Google

Google Apps Incident Report
Gmail Partial Outage - December 10, 2012
Prepared for Google Apps customers

The following is the updated incident report from December 10, 2012. We understand this update may cause issues, and we apologize to our customers.

Issue:
For the vast majority of users, this experience did not occur. However, a subset of users did experience decreased performance.

The resolution:
The root cause of this event was an update to Google's load balancing software.

Actions and Root Cause:
Background: The load balancing software processes requests from users’ requests to Google data centers around the world for processing and serves content, such as search results and email.

Between 8:45 AM PT and 9:13 AM PT, a routine update to Google's load balancing software was rolled out to production. A bug in the software update caused it to incorrectly interpret a portion of Google data centers as being unavailable. The Google load balancers have a fail-safe mechanism to prevent this type of failure from causing Google-wide service degradation, and they continued to route user traffic. As a result, most Google services, such as Google Search, Maps, and AdWords, were unaffected. However, some services, including Gmail, that require specific data center information to efficiently route users' requests, experienced a partial outage.

2014.11, Microsoft Azure

Update on Azure Storage Service Interruption

WEDNESDAY, NOVEMBER 19, 2014

JASON ZANDER
CVP, Microsoft Azure Team

A performance update to Azure Storage
• Reduced capacity across services utilizing Azure Storage

As part of a performance update to Azure Storage, an issue was discovered that resulted in reduced capacity across services utilizing Azure Storage, including Virtual Machines, Visual Studio Online, Websites, Search and other Microsoft services. Prior to applying the performance update, it had been tested over several weeks in a subset of our customer-facing storage service for Azure Tables. We typically call this “flighting,” as we work to identify issues before we broadly deploy any updates. The flighting test demonstrated a notable performance improvement and we proceeded to deploy the update across the storage service. During the rollout we discovered an issue that resulted in storage blob front ends going into an infinite loop, which had gone undetected during flighting. The root result was an inability for the front ends to take on further traffic, which in turn caused other services built on top to experience issues.
Impact of Erroneous Configuration Changes

2014.1, Dropbox

Outage post-mortem

- Planned maintenance to upgrade the OS on some machines
- Dropbox service been down for three hours

We use thousands of nodes in the Dropbox database, which has one master and two replica machines for redundancy. We keep our data on these nodes and create incremental data backups and store them in a separate environment.

On Friday at 5:30 PM PT, we had a planned maintenance scheduled to upgrade the OS on some of our machines. During this process, the upgrade script checks to make sure there is no active data on the machine before installing the new OS.

A subtle bug in the script caused the command to reinstall a small number of active machines. Unfortunately, some master-replica pairs were impacted which resulted in the site going down.

6/22/18
Impact of Erroneous Configuration Changes

2014.1, Dropbox

Outage post-mortem

- Planned maintenance to upgrade the OS on some machines
- Dropbox service been down for three hours

2014.6, Facebook

Facebook outage caused by software system update

- Update the configuration of the software systems
- Failed Facebook for 31 minutes
Impact of Erroneous Software Changes

• Poor user experience
Impact of Erroneous Software Changes

• Poor user experience

• A drop in revenue

A real-world example

The normalized number of successful orders
Manual Software Change Impact Assessment

Select a subset of KPIs that maybe impacted
Manual Software Change Impact Assessment

Select a subset of KPIs that may be impacted

Inspect KPI changes

![Graph showing performance measurements with positive and negative level shifts and ramp downs.](image-url)
Manual Software Change Impact Assessment

Select a subset of KPIs that maybe impacted

Inspect KPI changes

Decide whether to roll back

6/22/18

CoNEXT 2015
KPI (Key Performance Indicator) in Software Change

• KPIs of servers
  • CPU utilization
  • Memory utilization
  • NIC throughput
  • ...

6/22/18 CoNEXT 2015
KPI (Key Performance Indicator) in Software Change

• KPIs of servers
  • CPU utilization
  • Memory utilization
  • NIC throughput
  • ...

• KPIs of modules/processes
  • Web page view count
  • Web page view delay
  • ...

6/22/18

CoNEXT 2015
KPI (Key Performance Indicator) in Software Change

- KPIs of servers
  - CPU utilization
  - Memory utilization
  - NIC throughput
  - ...
- KPIs of modules/processes
  - Web page view count
  - Web page view delay
  - ...
- Up to hundreds of KPIs for a single software change
Definition of KPI Change: **Level Shift** or **Ramp up/down**

- KPI change
  - Indicative of performance increase/degradation
  - Hard to simulate in testbeds
  - Not reproducible
Manual Software Change Impact Assessment

Select a subset of KPIs that maybe impacted

Inspect KPI changes

Decide whether to roll back

• Labor-intensive
• Prone to error
• Not scalable
Design Goal

Software Change Impact Assessment System

- Automatic
- Scalable
- Robust to various software changes and KPIs

Decide whether to roll back
Outline

- Background and Motivation
- Challenges
- Key Ideas
- Results
- Conclusion
Challenge 1: Short Detection Delay Requirement Against Robustness

- Poor user experience
- A drop in revenue

A real-world example
Challenge 1: Short Detection Delay Requirement Against Robustness

• Poor user experience

• A drop in revenue

A real-world example
Challenge 1: Short Detection Delay Requirement Against Robustness

- Poor user experience
- A drop in revenue

Detect KPI changes rapidly and accurately

The number of successful orders (normalized)

A real-world example

Software upgrade
Challenge 2: Large Number of KPIs
Challenge 2: Large Number of KPIs

100+ Internet-based services
20+ Internet-based services has 100+ million users
10k+ modules
500+ thousand servers
Challenge 2: Large Number of KPIs

Monitored by one operations team
Challenge 2: Large Number of KPIs

Monitored by one operations team

10k+ software changes per day
Challenge 2: Large Number of KPIs

- Monitored by one operations team
- 10k+ software changes per day
- 100+ KPIs in a software change
Challenge 2: Large Number of KPIs

- Millions of KPIs should be monitored
- Monitored by one operations team
- 10k+ software changes per day
- 100+ KPIs in a software change
- Millions of KPIs should be monitored
Challenge 2: Large Number of KPIs

- **Millions of KPIs to be monitored**
- **10k+ software changes per day**

Detect KPI changes with low computational cost
Challenge 3: Diverse Types of Data

• Diverse types of KPI data

Seasonal
Page view count

Variable
NIC throughput

Stationary
Memory utilization
Challenge 3: Diverse Types of Data

- Diverse types of KPI data

<table>
<thead>
<tr>
<th>Seasonal</th>
<th>Variable</th>
<th>Stationary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page view count</td>
<td>NIC throughput</td>
<td>Memory utilization</td>
</tr>
</tbody>
</table>

Robust to various KPIs
Challenge 4: KPI Changes Maybe Caused by Other Factors

Seasonality

Network breakdowns

Malicious attacks
Challenge 4: KPI Changes Maybe Caused by Other Factors

Seasonality  Network breakdowns  Malicious attacks

Eliminate KPI changes induced by other factors
Outline

• Background and Motivation
• Challenges
• Key Ideas
• Results
• Conclusion
Step 1 - Identify the impact set

Software change in module A
Design Overview

Step 1 - Identify the impact set

KPIs in the impact set

Software change in module A
Identify the Impact Set: Automatically Retrieve the Relevant KPIs
Identify the Impact Set: Automatically Retrieve the Relevant KPIs

Input from operators

- Modules related module A: module B, C, D
- Servers/processes where the software change is deployed.
Design Overview

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Design Overview

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs

KPIs in the impact set

KPIs with behavior changes

Software change in module A
**Design Overview**

**Step 1** - Identify the impact set
- KPIs in the impact set

**Step 2** - Detect behavior changes in KPIs
- KPIs with behavior changes

- Short detection delay requirement against robustness
- Diverse types of data
- Large number of KPIs

Software change in module A
Improved Singular Spectrum Transform (SST)

- Improved singular spectrum transform (SST) $x_s(t) = 1 - \alpha(t)^T \beta(t)$

Advantage

Accurate

Short detection delay

Short detection delay requirement against robustness
Improved Singular Spectrum Transform (SST)

• Improved singular spectrum transform (SST)  
  \[ x_s(t) = 1 - \alpha(t)^T \beta(t) \]

Advantage

- Accurate
- Short detection delay

Drawbacks

- Accuracy degrades with noisy baseline
- High computational cost

T. Idé and K. Tsuda, SDM 2007
Improved Singular Spectrum Transform (SST)

• Improved singular spectrum transform (SST)

Advantage
- Accurate
- Short detection delay

Drawbacks
- Accuracy degrades with noisy baseline
- High computational cost

Improve robustness
- Utilize more information in the testing space

Diverse types of data
Improved Singular Spectrum Transform (SST)

- Improved singular spectrum transform (SST)
  \[ \varphi_i(t) \approx 1 - \sum_{j=1}^{\eta} x_{ij}^2 \]

**Advantage**
- Accurate
- Short detection delay

**Drawbacks**
- Accuracy degrades with noisy baseline
- High computational cost

**Improve robustness**
- Utilize more information in the testing space

**Reduce computational cost**
- Large number of KPIs
- Matrix compression
- Implicit inner product calculation

6/22/18
CoNEXT 2015
Design Overview

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Step 3 - Eliminate KPI changes induced by other factors

KPIs in the impact set

KPIs with behavior changes

Software change in module A
Design Overview

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Step 3 - Eliminate KPI changes induced by other factors

KPIs in the impact set

KPIs with behavior changes

KPIs with behavior changes induced by software change

Software change in module A
Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Step 3 - Eliminate KPI changes induced by other factors
Eliminate KPI Changes Induced by Other Factors
Eliminate KPI Changes Induced by Other Factors

• Split testing
  • Evaluation of interventions instituted at a specific time
  • Control group & treated group
Eliminate KPI Changes Induced by Other Factors

- **Split testing**
  - Evaluation of interventions instituted at a specific time
  - Control group & treated group

![Graph showing KPI changes due to software change](image-url)
Eliminate KPI Changes Induced by Other Factors

**Treated group**

- Servers/processes in the impact set
Eliminate KPI Changes Induced by Other Factors

Treated group

• Servers/processes in the impact set

Control group

• Servers/processes in the same module
• Without software change
Eliminate KPI Changes Induced by Other Factors

Treated group
- Servers/processes in the impact set

Control group
- Servers/processes in the same module
- Without software change

DiD method
Eliminate KPI Changes Induced by Other Factors

Treated group
- Servers/processes in the same module
- Without software change

Control group
- Servers/processes in the impact set

DiD method

KPI changes maybe caused by other factors

treated group
control group
Design Overview

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Step 3 - Eliminate KPI changes induced by other factors

KPIs in the impact set

KPIs with behavior changes

KPIs with behavior changes induced by software change

Software change in module A

improved SST

split testing
Design Overview

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Step 3 - Eliminate KPI changes induced by other factors

KPIs in the impact set

KPIs with behavior changes

KPIs with behavior changes induced by software change

Software change in module A

Operators
KPIs with behavior changes induced by software change

Step 1 - Identify the impact set
Step 2 - Detect behavior changes in KPIs
Step 3 - Eliminate KPI changes induced by other factors

Design Overview
Design Overview

- Step 1 - Identify the impact set
- Step 2 - Detect behavior changes in KPIs
- Step 3 - Eliminate KPI changes induced by other factors

KPIs in the impact set

KPIs with behavior changes

KPIs with behavior changes induced by software change

Software change in module A

Operators

CoNEXT 2015
Outline

• Background and Motivation
• Challenges
• Key Ideas
• Results
• Conclusion
Datasets of Evaluation

144 software changes of Baidu

72 introduced KPI changes
72 introduced no KPI changes
Datasets of Evaluation

144 software changes of Baidu

72 introduced KPI changes
72 introduced no KPI changes

Large amount of labelling work

9982 (software change, server/module/process, KPI)s
Manually labelled by operators
Datasets of Evaluation

144 software changes of Baidu

72 introduced KPI changes
72 introduced no KPI changes

Large amount of labelling work

9982 (software change, server/module/process, KPI)s
Manually labelled by operators

Diverse KPIs

Seasonal
Variable
Stationary
Datasets of Evaluation

- 144 software changes of Baidu
- Large amount of labelling work
- Diverse KPIs
- Comparison baseline

72 introduced KPI changes
72 introduced no KPI changes
9982 (software change, server/module/process, KPI)s
Manually labelled by operators
Seasonal
Variable
Stationary
CUSUM (SIGCOMM 10)
Multiscale Robust Local Subspace (CoNEXT 11)
Comparison of Accuracy

Stationary

- FUNNEL
- Improved SST
- CUSUM
- MRLS
Comparison of Accuracy

Stationary

Seasonal

FUNNEL
Improved SST
CUSUM
MRLS
Comparison of Accuracy

- Stationary
- Seasonal
- Variable

- FUnnel
- Improved SST
- CUSUM
- MRLS
Comparison of Computational Cost

• Real-world scenario
  • At least 1 million KPIs need to be monitored
  • The detection interval for each KPI is 1 minute
  • Runs on the same kinds of CPU as testing
Comparison of Computational Cost

• Real-world scenario
  • At least 1 million KPIs need to be monitored
  • Each KPI is detected every 1 minute
  • Runs on the same kinds of CPU as testing

• Comparison results

<table>
<thead>
<tr>
<th>Method</th>
<th>FUNNEL</th>
<th>CUSUM</th>
<th>MRLS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of cores for one million KPIs</td>
<td>7</td>
<td>31</td>
<td>47526</td>
</tr>
</tbody>
</table>
Comparison of Detection Delay

- Detection delay
  - time when a KPI change is detected - time when a KPI change starts

![Graph showing time when the change starts and time when the change is detected](image_url)
Comparison of Detection Delay

• Comparison results
Comparison of Detection Delay

- Comparison results

![Comparison of Detection Delay Graph](image)

- CCDF: Complementary Cumulative Distribution Function
- Detection Delay (minute)
- FUNNEL
- CUSUM
- MRLS
Case Study: An Erroneous Software Upgrade in Advertising

• Methodology
  • A fraction of software changes
  • Not deliver the results to the operators
  • The operators assessed the software changes independently
Case Study: An Erroneous Software Upgrade in Advertising

• Methodology
  • A fraction of software changes
  • Not deliver the results to the operators
  • The operators assess software changes independently

• FUNNEL
  • 10 minutes
  • Seasonal KPIs
Case Study: An Erroneous Software Upgrade in Advertising

• Methodology
  • A fraction of software changes
  • Not deliver the results to the operators
  • The operators assess software changes independently

• FUNNEL
  • 10 minutes
  • Seasonal KPIs

• The operators
  • 1.5 hours
Outline

• Background and Motivation
• Challenges
• Key Ideas
• Results
• Conclusion
Conclusion

Challenges of automatic software change impact assessment

• Short detection delay requirement against robustness
• Large number of KPIs
• Diverse types of data
• KPI changes maybe caused by other factors

FUNNEL

• Improved SST – main algorithm contribution of the paper.
• Split testing

Evaluation

• Real-world software changes
Thank you!

zhangsl12@mails.tsinghua.edu.cn
Q&A
Why 144 Software Changes

• Evaluation needs ground truth
  • FUNNEL
    • detect KPI changes
    • determine whether KPIs changes are induced by software change
  • Operators
    • Label whether there is behavior change in KPI
    • Label whether a KPI changes is caused by software change
  • 9982 (software change, server/module/process, KPI)s
  • A huge amount of work
  • Labelling for much more software changes is prohibitive
Why Using Cores

• The CPU utilization is 100% in testing
• Assume the CPU utilization is also 100% in deployment
• The operators care about how many servers/cores the system needs
Why just a single team

- For the efficiency purpose
- Build a single database to monitor all KPIs
- By natural
Unbalanced hotspot

• Split testing
• The number of hotspots is very small (3% in Microsoft)
• Compare the treated group and the control group
• The large number of KPIs in the control group makes the determination robust even in the face of hotspots.
The parameters of FUNNEL, CUSUM and MRLS

- Two parameters
  - $\alpha$ in DiD method
  - $\omega$ in Improved SST
- Best for accuracy
- Operators care most about the accuracy
- Fair for the four methods
About the detection delay comparison

- Set a threshold for FUNNEL
- MRLS can detect behavior changes with smaller detection delay than FUNNEL at sometimes
- Sacrificing the accuracy
Why not Just Split Testing?

- Set threshold small
  - Sensitive to spikes
  - Many false positives

- Set threshold large
  - The detection delay is large

- Almost impossible to find a balance in our scenario

- The improved SST
  - Robust
  - Short detection delay
Obtain the Relationship of Modules

• The operators name the modules based on the module hierarchy
• The operators know the relationship of modules
Why not decide to roll out/back by FUNNEL?

• The KPI changes & the decision
  • Hard to learn
  • Few cases for a specific combination of KPI change and software change

• Rolling back a software change is a big thing
  • The operators would like to decide themselves.

• FUNNEL is helpful for the operators to make decision
  • The number of KPIs with behavior changes induced by software changes is small
  • The work of the operators is small.
About the Deployment

• Assess the software changes of a few dozens of Internet-based services

<table>
<thead>
<tr>
<th>Number of software changes</th>
<th>Number of changes that have impact</th>
<th>Number of KPIs</th>
<th>Number of KPI changes</th>
<th>Precision</th>
</tr>
</thead>
<tbody>
<tr>
<td>24119</td>
<td>268</td>
<td>2256390</td>
<td>10249</td>
<td>98.21%</td>
</tr>
</tbody>
</table>
If a Software Change is Deployed to All Servers ...

- **Treated group**
  - Measurements of KPIs in the impact set around the software change
- **Control group**
  - Measurements of KPIs in the impact set in the same period but on historical days
About the Number of Software Changes

- If a software change is deployed on a subset of servers firstly, and then on another subset of servers
- From the operators’ perspective
- They are two software changes