

Why do big data and cloud systems slow down and stop?

Shan Lu



What are?

Why do **big data and cloud systems**
slow down and stop?

Big data & cloud systems



PostgreSQL



Amazon Aurora



APACHE
HBASE



Apache
Zookeeper



cassandra



Big data & cloud systems

- DB-backed web applications
- Cloud services



PostgreSQL



Amazon Aurora



APACHE
HBASE



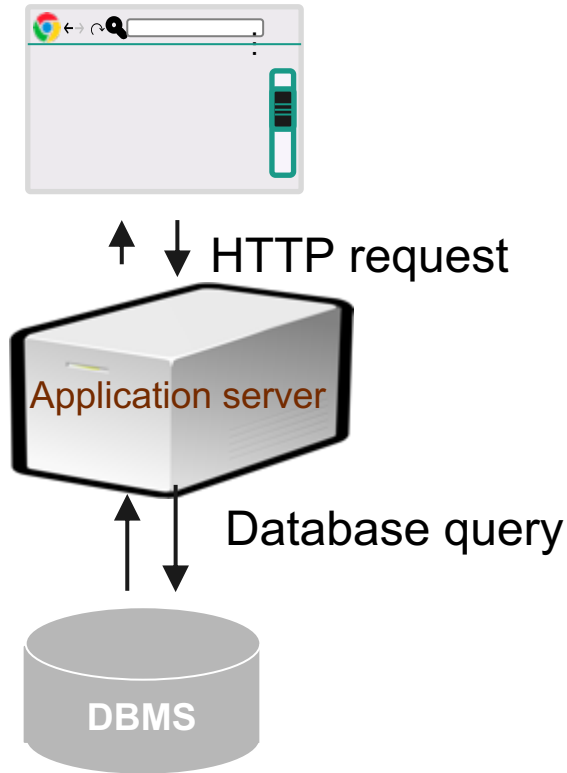
Apache
Zookeeper



cassandra



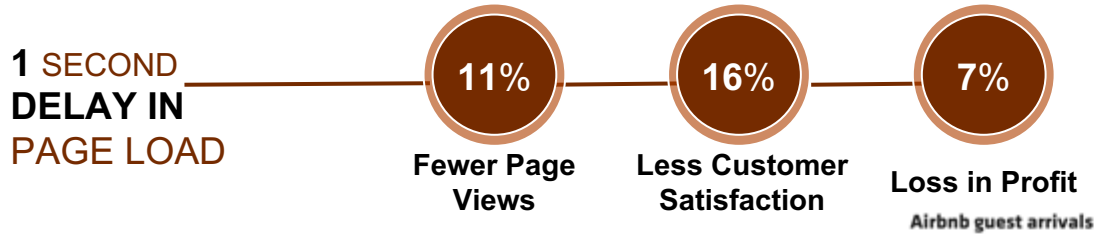
DB-backed web applications



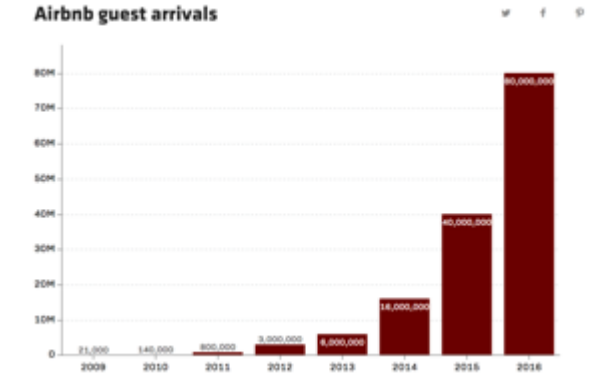
Performance is critical for web applications

- Low latency is critical

Nearly **half** of the users expect a site to load in less than **2** seconds



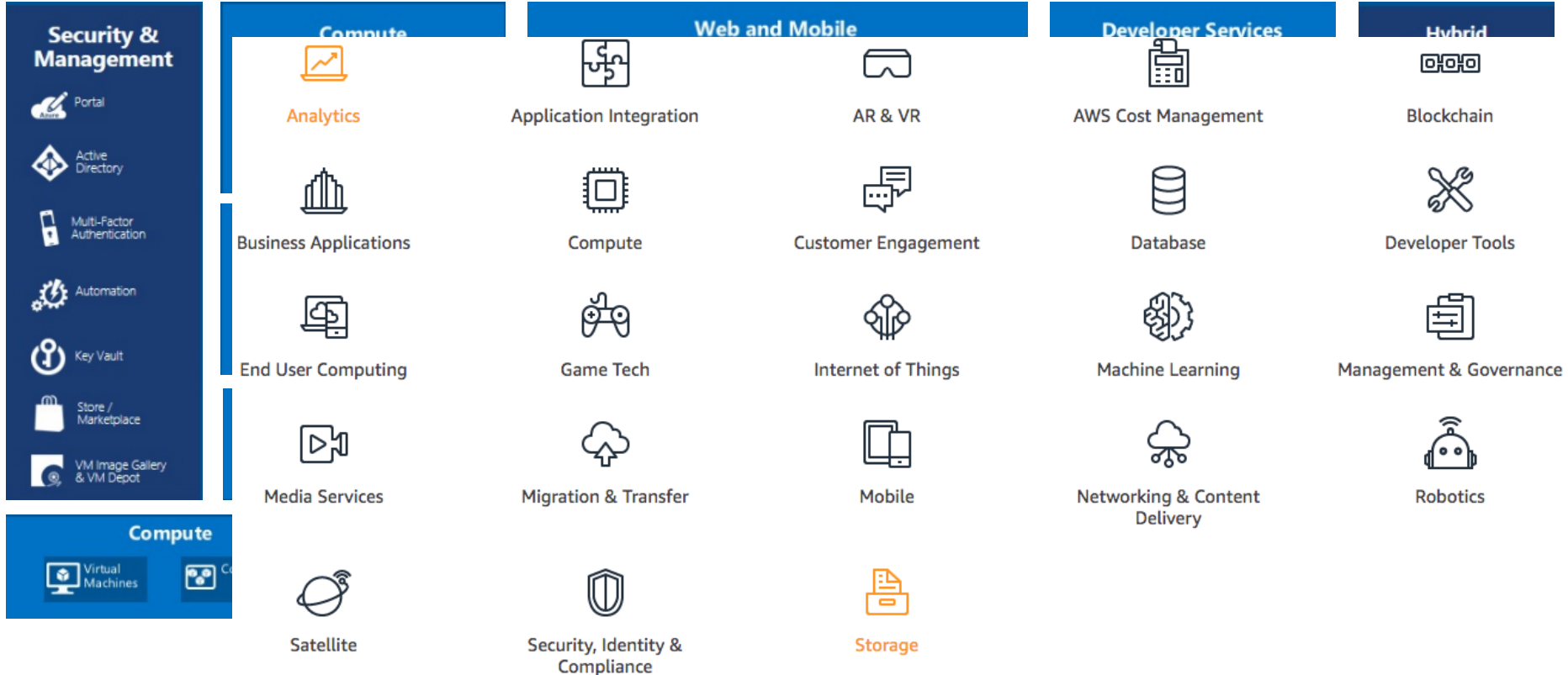
- Low latency is challenging given the data size



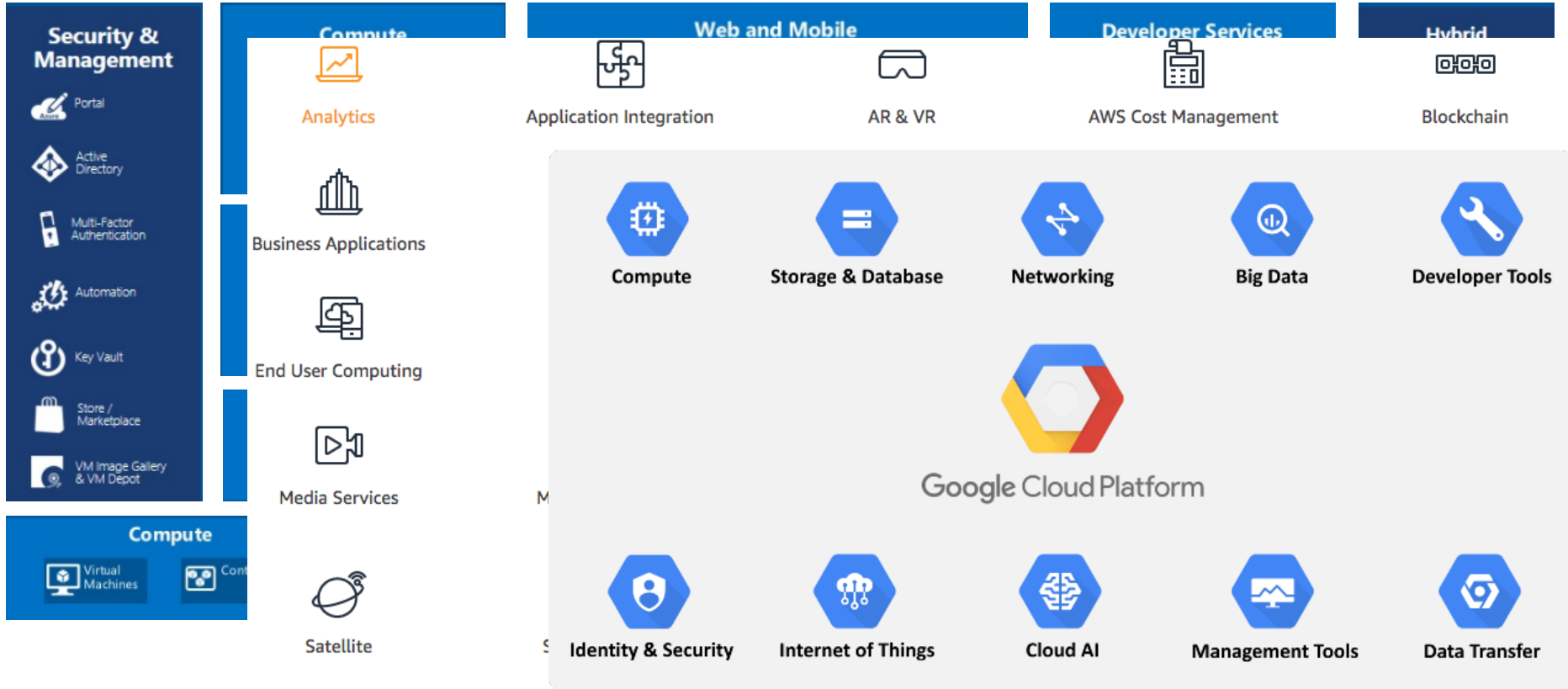
Cloud services

<h3>Security & Management</h3> <ul style="list-style-type: none"> Portal Active Directory Multi-Factor Authentication Automation Key Vault Store / Marketplace VM Image Gallery & VM Depot 	<h3>Compute</h3> <ul style="list-style-type: none"> Cloud Services Service Fabric Batch Remote App 	<h3>Web and Mobile</h3> <ul style="list-style-type: none"> Web Apps API Apps API Management Mobile Apps Logic Apps Notification Hubs 	<h3>Developer Services</h3> <ul style="list-style-type: none"> Visual Studio Azure SDK Team Project Application Insights 	<h3>Hybrid Operations</h3> <ul style="list-style-type: none"> Azure AD Connect Health AD Privileged Identity Management Backup Operational Insights Import/Export Site Recovery StorSimple 		
<h3>Integration</h3> <ul style="list-style-type: none"> Storage Queues Biztalk Services Hybrid Connections Service Bus 	<h3>Analytics & IoT</h3> <ul style="list-style-type: none"> HDInsight Machine Learning Data Factory Event Hubs Stream Analytics Mobile Engagement 	<h3>Data</h3> <ul style="list-style-type: none"> SQL Database SQL Data Warehouse Redis Cache Search DocumentDB Tables 				
<h3>Media & CDN</h3> <ul style="list-style-type: none"> Media Services Content Delivery Network (CDN) 	<table border="1"> <tr> <td data-bbox="59 882 455 1005"> <h3>Compute</h3> <ul style="list-style-type: none"> Virtual Machines Containers </td> <td data-bbox="463 882 927 1005"> <h3>Storage</h3> <ul style="list-style-type: none"> BLOB Storage Azure Files Premium Storage </td> <td data-bbox="935 882 1910 1005"> <h3>Networking</h3> <ul style="list-style-type: none"> Virtual Network Load Balancer DNS Express Route Traffic Manager VPN Gateway Application Gateway </td> </tr> </table>			<h3>Compute</h3> <ul style="list-style-type: none"> Virtual Machines Containers 	<h3>Storage</h3> <ul style="list-style-type: none"> BLOB Storage Azure Files Premium Storage 	<h3>Networking</h3> <ul style="list-style-type: none"> Virtual Network Load Balancer DNS Express Route Traffic Manager VPN Gateway Application Gateway
<h3>Compute</h3> <ul style="list-style-type: none"> Virtual Machines Containers 	<h3>Storage</h3> <ul style="list-style-type: none"> BLOB Storage Azure Files Premium Storage 	<h3>Networking</h3> <ul style="list-style-type: none"> Virtual Network Load Balancer DNS Express Route Traffic Manager VPN Gateway Application Gateway 				

Cloud services in Azure, AWS



Cloud services in Azure, AWS, GCP



Reliability is critical for cloud services

The image displays a grid of cloud service categories. A central focus is a red key on a keyboard labeled 'Reliability'. The grid is organized as follows:

- Security & Management** (Left sidebar): Portal, Active Directory, Multi-Factor Authentication, Automation, Key Vault, Store / Marketplace, VM Image Gallery & VM Depot.
- Compute** (Top row, left): Virtual Machines.
- Web and Mobile** (Top row, middle): Analytics, Application Integration, AR & VR, Customer Engagement, Internet of Things, Mobile.
- Developer Services** (Top row, right): AWS Cost Management, Database, Machine Learning, Networking & Content Delivery.
- Hybrid** (Top row, far right): Blockchain, Developer Tools, Management & Governance, Robotics.
- Bottom row**: Satellite, Security, Identity & Compliance, Storage.

Relia Businesses Losing \$700 Billion a Year to IT Downtime, Says IHS

Security & Management

- Portal
- Active Directory
- Multi-Factor Authentication
- Automation
- Key Vault
- Store / Marketplace
- VM Image Gallery & VM Depot

Network Interruptions Are the Biggest Culprit

Monday, January 25, 2016 9:01 am EST



Data Centre ▶ **Cloud**

AWS's S3 outage was so bad Amazon couldn't get into its own dashboard to warn the world

Websites, apps, security cams, IoT gear knackered

By [Shaun Nichols](#) in [San Francisco](#) 1 Mar 2017 at 03:00

122

SHARE ▼

ontent

Compute

- Virtual Machines
- Containers

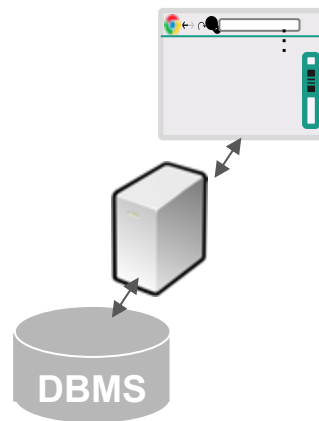


Hybrid

- Blockchain
- Developer Tools
- Management & Governance
- Robotics

Outline

- What slows down (big data) web applications [ICSE'18]
 - What can we do about it? [CIKM'17, FSE'18, ICSE'19, CIDR'20]
1000+ bugs found
- What stops cloud systems? [HotOS'19]
 - What can we do about it? [ASPLOS'16, ASPLOS'17, ASPLOS'18, PLDI'19, SOSP'19]
1000+ bugs found





What Slowed Down Database-Backed Web Applications




hyperloop.cs.uchicago.edu

Shan Lu



UCHICAGO

W UNIVERSITY of
WASHINGTON

View-Centric Performance Optimization for Database-Backed Web Applications. *ICSE'19*  
How not to structure your database-backed web applications: a study of performance bugs in the wild. *ICSE'18*.
PowerStation: Automatically detecting and fixing inefficiencies of database-backed web applications in IDE. *FSE'18* 

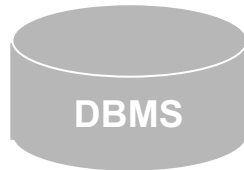
Common Web-app Architecture



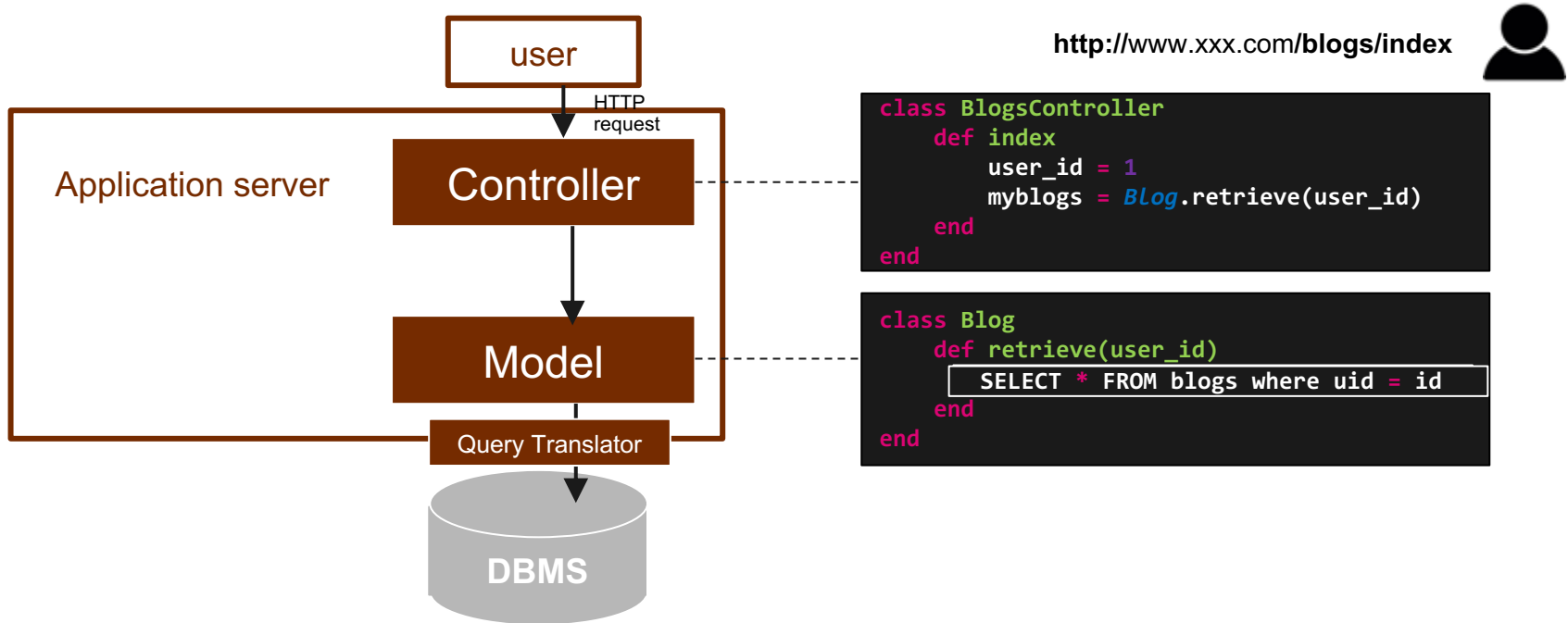
↑ ↓ HTTP request



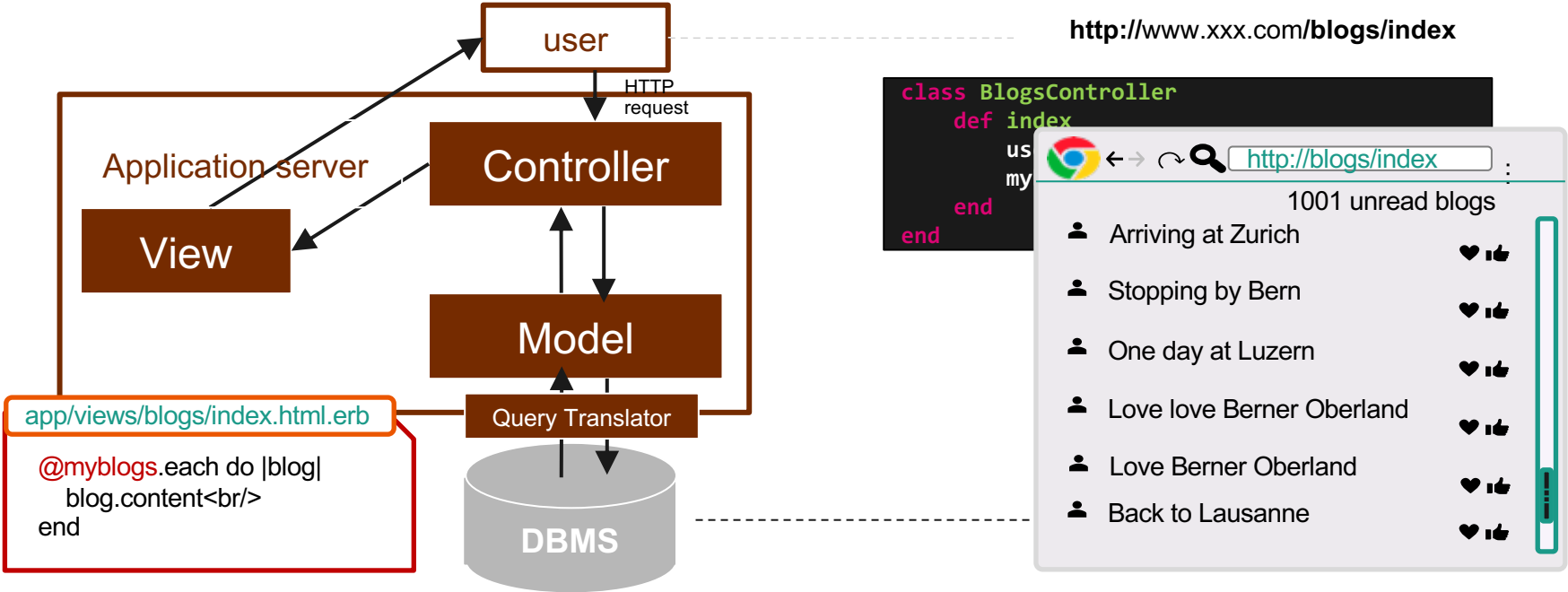
↑ ↓ Database query



Common Web-app Architecture

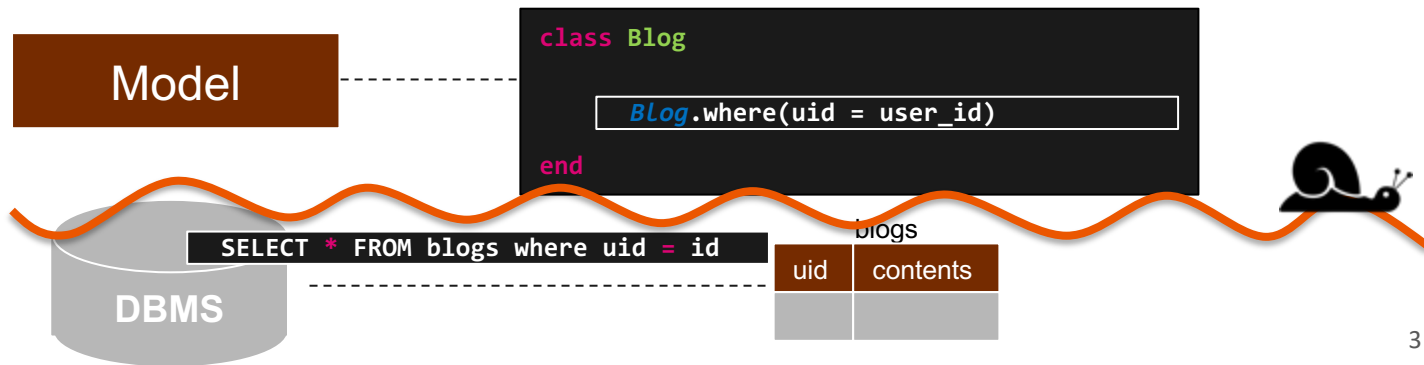


Common Web-app Architecture

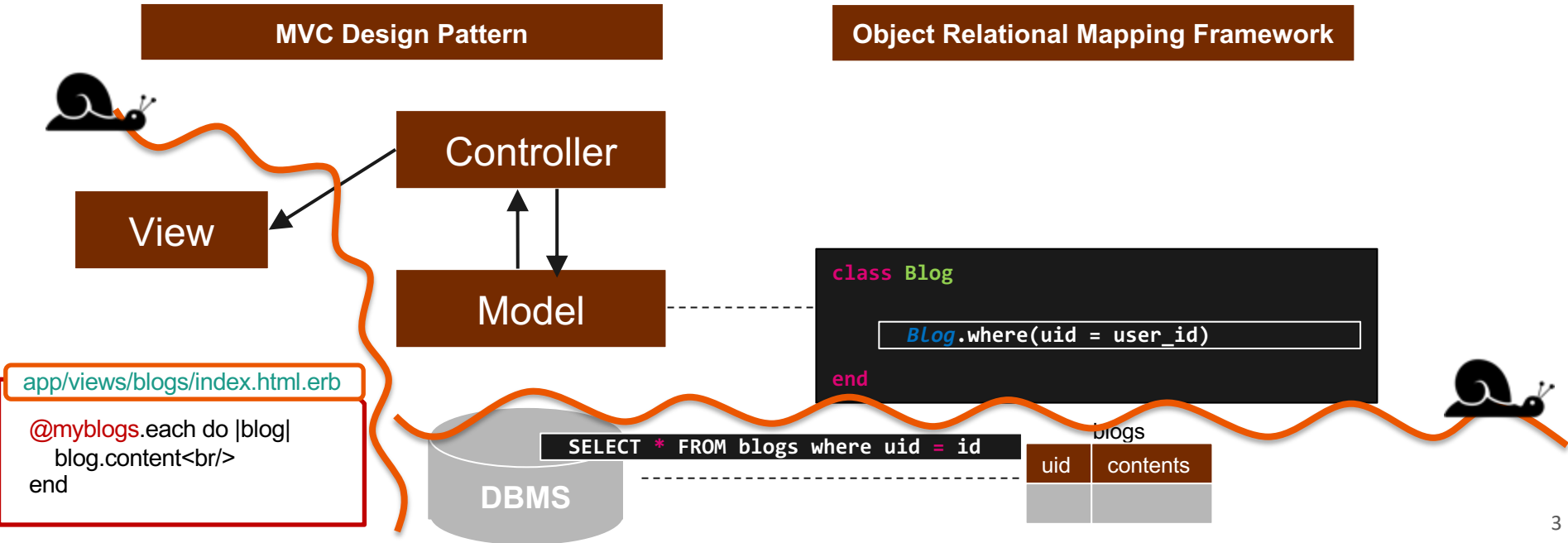


Potential sources of inefficiencies

Object Relational Mapping Framework



Potential sources of inefficiencies



Outline



How severe is the problem?

Profile 12 apps from 6 common categories

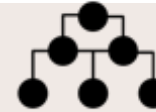


64 issues in
40 pages



What are the common inefficiency patterns?

Build performance-bug taxonomy



9 anti-
patterns



How to solve the problem?

Design automated bug detection & fixing



1000 +
bugs

Outline



Profile 12 apps from 6 common categories



64 issues in
40 pages



Build performance-bug taxonomy



Design automated bug detection & fixing

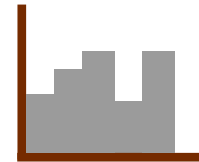
Profiling methodology



Table 1: Details of the applications chosen in our study

Category	Abbr.	Name	Stars	Commits	Contributors
Forum	Ds	Discourse	21238	22501	568
	Lo	Lobster	1304	1000	48
Collaboration	Gi	Gitlab	19255	49810	1276
	Re	Redmine	2399	13238	6
E-commerce	Sp	Spree	8331	17197	709
	Ro	Ror_ecommerce	1109	1727	21
Task-management	Fu	Fulcrum	1502	697	44
	Tr	Tracks	835	3512	62
Social Network	Da	Diaspora	11183	18734	335
	On	Onebody	1592	1220	6
Map	OS	Openstreetmap	664	8000	112
	FF	Fallingfruit	41	1106	7

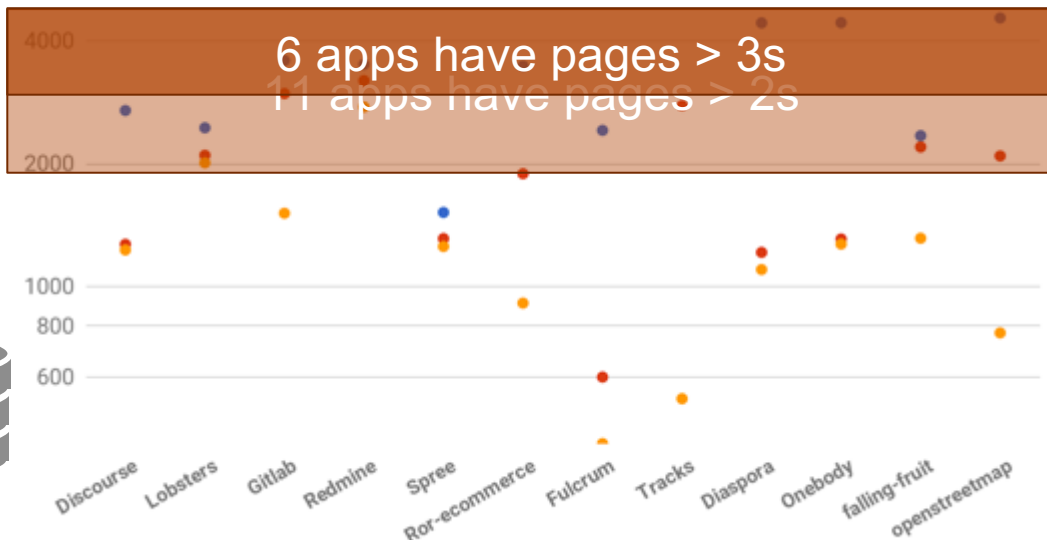
Synthesize DB content based on real-world website statistics



Top 2 Apps in 6 popular categories



Profiling End-to-end Page Time



40 problematic pages
Server takes most time



20000 record



Why is it slow?



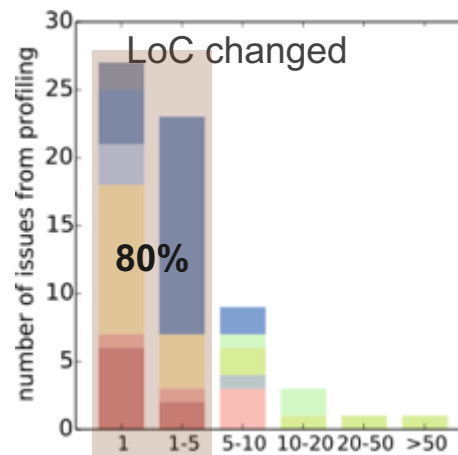
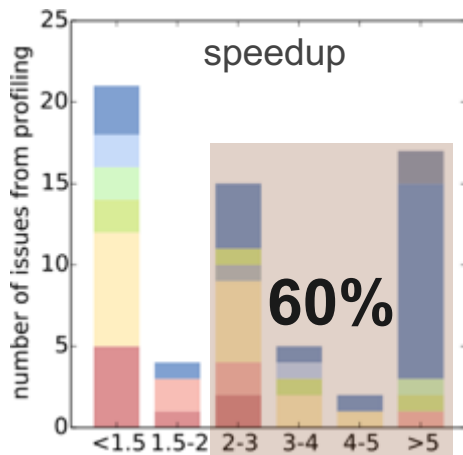
There are inefficiency bugs!



Why is it slow?



- We manually fix the 64 issues we found across 39 pages



There are bugs!



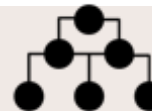
Outline



Profile 12 apps from 6 common categories



Build performance-bug taxonomy



9 anti-patterns

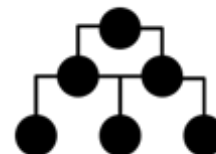


Design automated bug detection & fixing





Common Performance Anti-patterns



64 performance issues
from profiling

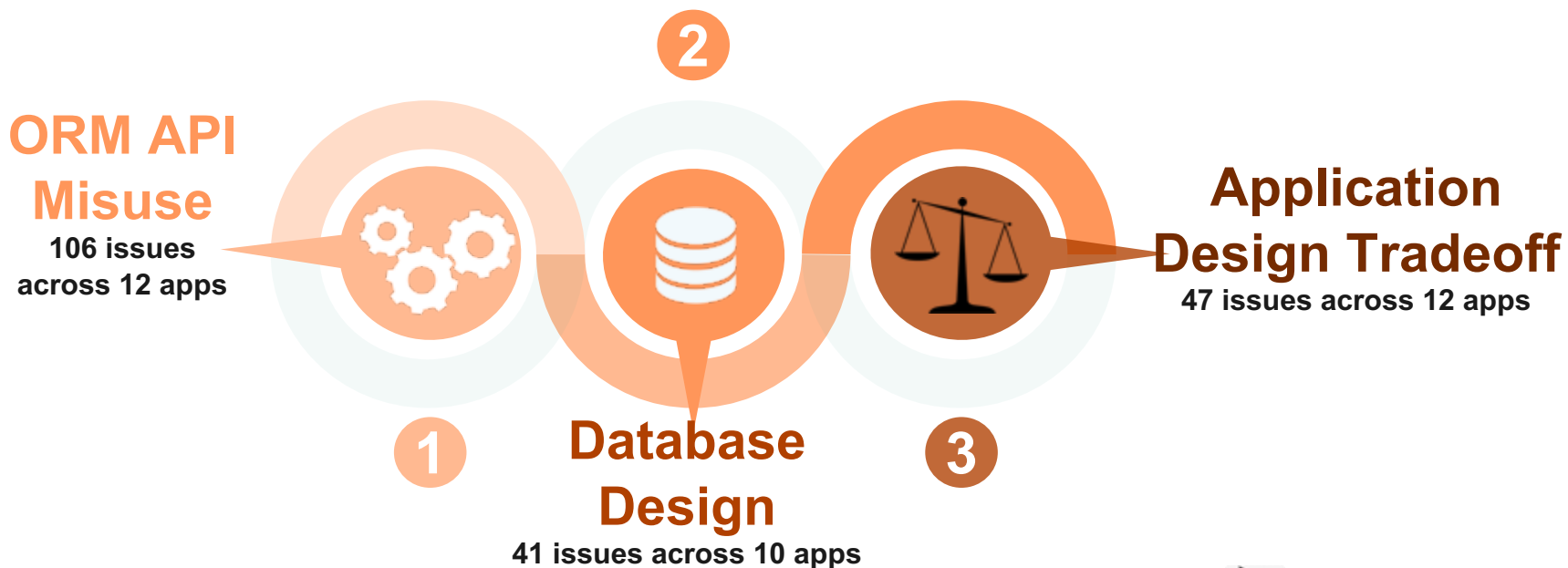
140 performance issues
from bug tracking system

9 anti-patterns





Common Performance Anti-patterns





ORM API Misuse

Inefficient Computation
26 issues across 8 apps



Unnecessary Computation
22 issues across 10 apps

Inefficient Data Access
44 issues across 11 app



Inefficient Rendering
5 issues across 4 apps

Unnecessary Data Retrieval
9 issues across 4 apps





ORM API Misuse

Inefficient Computation
26 issues across 8 apps



Unnecessary Computation
22 issues across 10 apps

Inefficient Data Access
44 issues across 11 app



Inefficient Rendering
5 issues across 4 apps

Unnecessary Data Retrieval
9 issues across 4 apps





ORM API Misuse: inefficient computation

project.issues.count>0

```
SELECT COUNT(*) FROM issues WHERE project_id = ?
```

inefficient

project.issues.any?

```
SELECT COUNT(*) FROM issues WHERE project_id = ?
```

inefficient

project.issues.exists?

```
SELECT 1 AS ONE FROM issues WHERE project_id = ? LIMIT 1
```

efficient



2X speedup





ORM API Misuse: unnecessary computation

SQL

```
values.each do |value|  
  u.issues.include? value  
end
```





ORM API Misuse: unnecessary computation

```
values.each do |value|  
-   u.issues.include?value  
end
```

```
+   rans = u.issues  
values.each do |value|  
+   rans.include?value  
end
```



20X speed up



ORM API misuses that affect memory consumption

- *map (:id) VS pluck (:id)*
- *pluck(size).sum VS sum(size)*
- *pluck + pluck VS SQL UNION*
- ...



How to tackle API Misuses?

- Why cannot existing compiler handle this?
- Can we extend compiler to
 - Understand ORM APIs and queries?
 - Detect the problem?
 - Solve the problem?



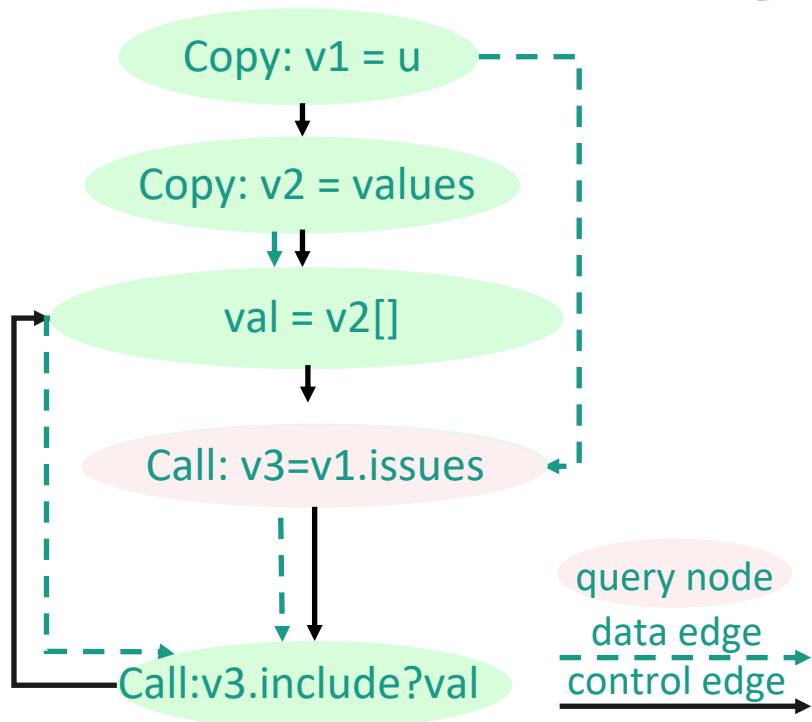
Database-aware PDG

```

v1 = u
v2 = values
values.reject |val|
  u.issues.include?val
end
end
    
```

```

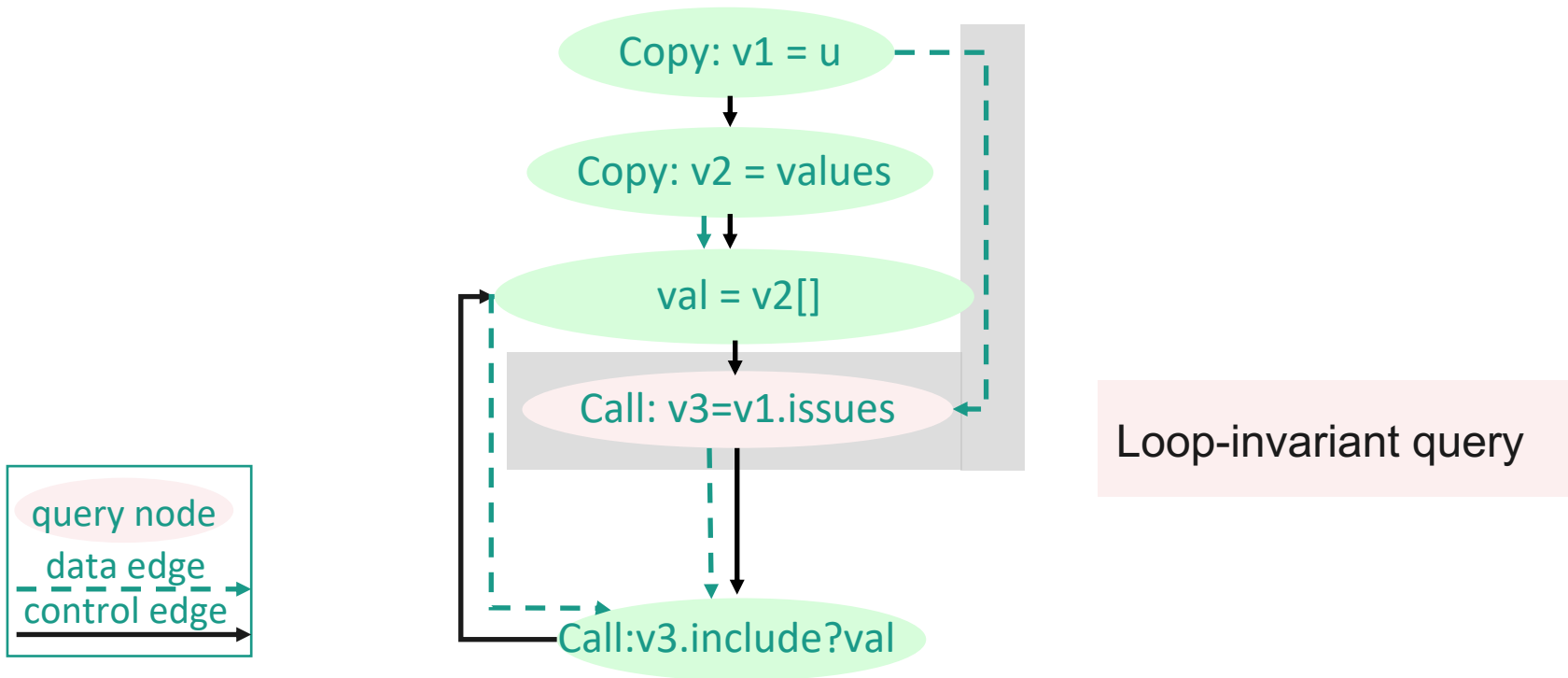
SQL: SELECT * from issues
      WHERE user_id=?
    
```



(b) PDG



Detect and Fix



PowerStation (Integrated with RubyMine)



The screenshot shows the RubyMine IDE interface. The top menu bar includes 'RubyMine', 'File', 'Edit', 'View', 'Navigate', 'Code', 'Refactor', 'Run', 'Tools', 'VCS', 'Window', and 'Help'. The title bar shows the current file path: 'blog [~/Research/blog] - .../app/controllers/blogs_controller.rb [blog]'. The breadcrumb navigation shows 'blog > app > controllers > blogs_controller.rb'. The main editor displays the following Ruby code:

```

1 class IssuesController < ApplicationController
2   def read_only_fields
3     ran = run_query
4     values.reject do |val|
5       ran.include?val
6     end
7   end
8 end

```

On the right side, the PowerStation sidebar is open, showing a table of issues:

PowerStation		issues
LI	IA	CS IR DS RD
blogs_controller.rb 4		FIX

Try our Powerstation!



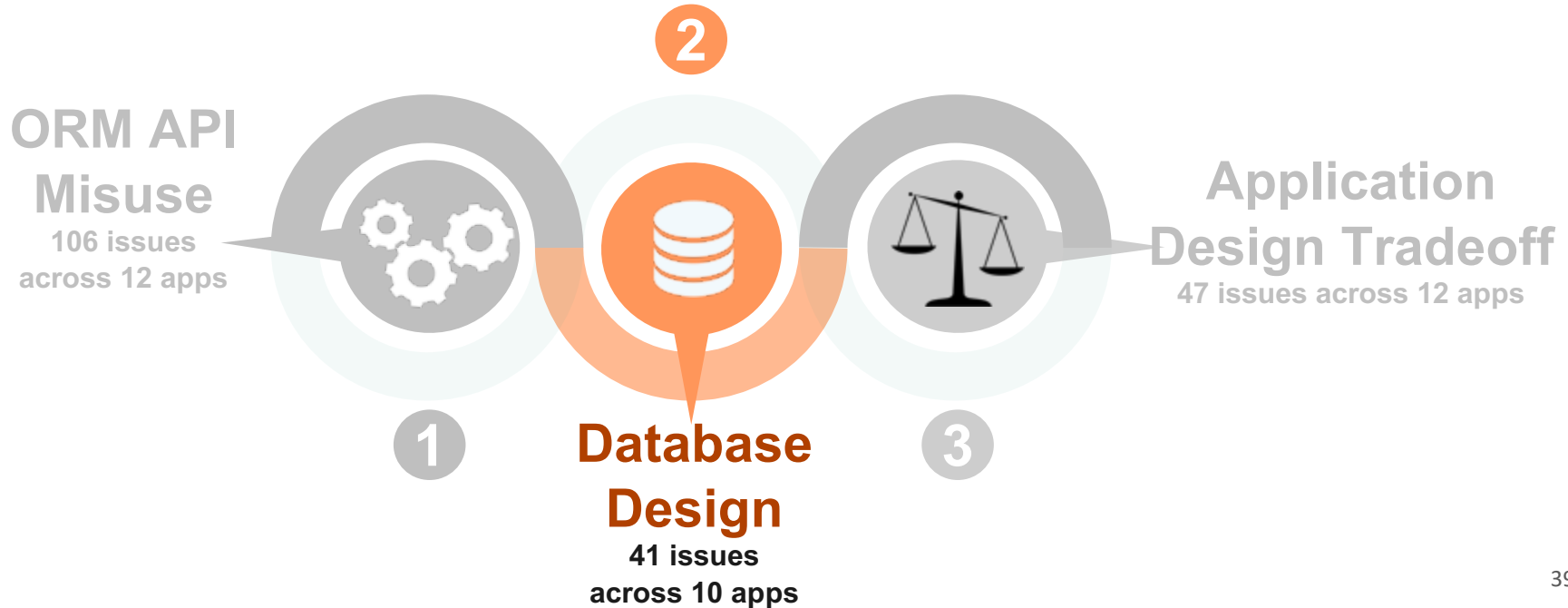
- 12 real world apps
- 1221 inefficiencies found

The screenshot shows the JetBrains Marketplace page for the 'powerstation' plugin. The page includes a navigation bar with links for IDEs, Team Tools, and Dev Guide, along with a user profile icon and a search icon. The main content area features the plugin name 'powerstation' in a large, bold font, followed by a 'Downloads' link. Below this, there is a compatibility section listing supported IDEs and versions, a date of release (Jul 17, 2018), and a download count (250). There are also links for 'License' and 'Vendor: PowerStation'. The 'Download plugin' section is highlighted, showing a table of available versions and their compatibility ranges. The table has columns for 'VERSION', 'COMPATIBILITY', and 'UPDATE DATE', with a 'DOWNLOAD' link for each row.

VERSION	COMPATIBILITY	UPDATE DATE
✓ 1.5-SNAPSHOT	181.4203-181.*	Jul 17, 2018
✓ 1.4-SNAPSHOT	181.4203-181.*	Jul 15, 2018
✓ 1.3-SNAPSHOT	181.4203-181.*	Jun 19, 2018



Common Performance Anti-patterns





Database Design Problem

- Missing fields (8 issues across 5 apps):
 - fields derivable from other fields and not persistently stored

id	longitude	latitude	location



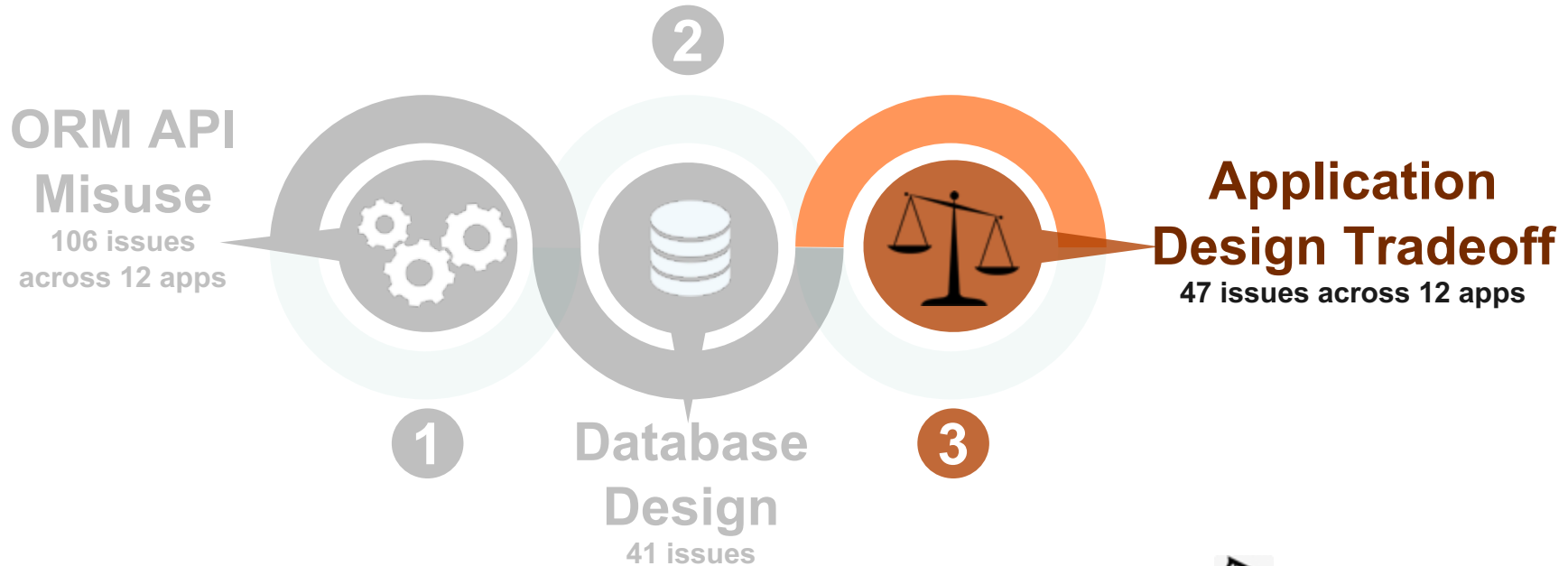
2X

- Missing index (33 issues across 10 apps)





Common Performance Anti-patterns



1001 unread blogs

- Arriving at Zurich
- Stopping by Bern
- One day at Luzern
- Love love Berner Oberland
- Love Berner Oberland
- Back to Lausanne

A vertical scrollbar is highlighted with a red box, showing the current scroll position.

1001 unread blogs

- Arriving at Zurich
- Stopping by Bern
- One day at Luzern
- Love love Berner Oberland
- Love love Berner Oberland

A pagination control is highlighted with a red box, showing page numbers 1, 2, 3, and ellipsis, along with navigation arrows.

1001 unread blogs

- Arriving at Zurich
- Stopping by Bern
- One day at Luzern
- Love love Berner Oberland
- Love Berner Oberland

< 1 2 3 ... >

This screenshot shows a web browser window with the address bar containing "http://blogs/index". The page displays a list of five blog entries, each with a user icon, the title, and heart and thumbs-up icons. A red box highlights the text "1001 unread blogs" at the top of the list. At the bottom, there is a pagination control with buttons for "<", "1", "2", "3", "...", and ">".

More than 20 unread blogs

- Arriving at Zurich
- Stopping by Bern
- One day at Luzern
- Love love Berner Oberland
- Love Berner Oberland

< 1 2 3 ... >

This screenshot shows the same web browser window and page content as the first screenshot. However, the text at the top of the list now reads "More than 20 unread blogs". The pagination control at the bottom remains the same.



Submit a Story

URL:

Title:

Tags:

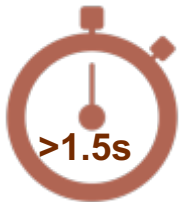
Text: *Optional when submitting a URL; please see guidelines*

```
SELECT count(*) FROM moderations JOIN stories where
stories.user_id = @user.id AND moderations.created_at > 5.days.ago
```



Story submission guidelines

- To be able to easily submit a page you're viewing in your browser to Example News, drag the bookmarklet to the right to your bookmark bar. You'll be taken to this page with the viewed page's URL and title.
- When submitting a URL, the text field is optional and should only be used when additional context or explanation of the URL is needed. Commentary or opinion should be reserved for a comment, so that it can be voted on separately from the story.
- Do not editorialize story titles, but when the original story's title has no context or is unclear, please change it. **Please remove extraneous components from titles such as the name of the site or section.**
- If no tags clearly apply to the story you are submitting, chances are it does not belong here. Do not overreach with tags if they are not the primary focus of the story.
- When the story being submitted is more than a year or so old, please add the year the story was written to the post title in parentheses.



Whether to show this guideline

Author: I am the author of the story at this URL (or this text)

[Markdown formatting available](#)

How to tackle application design tradeoffs?



- Can we do automated optimization?
- Help developers make informed decision, by providing
 - Cost information
 - Alternative display/functionality options



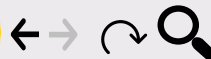


`app/controllers/blogs_controller.rb`

```
def index
  @blogs = blog.all
  render "index"
end
```

`app/views/blogs/index.html.erb`



















```
@blogs.each do |blog|
  blog.content<br/>
end
```



`http://blogs/index`



1001 unread blogs

-  Arriving at Zurich  
-  Stopping by Bern  
-  One day at Luzern  
-  Love love Berner Oberland  
-  Love Berner Oberland  
-  Back to Lausanne  





app/controllers/blogs_controller.rb

```
def index
  @blogs = blog.all
  render "index"
end
```

app/views/blogs/index.html.erb

```
@blogs.each do |blog|
  blog.content<br/>
end
```



http://blogs/index



1001 unread blogs



Arriving at Zurich



Stopping by Bern



One day at Lu



Love love Berner Oberland



Love Berner Oberland



Back to Lausanne



pagination





app/controllers/blogs_controller.rb

```
def index
  @blogs = Blog.all.paginate(...)
  render "index"
end
```

app/views/blogs/index.html.erb

```
@blogs.each do |blog|
  blog.content<br/>
end
will_paginate @blogs
```



http://blogs/index



1001 unread blogs



Arriving at Zurich



Stopping by Bern



One day at Luzern



Love love Berner Oberland



Love Berner Oberland





app/controllers/blogs_controller.rb

```
def index
  @blognum = blog.count
  render "index"
end
```

app/views/blogs/index.html.erb

There are @blognum blogs



http://blogs/index



1001 unread blogs



Arriving at Zurich

remove



Stopping by Bern

approximation



One day at Luzern

async loading



Love love Berner Oberland



Love Berner Oberland





app/controllers/blogs_controller.rb

```
def index
  @blognum = blog.limit(21).count
  render "index"
end
```

app/views/blogs/index.html.erb

```
There are
@blognum>20?'more than 20':@blognum
blogs
```



http://blogs/index



more than 20 unread blogs



Arriving at Zurich

remove

async loading



Stopping by Bern



One day at Luzern



Love love Berner Oberland



Love Berner Oberland





app/controllers/blogs_controller.rb

```
def index  
  @blognum = blog.count  
  render "index"  
end
```

app/views/blogs/index.html.erb

```
@blognum unread_blogs
```



http://blogs/index



Arriving at Zurich



Stopping by Bern



One day at Luzern



Love love Berner Oberland



Love Berner Oberland



Try our Panorama!



- 12 real world apps
- 149 performance-enhancing opportunities identified for 119 costly HTML tags
- 4.5X average page-load time speedup
- User study agrees!



Slow downs in web applications



Junwen Yang



Real world database-backed applications perform poorly



Data-related performance anti-patterns exist



Automatic tools are built to detect and fix performance issues



What stopped cloud services?

Efficient and Scalable Thread-Safety Violation Detection --- Finding thousands of concurrency bugs during testing. *SOSP'19*

DFix: Automatically Fixing Timing Bugs in Distributed Systems. *PLDI'19*

FCatch: Automatically detecting time-of-fault bugs in cloud systems. *ASPLOS'18*

DCatch: Automatically Detecting Distributed Concurrency Bugs in Cloud Systems. *ASPLOS'17*

TaxDC: A Comprehensive Taxonomy of Non-Deterministic Concurrency Bugs in Cloud Distributed Systems. *ASPLOS'16*.

What Bugs Cause Production Cloud Incidents? *HotOS'19*



UCHICAGO



Microsoft

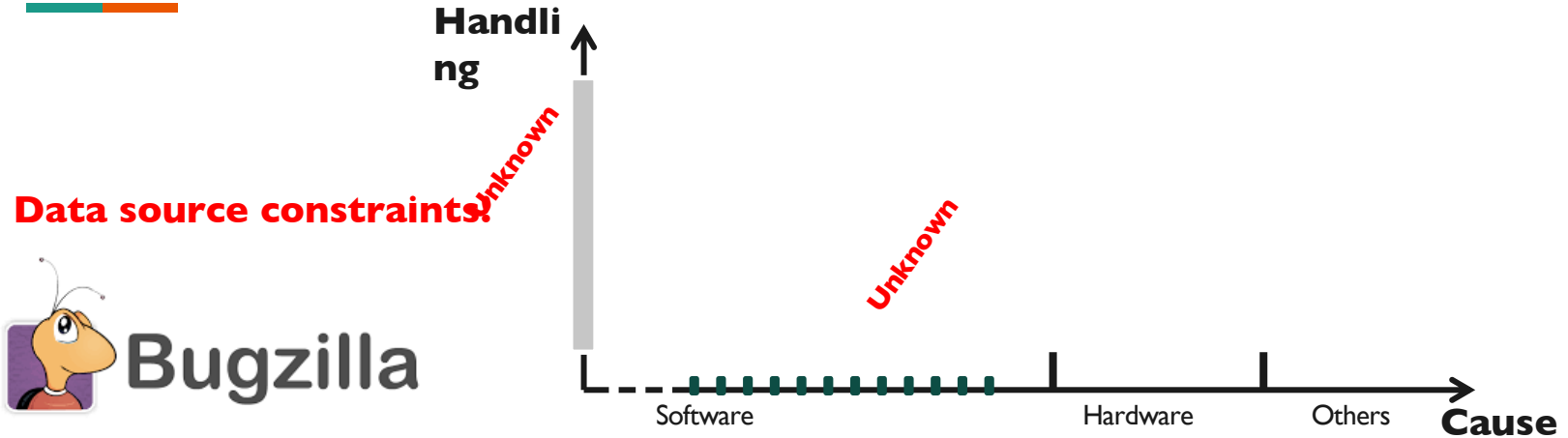
Need to study real-world cloud incidents



Handling

Cause

Existing studies for cloud incidents



- [6] Leesatapornwongsa. TaxDC. In ASPLOS'16
- [5] Leesatapornwongsa. Scalability bugs. In HotOS'17
- [4] Huang. Gray failure. In HotOS'17
- [3] Yuan. Simple test can prevent most critical failures. In OSDI'14
- [2] Gunawi. Why does the cloud stop computing? In SoCC'16
- [1] Gunawi. What bugs live in the cloud? In SoCC'14

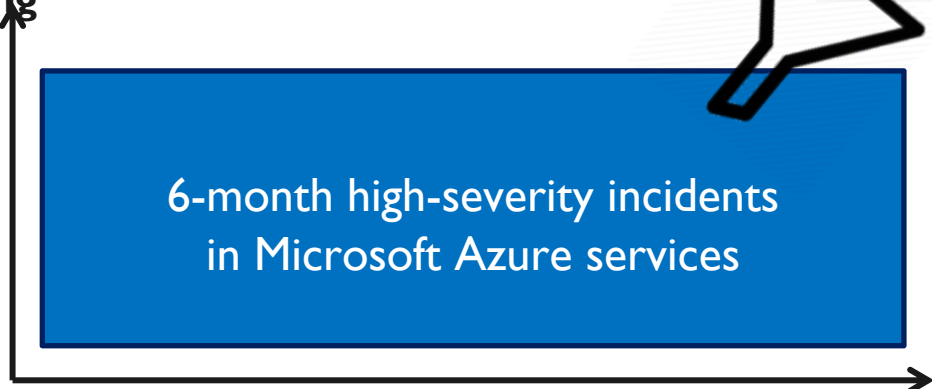


Our work

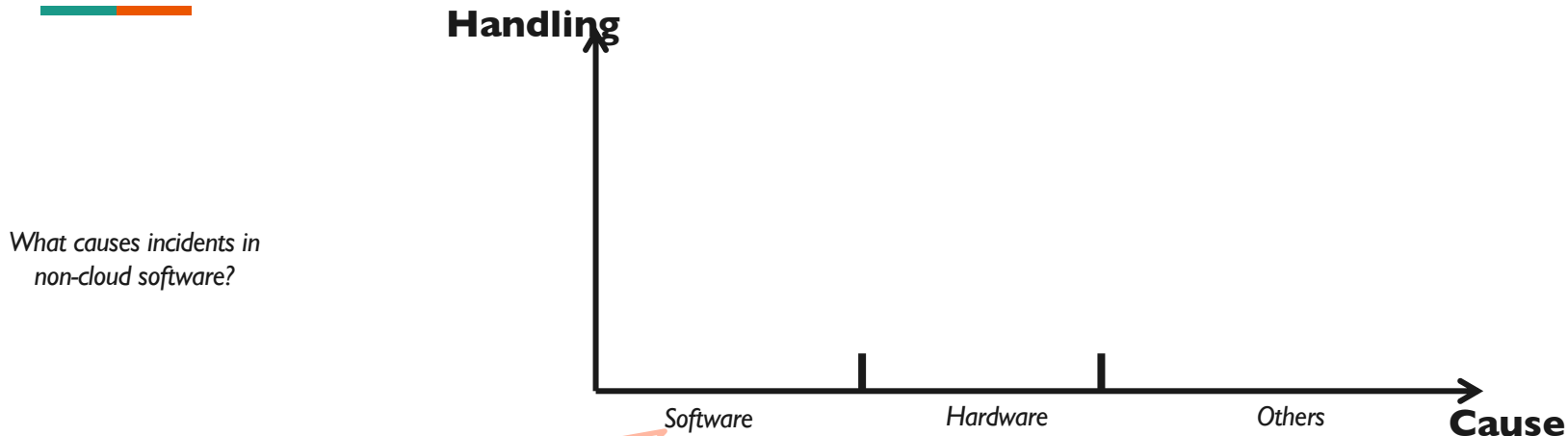
Handling

6-month high-severity incidents
in Microsoft Azure services

Cause



One more background ...



Concurrency bugs

Semantic bugs

Memory bugs

Our findings

Handling

6-month high-severity incidents
in Microsoft Azure services



Software

Hardware

Others

Cause

Concurrency bugs

Semantic bugs

Memory bugs

Our findings

Handling

6-month high-severity incidents
in Microsoft Azure services

Software

Hardware

Others

Cause



Concurrency bugs

Semantic bugs

Memory bugs

Our findings

Handling

6-month high-severity incidents
in Microsoft Azure services

Software

Hardware

Others

Cause



Concurrency bugs

Memory bugs

Semantic bugs

Our findings

Handling

6-month high-severity incidents
in Microsoft Azure services

Software

Hardware

Others

Cause

Concurrency bugs

Semantic bugs

Resource (memory) leaks

Our findings

Handling

6-month high-severity incidents
in Microsoft Azure services

Software

Hardware

Others

Cause



Concurrency bugs
(50% persistent races)

Semantic bugs

Resource (memory) leaks

Our findings

Handling

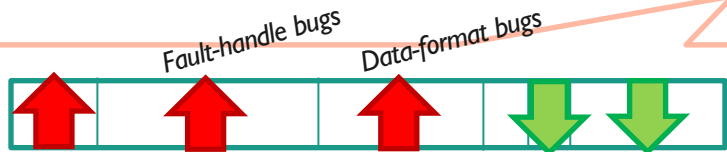
6-month high-severity incidents
in Microsoft Azure services

Software

Hardware

Others

Cause

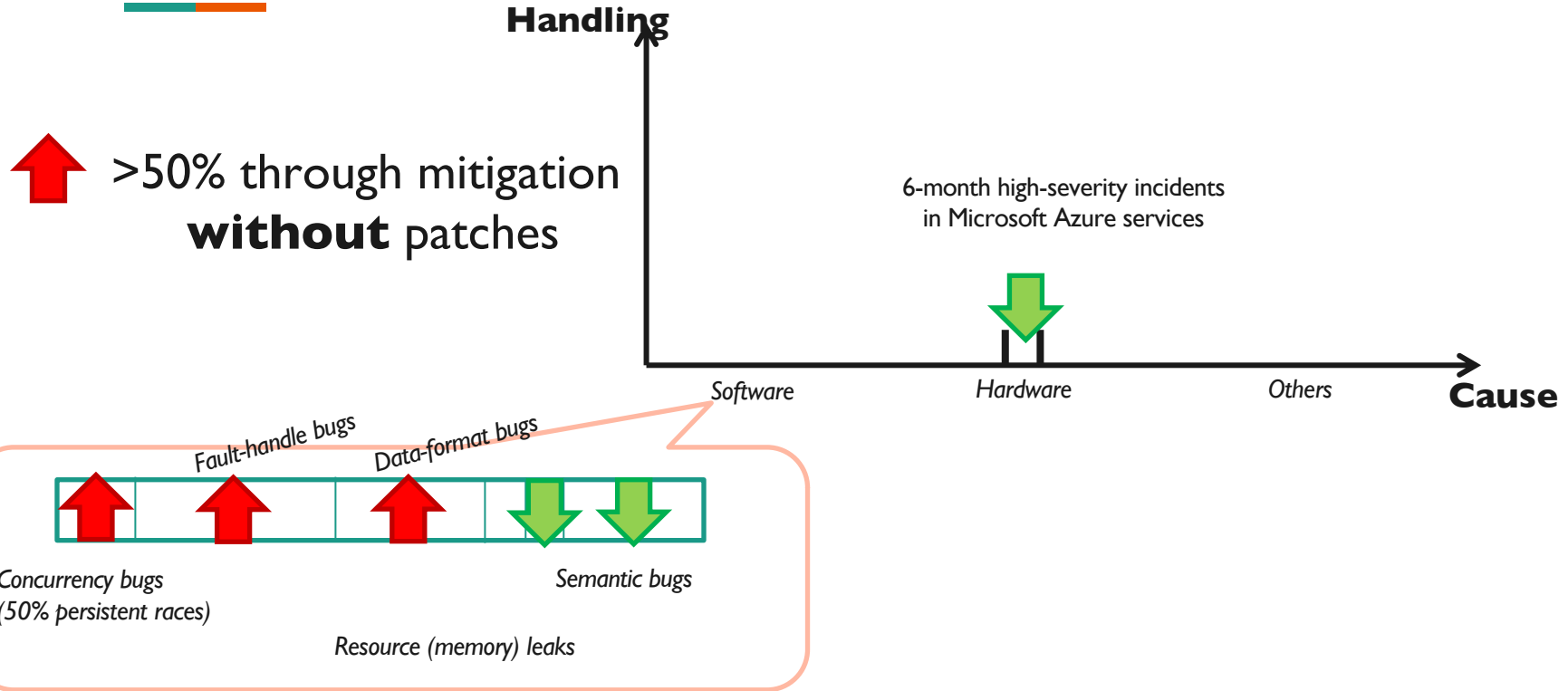


Concurrency bugs
(50% persistent races)

Resource (memory) leaks

Semantic bugs

Our findings



DFix: Automatically Fixing Timing Bugs in Distributed Systems

Guangze Li University of Chicago, USA
Haoyang Li University of Chicago, USA
Xiangfan Chen* University of Chicago, USA
Shan Lu* University of Chicago, USA
Harvard S. Gomani University of Chicago, USA
Shan Lu University of Chicago, USA



Figure 3. A message timing bug in MapReduce [1].

1. Introduction
 Distributed systems such as main data storage systems [2], [3], [4], [5] and cloud computing environments [1], [6] are the backbone of our computing ecosystem. High availability of these systems is crucial, with minutes of outage costing millions of dollars [7], [8]. We recently identified an addressable class of bugs, particularly distributed timing bugs (e.g., [1], [9]), that are triggered by non-deterministic timing of message transmission or a component failure (e.g., [10], [11]).

DFix is a tool that automatically fixes timing bugs in distributed systems. It automatically generates code patches that fix the bugs. DFix is implemented as a set of plugins that can be applied to a wide range of distributed systems. It is implemented in Python and runs on Linux. It is implemented as a set of plugins that can be applied to a wide range of distributed systems. It is implemented in Python and runs on Linux.

FCatch: Automatically Detecting Time-of-fault Bugs in Cloud Systems

Guangze Li University of Chicago, USA
Haoyang Li University of Chicago, USA
Shan Lu University of Chicago, USA
Xiangfan Chen* University of Chicago, USA
Harvard S. Gomani University of Chicago, USA
Shan Lu University of Chicago, USA

we do?
Handling

mitigation
: pa

DCatch: Automatically Detecting Distributed Concurrency Bugs in Cloud Systems

Guangze Li University of Chicago, USA
Jeffrey F. Lukman IBM Research, USA
Harvard S. Gomani University of Chicago, USA
Chen Tian* IBM Research, USA
Shan Lu* University of Chicago, USA

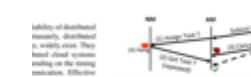


Figure 4. A timing bug in MapReduce [1].

5. Introduction
 Distributed systems such as main data storage systems [2], [3], [4], [5] and cloud computing environments [1], [6] are the backbone of our computing ecosystem. High availability of these systems is crucial, with minutes of outage costing millions of dollars [7], [8]. We recently identified an addressable class of bugs, particularly distributed timing bugs (e.g., [1], [9]), that are triggered by non-deterministic timing of message transmission or a component failure (e.g., [10], [11]).

DCatch is a tool that automatically detects distributed concurrency bugs in cloud systems. It automatically generates code patches that fix the bugs. DCatch is implemented as a set of plugins that can be applied to a wide range of distributed systems. It is implemented in Python and runs on Linux.

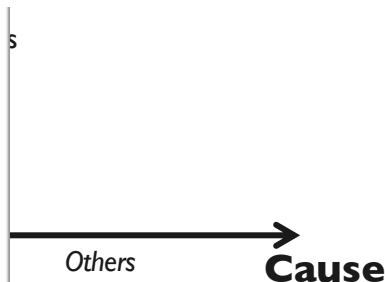
Efficient Scalable Thread-Safety-Violation Detection

Guangze Li University of Chicago, USA
Simon Nath Microsoft Research, USA
Shan Lu University of Chicago, USA
Michael Mross Microsoft Research, USA
Robert Palfrey University of California, Berkeley, USA

Abstract
 Concurrency bugs are hard to find, reproduce, and debug. They often emerge sporadically in testing, but result in long-scale outages in production. Existing concurrency-bug detection techniques suffer from either being too expensive for production systems, or being too imprecise to find bugs in large-scale systems. We present a new method for detecting concurrency bugs in production systems. Our method is implemented as a set of plugins that can be applied to a wide range of distributed systems. It is implemented in Python and runs on Linux.

1. Introduction
 Concurrency bugs are hard to find, reproduce, and debug. They often emerge sporadically in testing, but result in long-scale outages in production. Existing concurrency-bug detection techniques suffer from either being too expensive for production systems, or being too imprecise to find bugs in large-scale systems. We present a new method for detecting concurrency bugs in production systems. Our method is implemented as a set of plugins that can be applied to a wide range of distributed systems. It is implemented in Python and runs on Linux.

DFix is a tool that automatically fixes timing bugs in distributed systems. It automatically generates code patches that fix the bugs. DFix is implemented as a set of plugins that can be applied to a wide range of distributed systems. It is implemented in Python and runs on Linux.



github.com/microsoft/TSVD



Concurrency bugs (50% persistent races)

Conclusions

- Software bugs widely exist in big data & cloud systems
- Software bugs are taking on new forms in big data & cloud systems
 - Memory data \leftrightarrow Persistent data
- A lot of bug fighting can be done and to be done
- Our are making our bug set and tools open source!



Junwen Yang



Guangpu Li



Thanks!

