FluxRank: A Widely-Deployable Framework to Automatically Localizing Root Cause Machines for Software Service Failure Mitigation

Ping Liu[1], Yu Chen[2], Xiaohui Nie[1], Jing Zhu[1], Shenglin Zhang[3], Kaixin Sui[4]
Ming Zhang[5], Dan Pei[1]

1 清華大學 Tsinghua University
2 Baidu 百度
3 南開大學 Nankai University
4 必示 BIZSEER
5 中国建设银行 China Construction Bank

Background

Design

Evaluation

Case Study

Background

Design

Evaluation

Case Study

Background

Why we focus on **failure mitigation** ?

Because it took **too long** for
a complex distributed service

Service outages of 2019

Three and a half hours before successful mitigation

Google Gmail and Drive: March 12

Major problems with Gmail and Google Drive were first reported just before 8 pm PT on the evening of March 12, followed by glitches affecting YouTube.

On its status page, Google said users were seeing "error messages, high latency, and/or other unexpected behavior."

Gmail users complained of problems sending emails. Google Drive users reported that certain files weren't opening, and that performance of the cloud storage solution was degraded. The outages lasted for roughly three-and-a-half hours.

Source: https://www.crn.com/slide-shows/cloud/the-10-biggest-cloud-outages-of-2019-so-far-

Service outages of 2019

Almost a full day before successful mitigation

Facebook, Instagram: March 13

Facebook and its photo-sharing subsidiary, Instagram, both suffered partial service outages the morning of March 13. The outages not only impacted consumers, but also developers building apps on the world's largest social network.

A Facebook engineer on the company's server status page initially wrote the company was "experiencing issues that may cause some API requests to take longer or fail unexpectedly."

It took almost a full day before error rates returned to normal.

Source: https://www.crn.com/slide-shows/cloud/the-10-biggest-cloud-outages-of-2019-so-far-

Service outages of 2019

Almost three hour before successful mitigation

Microsoft Azure: May 2

Several core Microsoft cloud services, including compute, storage, an application development platform, Active Directory and SQL database services, were impacted by a nearly three-hour DNS outage on May 2.

Some of Microsoft's cloud-based applications, including Microsoft 365, Dynamics and Azure DevOps, were also impacted.

According to Microsoft's Azure status page, the underlying root cause was a nameserver change that affected DNS resolution, harming the downstream services.

Source: https://www.crn.com/slide-shows/cloud/the-10-biggest-cloud-outages-of-2019-so-far-

Background

So Long !!!

...

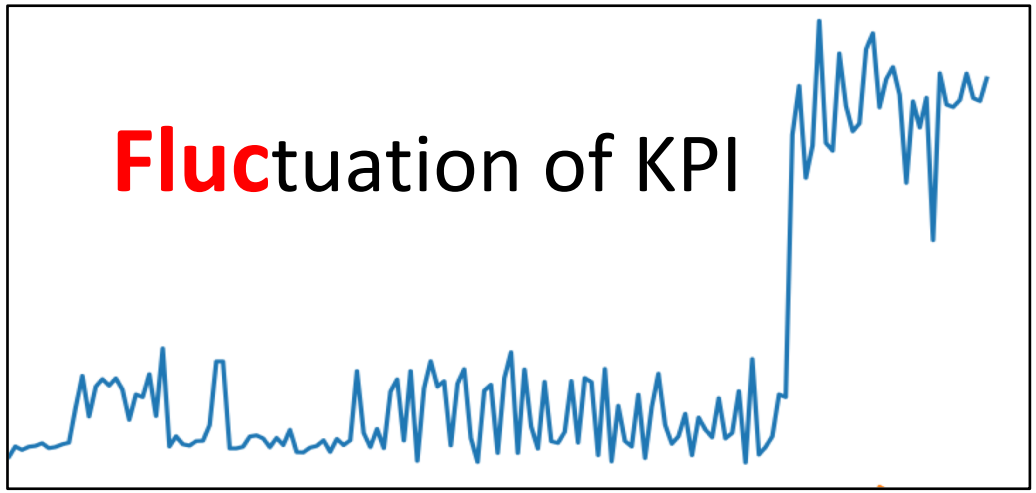
Mitigation Time

- Three and a half hours
- A full day
- Three hour
- ...

Our algorithm cuts the mitigation time by more than 80% on average.

FluxRank

Rank

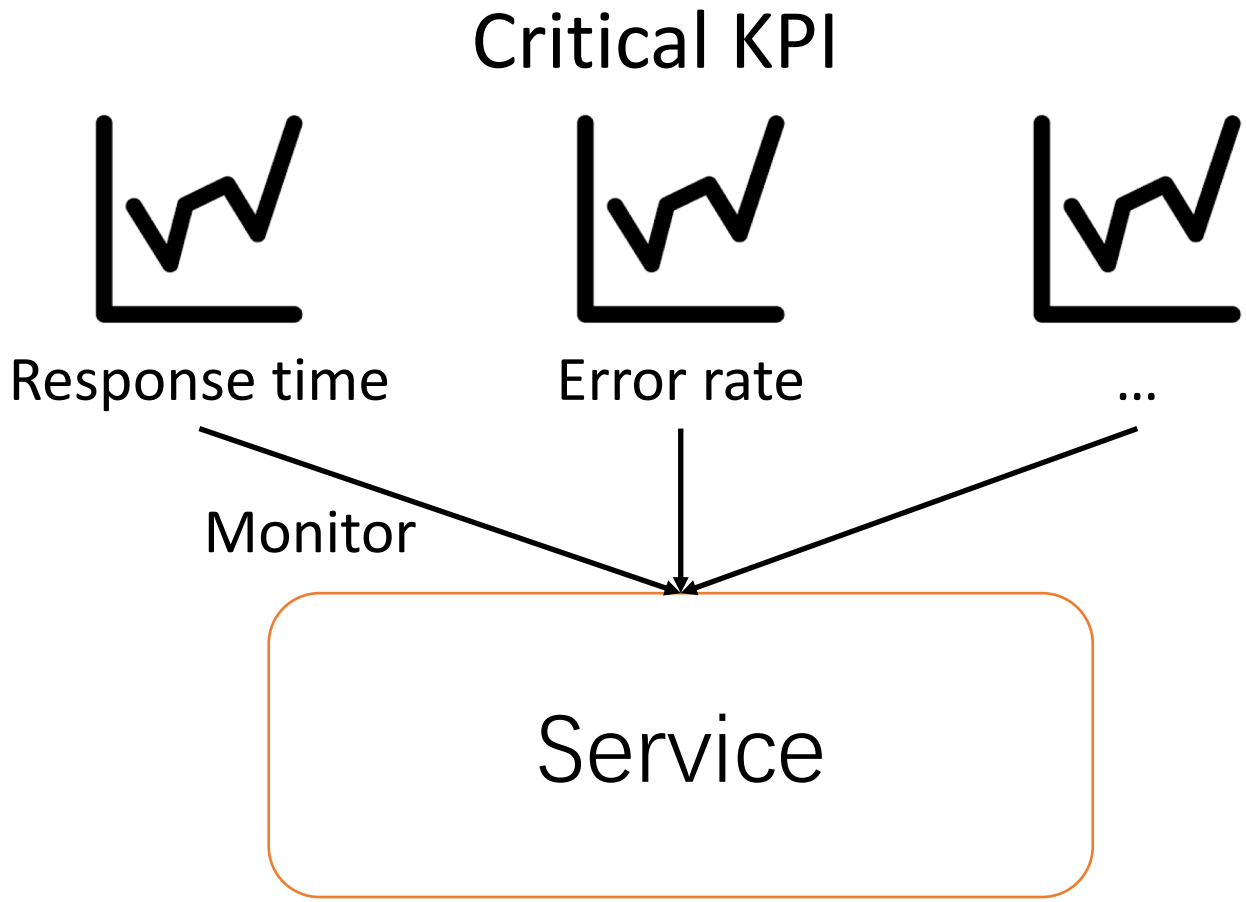


Background

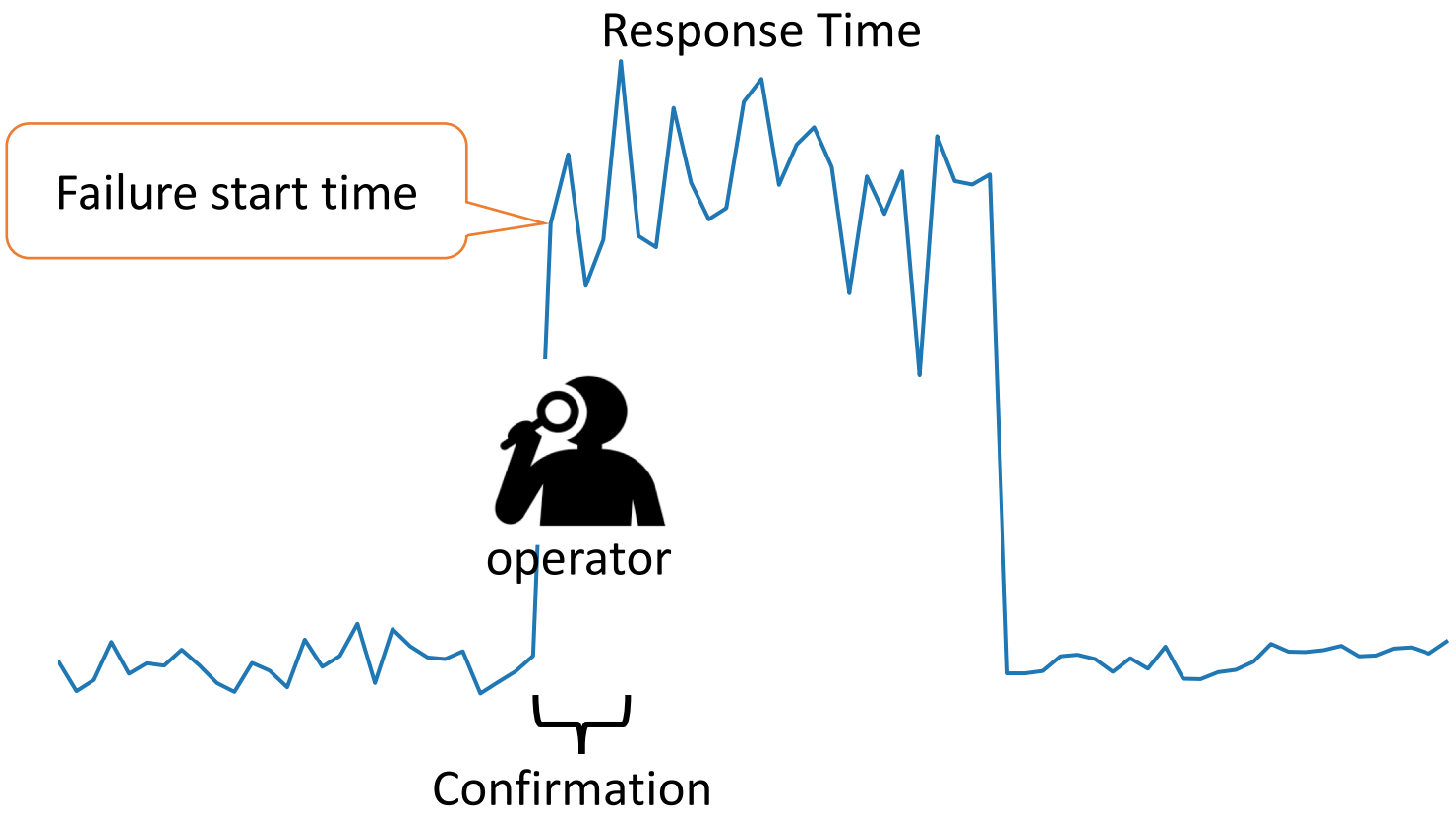
Failure mitigation takes too much time.

Why?

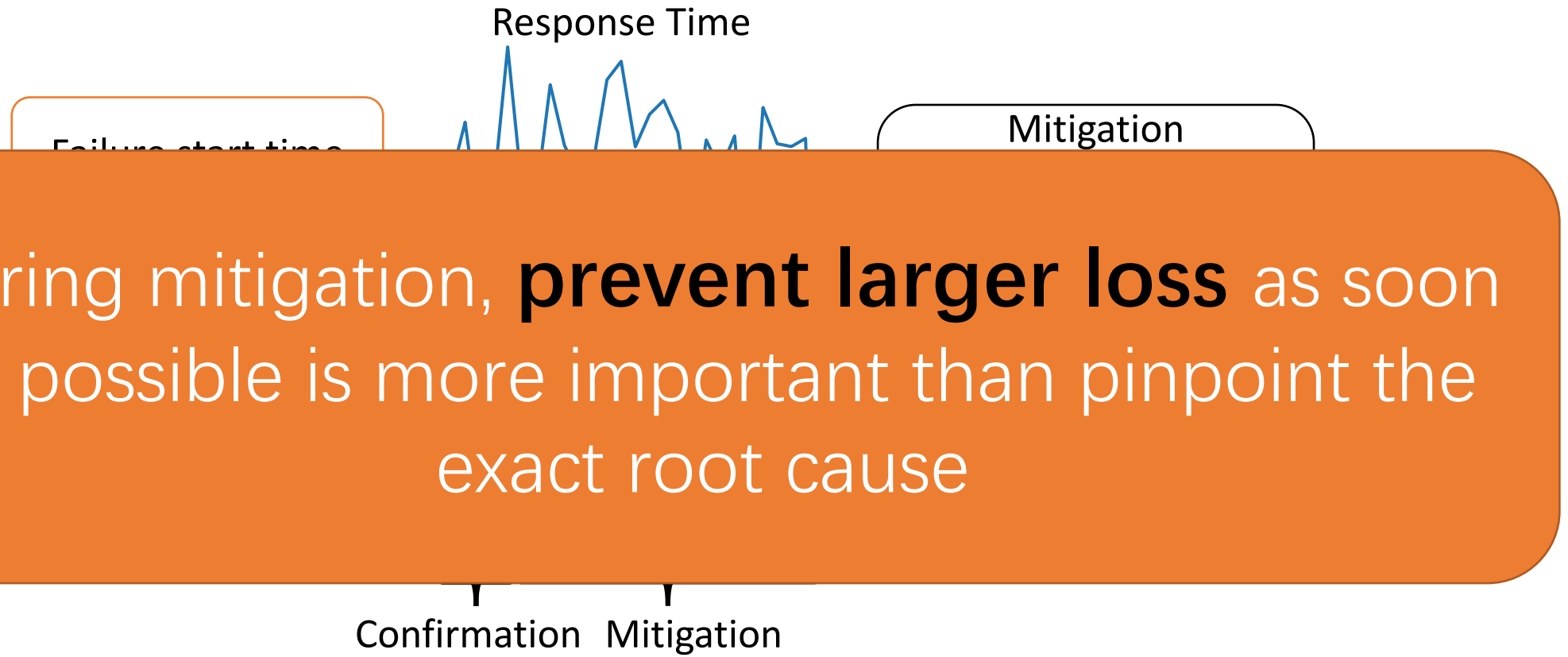
Troubleshooting process



Troubleshooting process

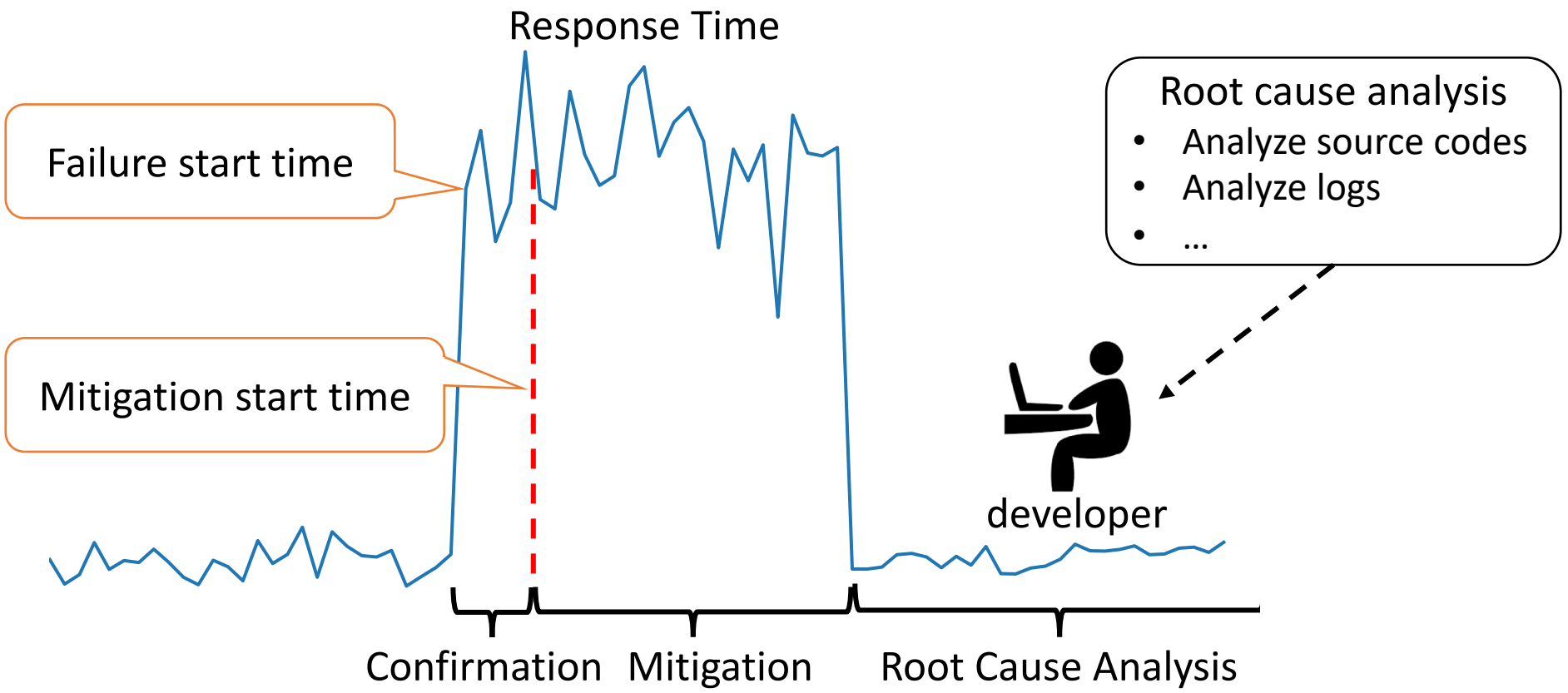


Troubleshooting process

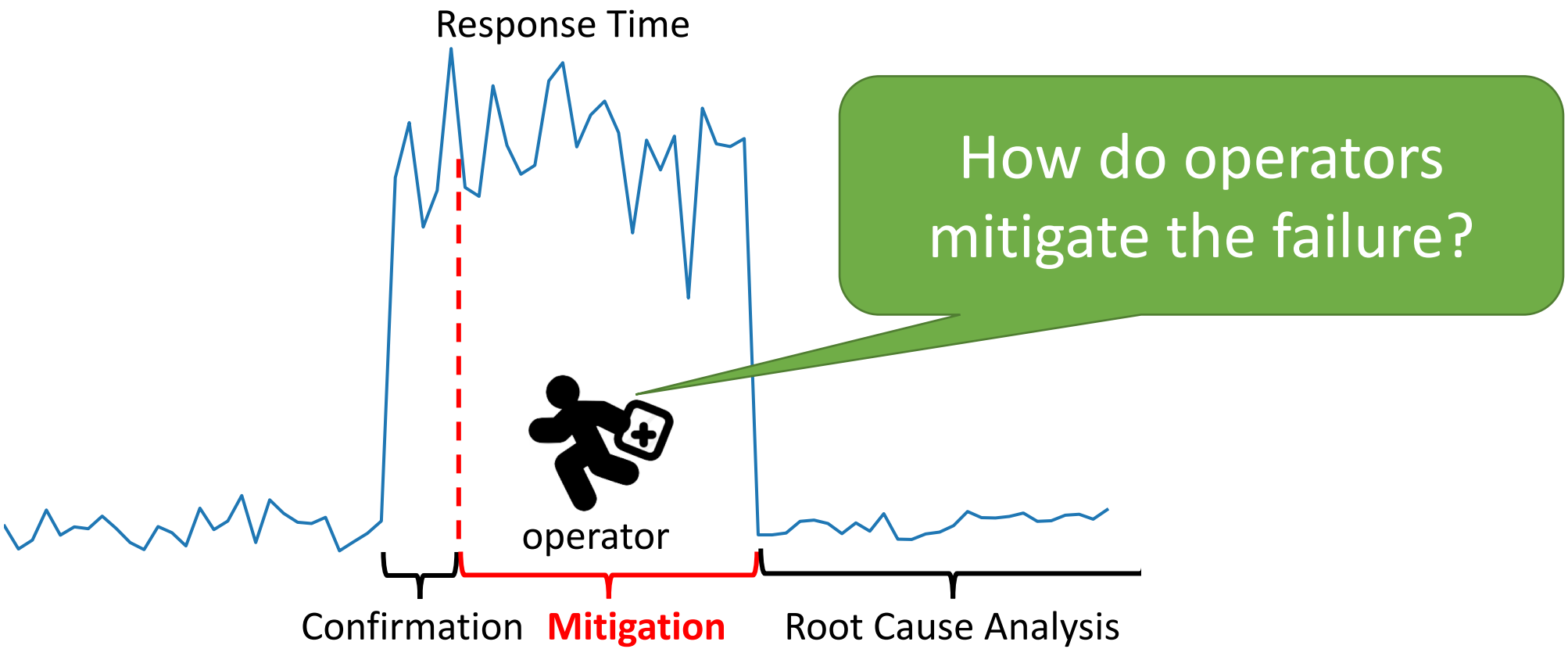


During mitigation, **prevent larger loss** as soon as possible is more important than pinpoint the exact root cause

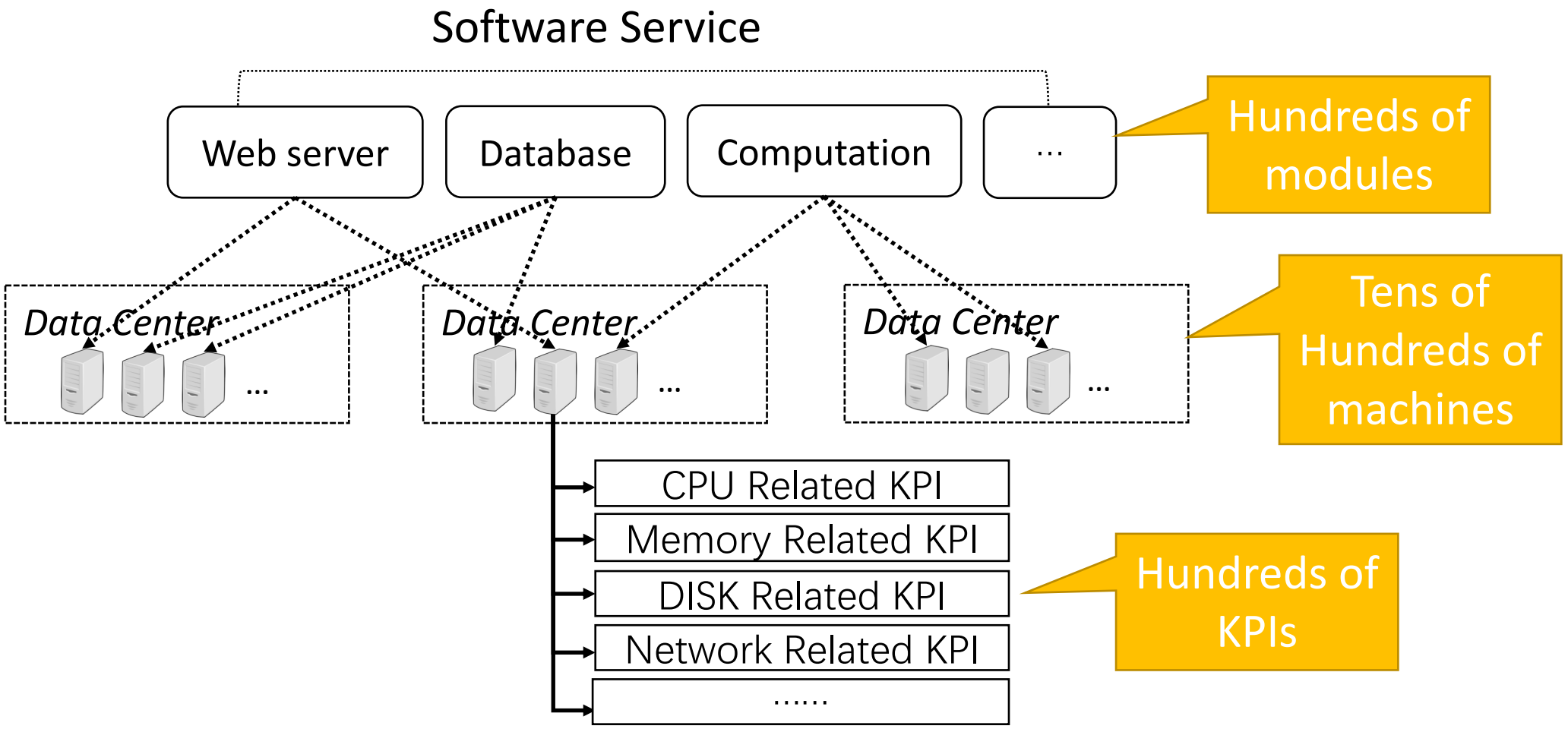
Troubleshooting process



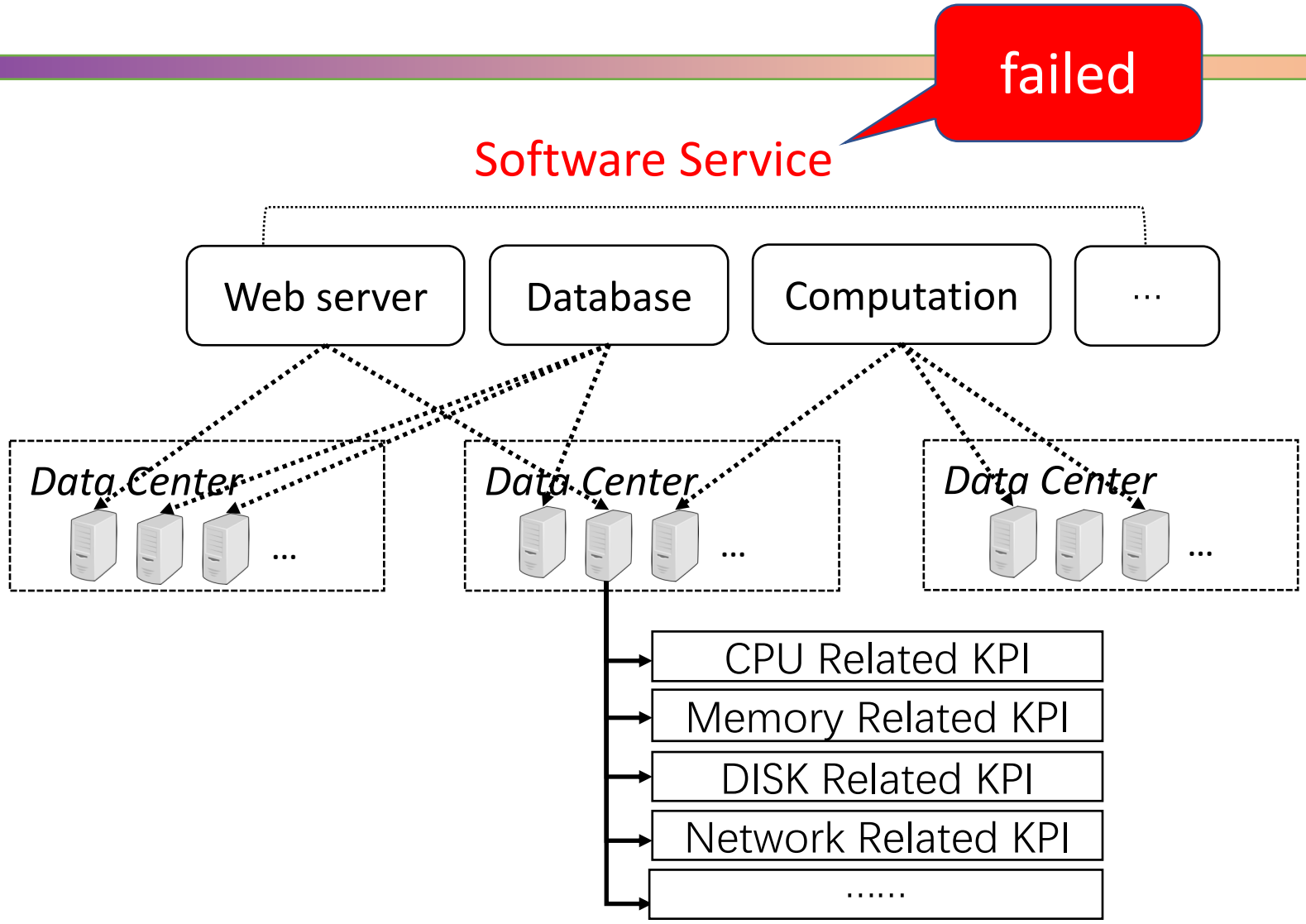
Troubleshooting process



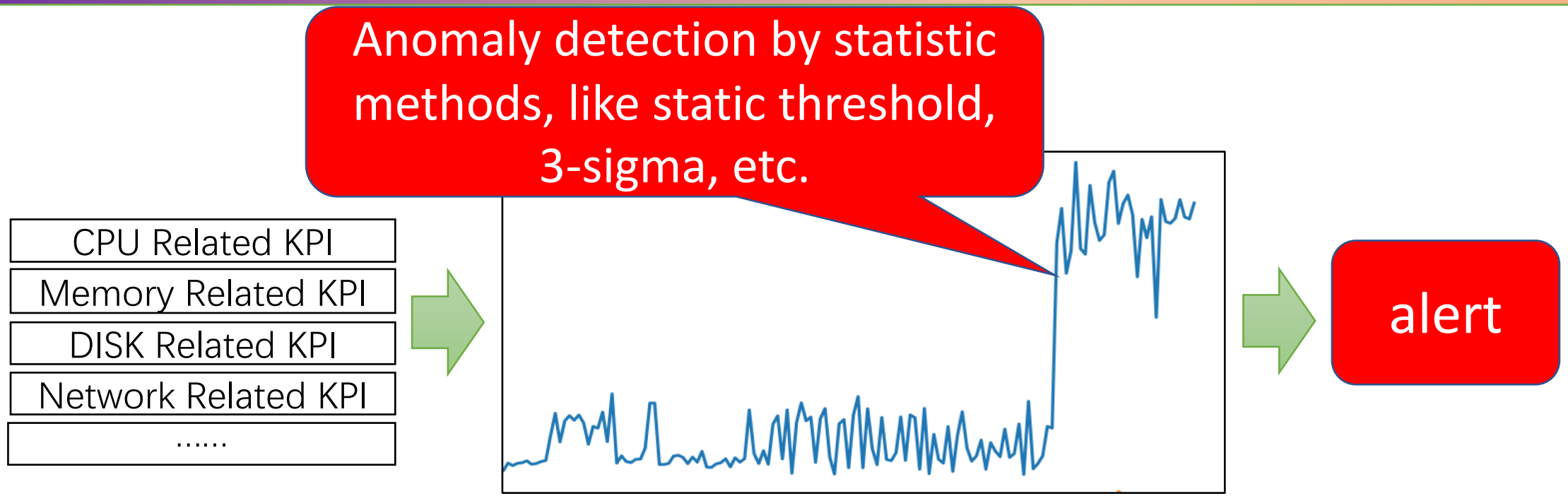
Mitigation


Software Service
Web server
Database
Computation
…
Hundreds of modules
Data Center
Data Center
Data Center
Tens of Hundreds of machines
CPU Related KPI
Memory Related KPI
DISK Related KPI
Network Related KPI
……
Hundreds of KPIs

Mitigation


failed
Software Service
Web server
Database
Computation
…
Data Center
Data Center
Data Center
CPU Related KPI
Memory Related KPI
DISK Related KPI
Network Related KPI
……

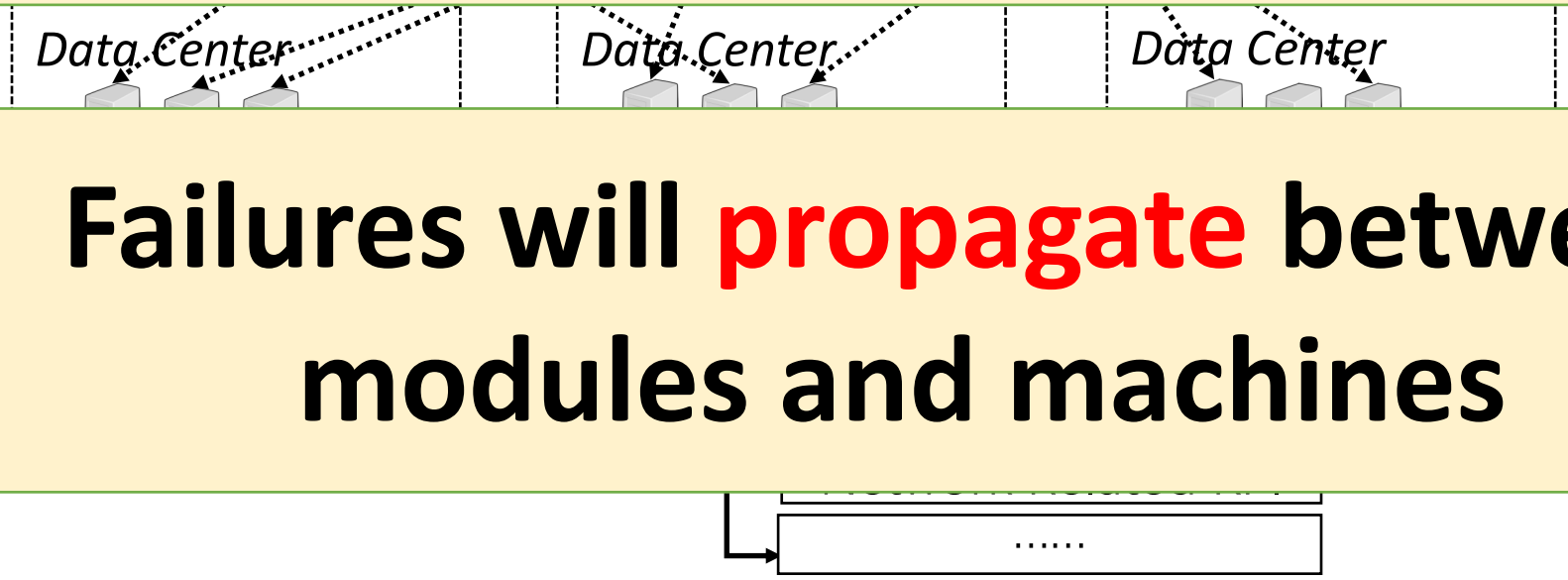
Mitigation


Anomaly detection by statistic methods, like static threshold, 3-sigma, etc.
CPU Related KPI
Memory Related KPI
DISK Related KPI
Network Related KPI
.....
alert

Mitigation

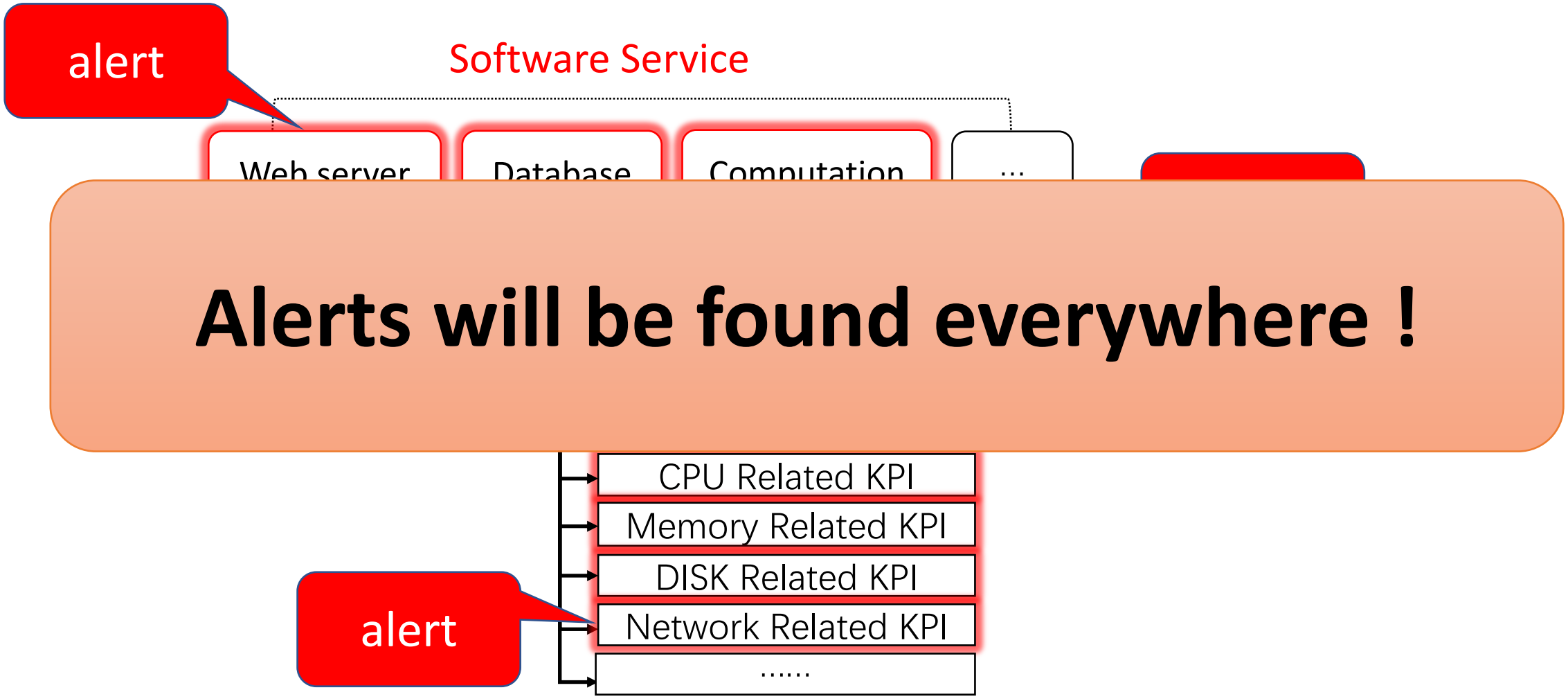
failed

Because of the dependencies between modules and machines

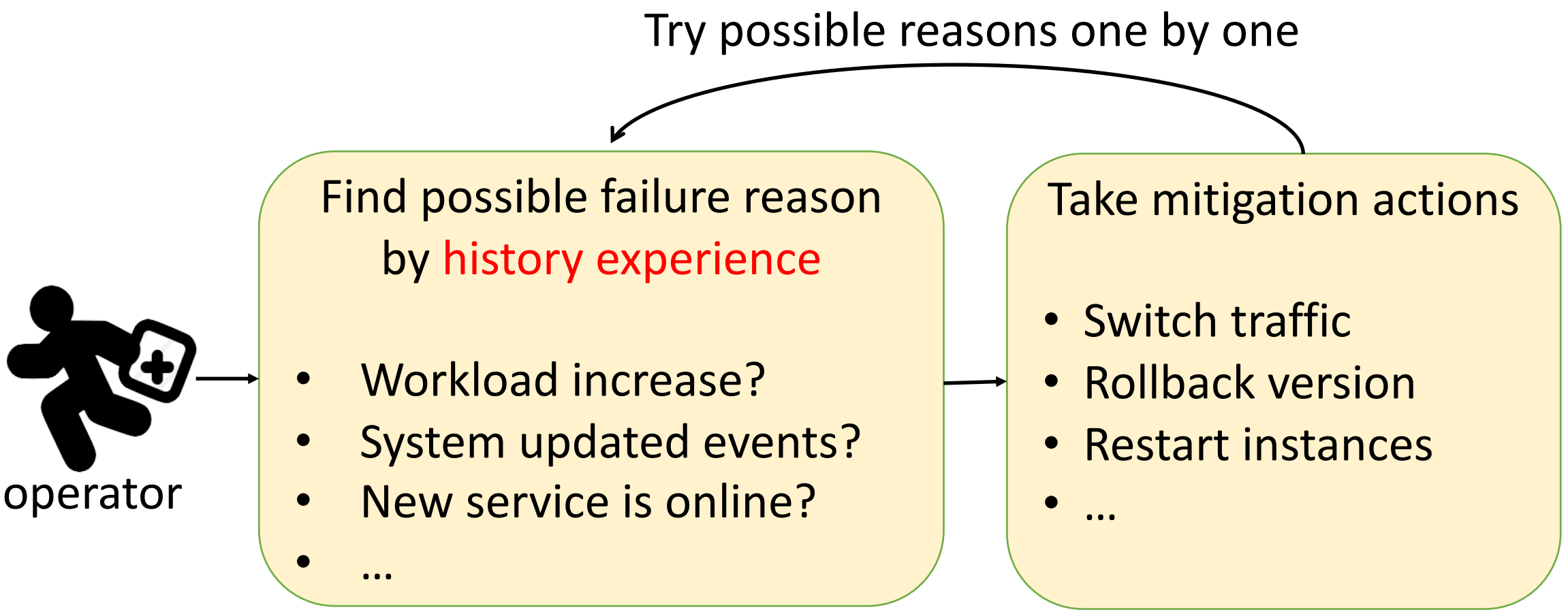


Failures will propagate between modules and machines

Mitigation


alert
Software Service
Web server
Database
Computation
…
Alerts will be found everywhere !
CPU Related KPI
Memory Related KPI
DISK Related KPI
Network Related KPI
……
alert

Mitigation


Try possible reasons one by one
Find possible failure reason by history experience
• Workload increase?
• System updated events?
• New service is online?
• …
Take mitigation actions
• Switch traffic
• Rollback version
• Restart instances
• …
operator

Mitigation

If the mitigation is failed by trying possible reasons

Operators will manually scan KPIs to find the root cause location

Mitigation

Why are operators reluctant to check the codes and exception logs?

developer

Only service developers can understand the details of codes and exception logs

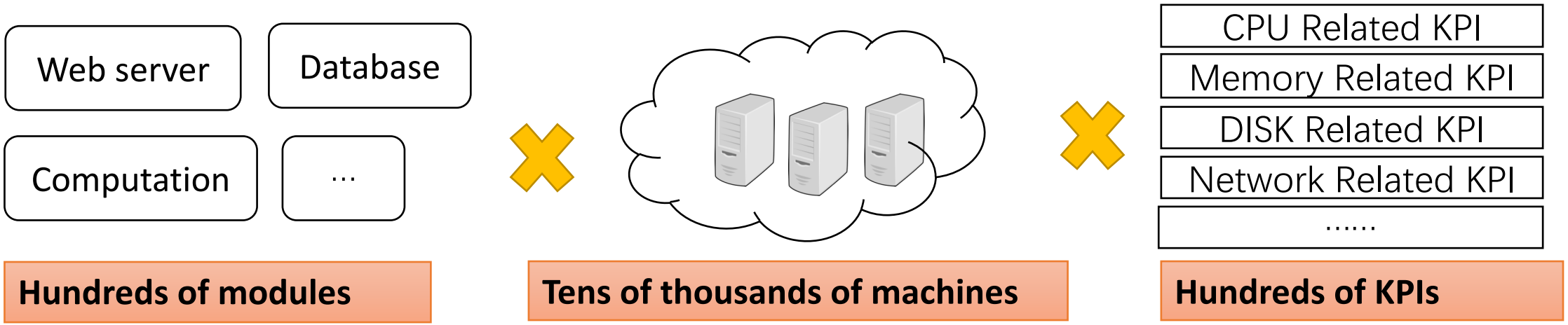
Mitigation

Why are operators reluctant to check the codes and exception logs?

operator

Operators mostly scan the KPIs to monitor the running status of modules and machines

Mitigation

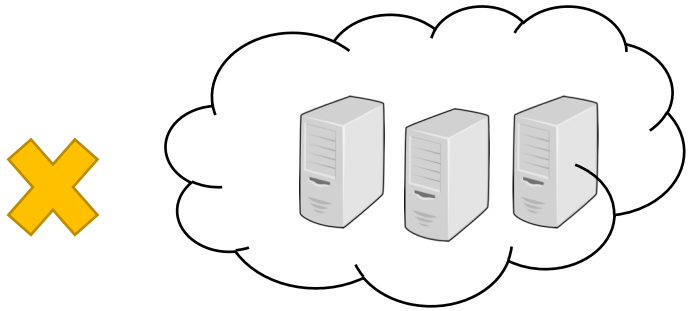


The search space is too huge !

Mitigation

Web server | Database | Computation | …

Hundreds of modules



Tens of thousands of machines

CPU Related KPI
Memory Related KPI
DISK Related KPI
Network Related KPI
……

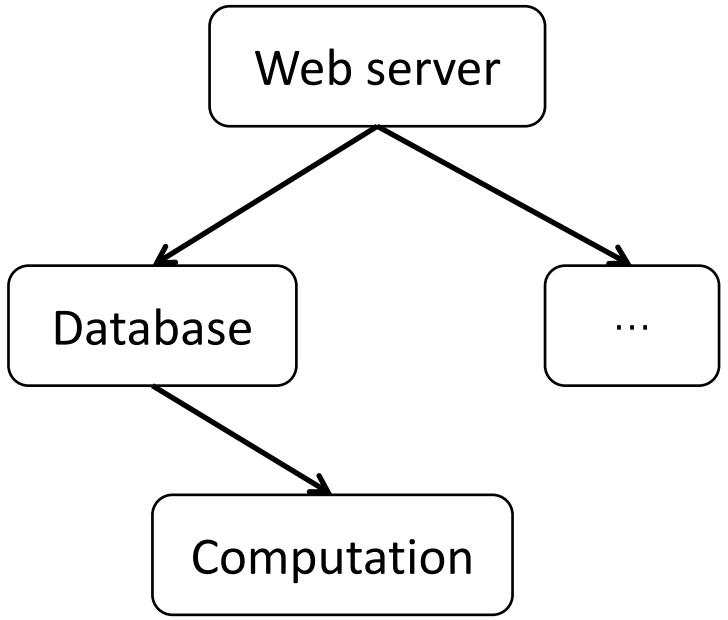
Hundreds of KPIs

I have to mitigate the failure quickly!

Mitigation

Root cause location can be localized along the dependency graph

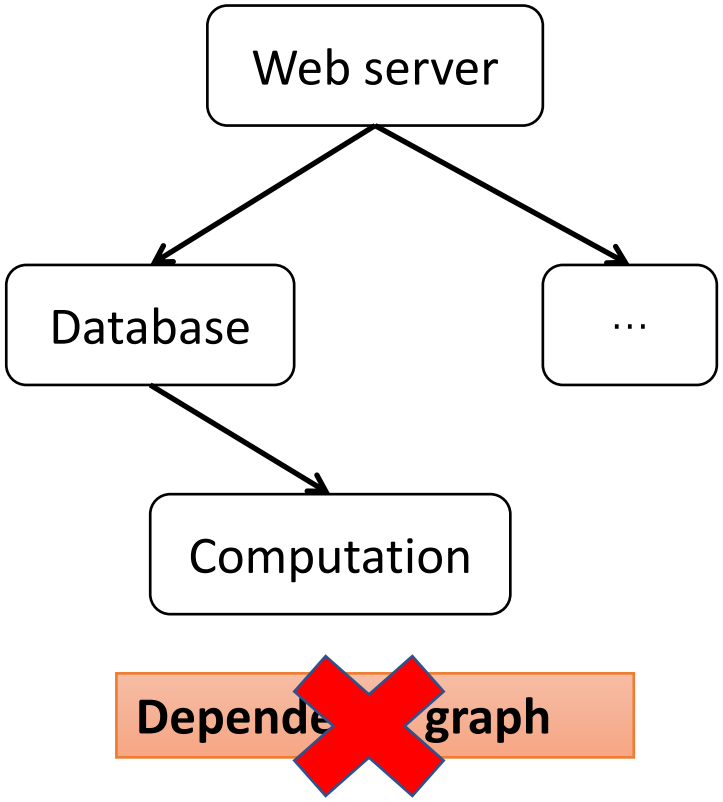
- Dependency graph based approaches
  - Sherlock [SIGCOMM' 07]
  - MonitorRank [SIGMETRICS' 13]
  - Fchain [ICDCS' 13]
  - CauseInfer [INFOCOM' 14]
  - BRCA [IPCCC' 16]



Dependency graph

- Dependency graph represents the dependencies between modules

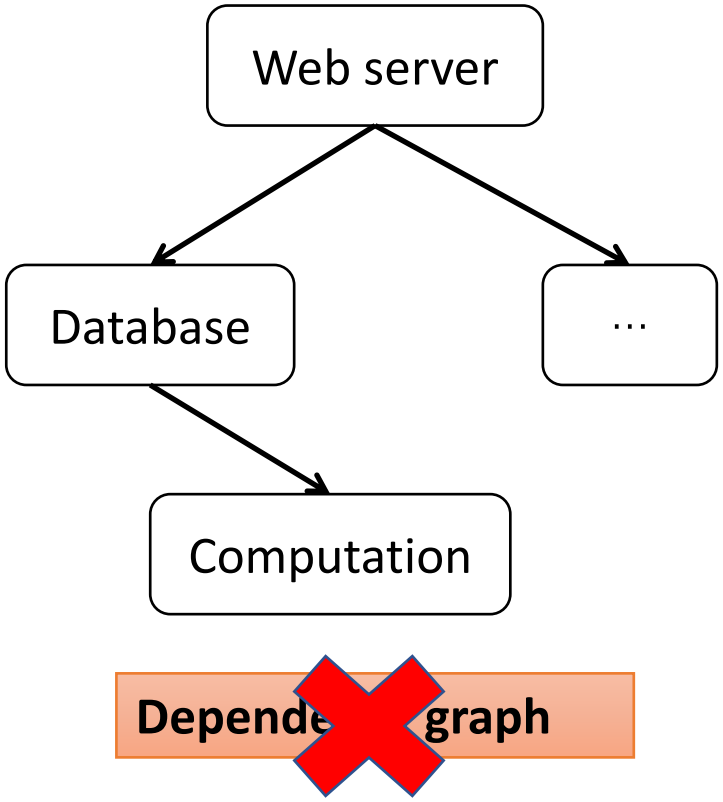
Mitigation



In practice, automatically obtaining the dependency graph of a online complex distributed service is difficult:

- Additional data collection codes need to be added, like Google's Dapper.
- For an online complex distributed service, it is infeasible.

Mitigation



The dependency graph also can be manually obtained by the experience of developers and operators:

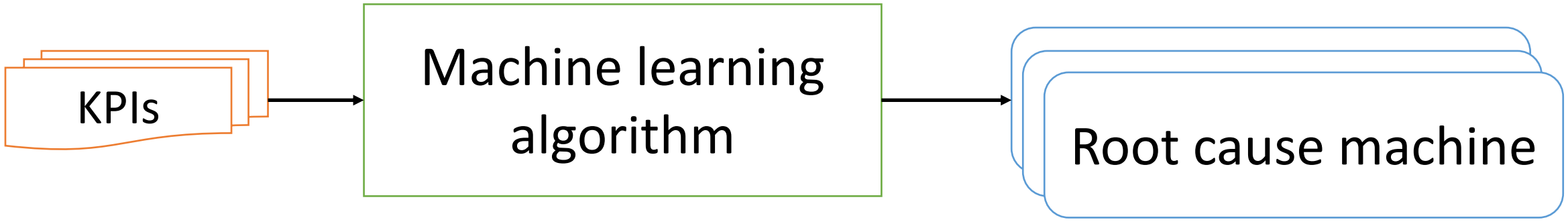
- Maintaining the graphs for the rapidly changing software services is difficult
- because the quick change of the codes makes the dependency graph elusive.

Mitigation

Therefore, in practice, the localizing process is still a manual process.

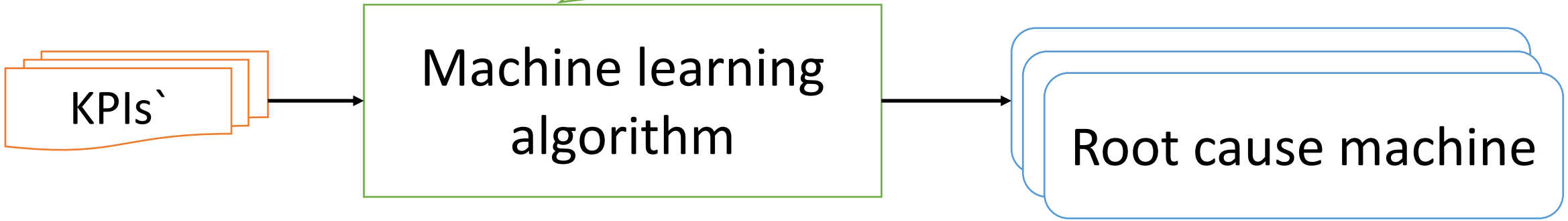
Core idea

If the manually scanning process can be automated by machine learning, then the overall mitigation time can be greatly reduced.



Core idea

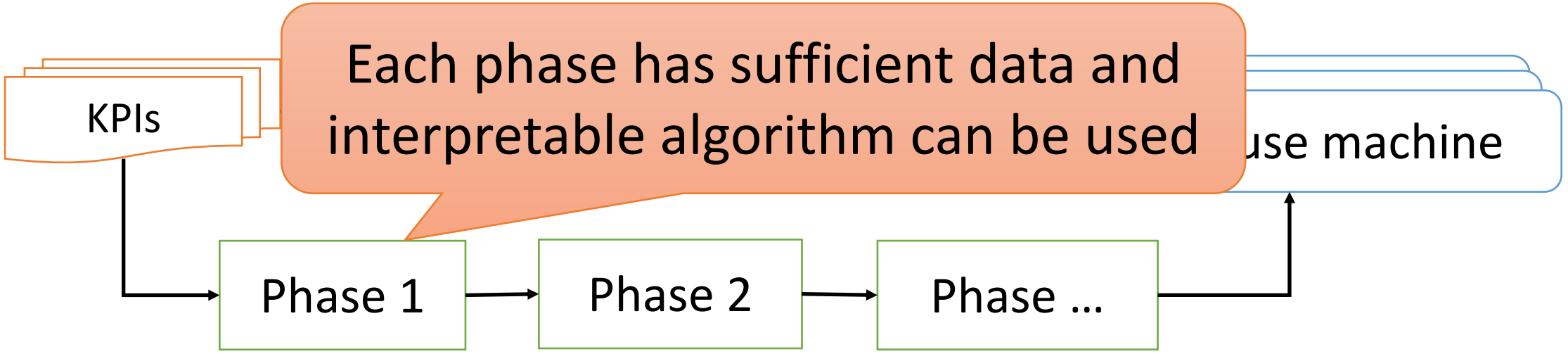
Directly training machine learning models in an end-to-end manner does not work



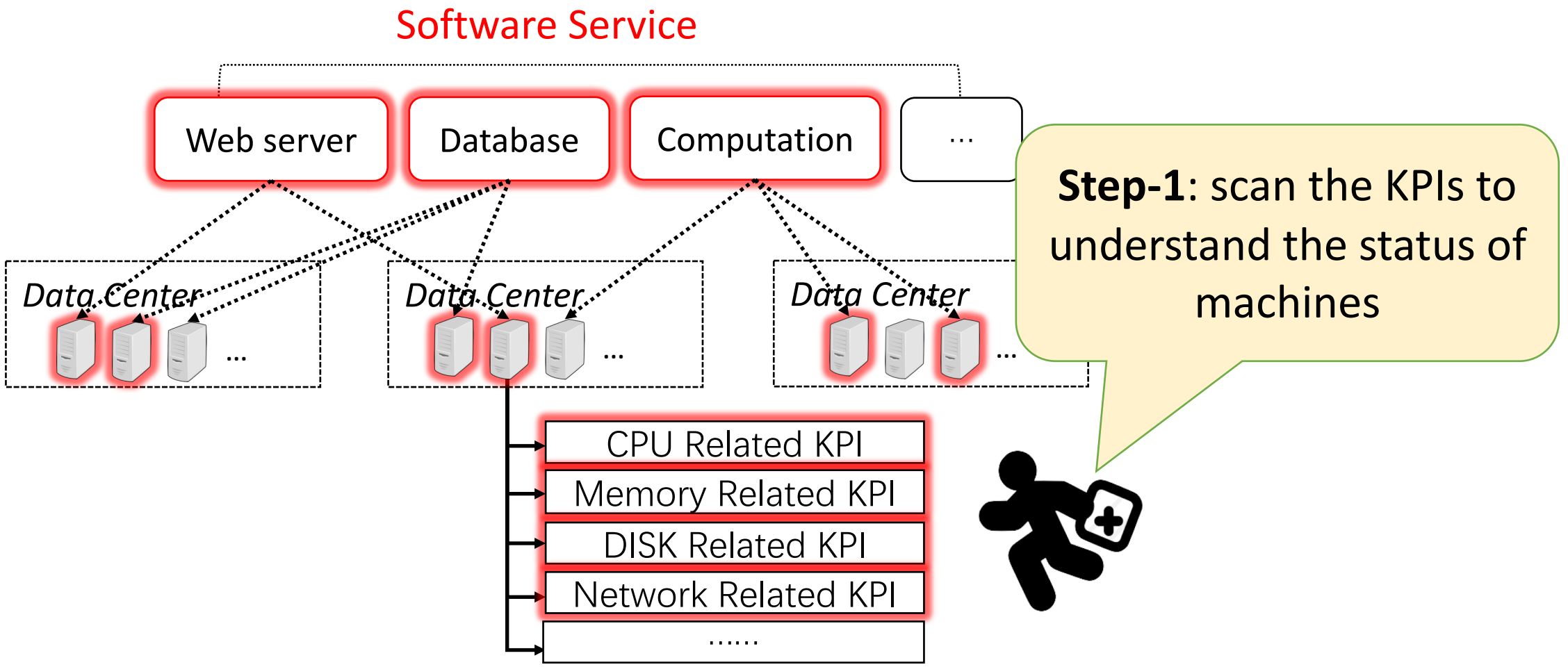
- Lack of interpretability.
- Insufficient failure cases.

Core idea

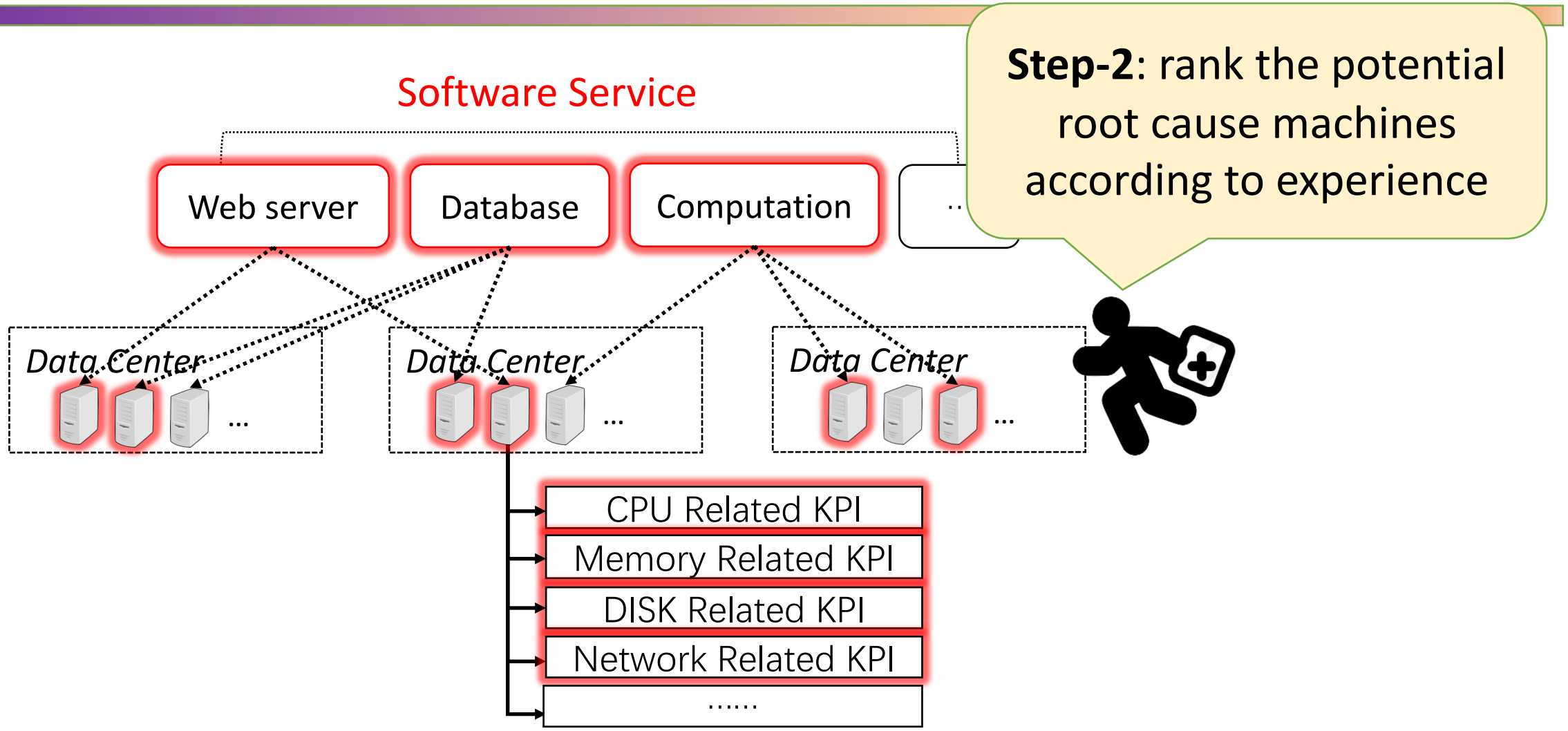
Domain-knowledge can be utilized to divide the problem into several phases



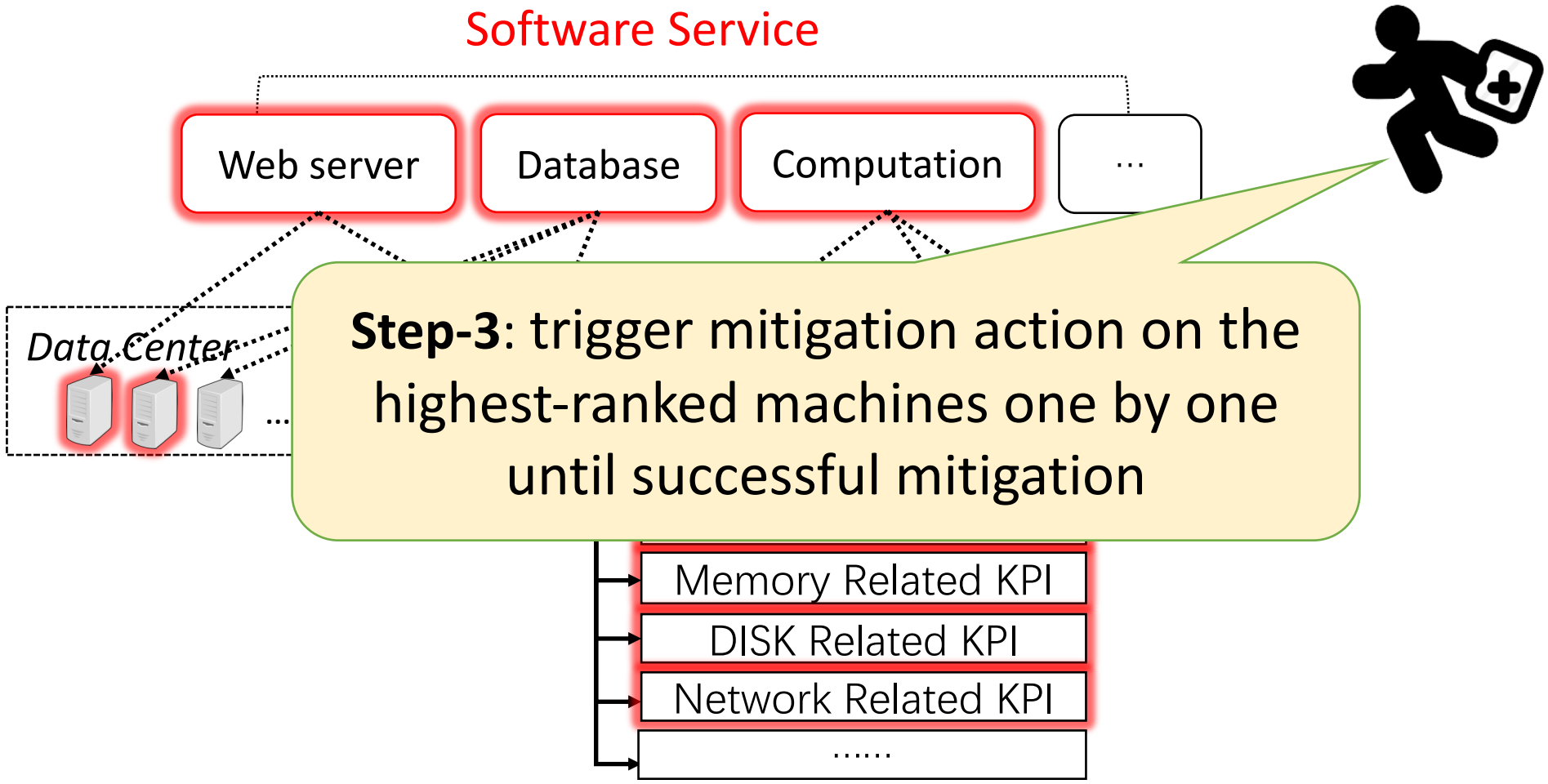
Manual localization without dependency graph



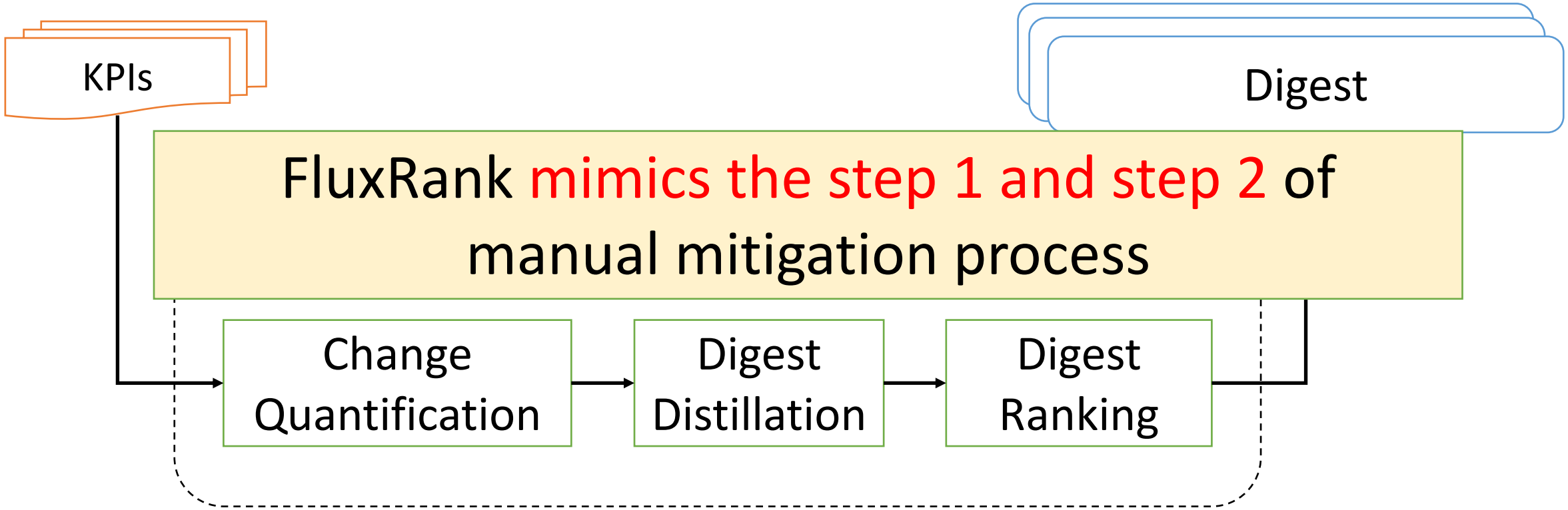
Manual localization without dependency graph



Manual localization without dependency graph



Core idea


KPIs
Digest
FluxRank mimics the step 1 and step 2 of manual mitigation process
Change Quantification
Digest Distillation
Digest Ranking

Background

Design

Evaluation

Case Study

Design

FLuxRank Distills valuable digest from the huge number of KPIs

KPIs

Digest

FluxRank

Change Quantification

Digest Distillation

Digest Ranking

Digest

FluxRank's output of a real failure case

Digest

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
	DC1_m2_1;		MEM_BUFFERS	21.9	4.6

A digest represents the **change patterns** of several machines from the same module.

Digest

Module name

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

Machines list

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

$$Ratio = \frac{\#\ of\ machines\ in\ the\ digest}{\#\ of\ machines\ in\ the\ module}$$

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

KPI list

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

Downward change score of KPI

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

upward change score of KPI

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

In the top digest, CPU idle KPIs dropped abnormally, and CPU load KPIs rose abnormally

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
	DC1_m2_1;		MEM_BUFF	21.9	4.6

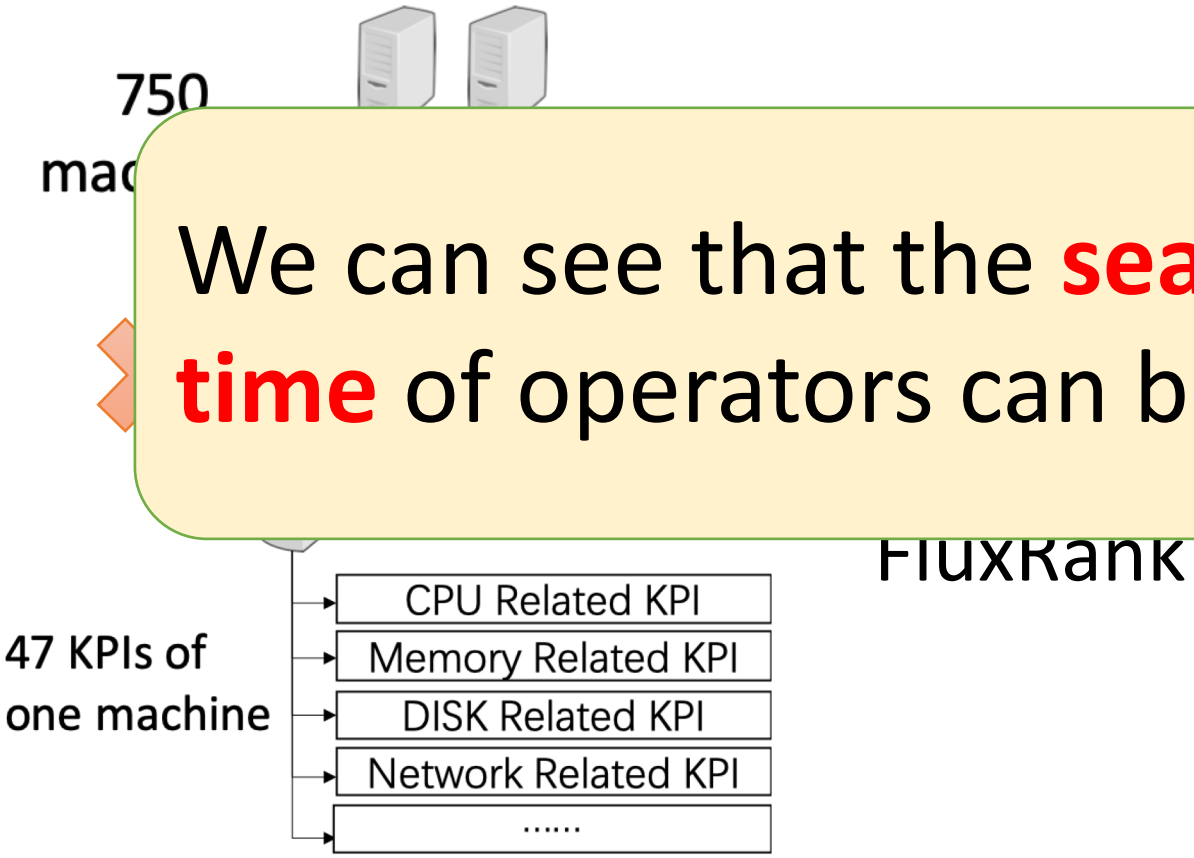
Operators can easily understand that **27** machines of module **M1** from **data center 1** have **CPU overload exception**

Digest

Module M1 is deployed on **750** machines, each machine has **47** standard Linux KPIs

Module	Ma		KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
M2	DC1_m2_1; ..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	MEM_BUFFERS	21.9	4.6
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

Digest

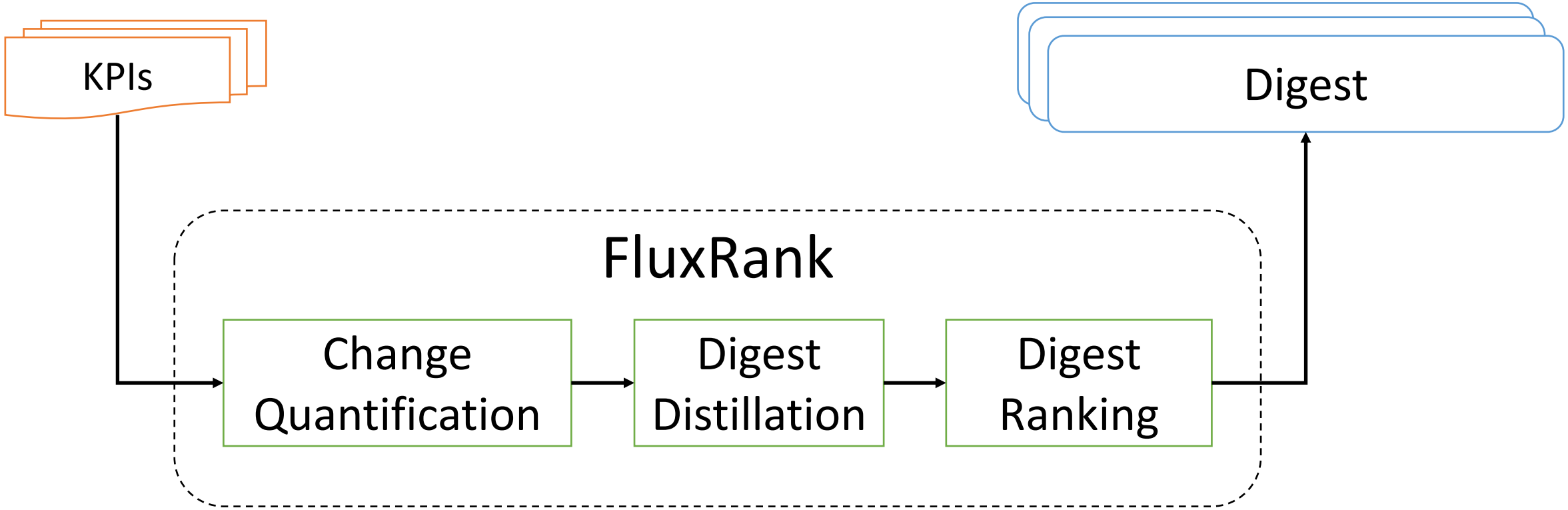


FluxRank

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M2	..... DC1_m2_31; DC2_m2_1 ..... DC2_m2_34;	0.21 (65/312)	CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
...	...	...	...	...	...

We can see that the **search space** and **analysis time** of operators can be **greatly reduced** !

Design


KPIs
Digest
FluxRank
Change Quantification
Digest Distillation
Digest Ranking

Change quantification

The change of each KPI is quantified into two change scores: upward change score (**o**) and downward change score (**u**)

Change quantification

Because the **quantified change scores will be used for ranking in phase three**, the scores have to satisfy the following requirement:

The change scores **are comparable** among diversified KPI characteristics

⇨ Probability

Change quantification

The change quantification also must be **lightweight** , because **hundreds of thousands of KPIs** need to be quickly quantified

Change quantification

Probability

+

lightweight

+

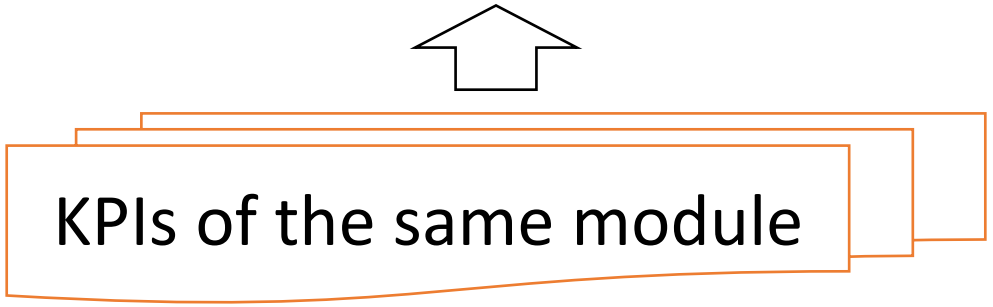
Comparable between diversified KPI characteristics

→ Kernel Density Estimation (KDE) based quantification algorithm

→ In KDE, Choose different kernels for different KPIs

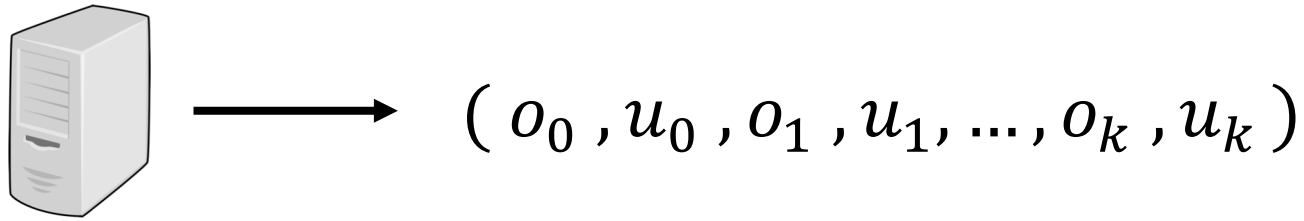
Digest distillation

Construct vectors representation of the change pattern of machines



Digest distillation

- Suppose each machine has k KPIs, then the KPIs upward change score o and downward change score u can form a vector to represent the change pattern of the machine



Digest distillation

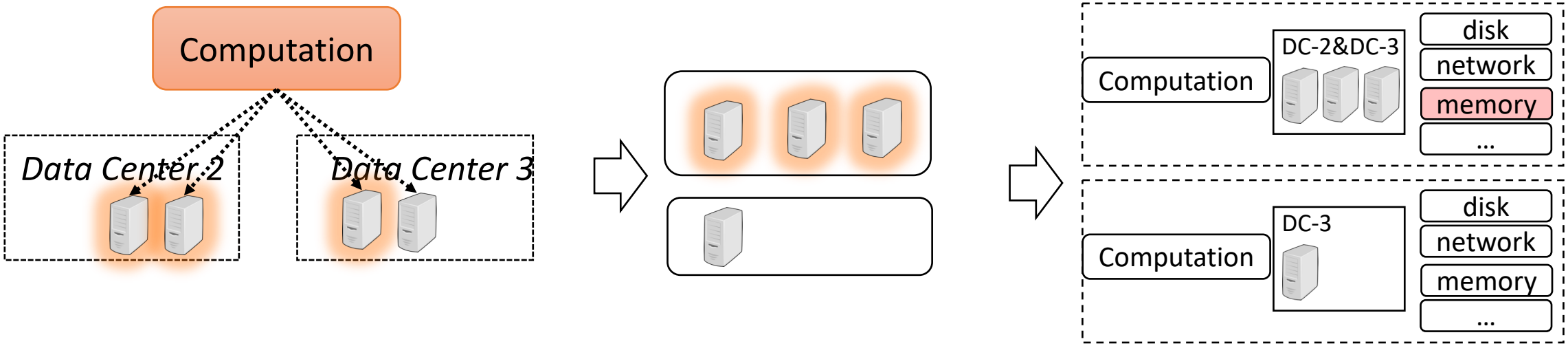
Use constructed vectors to cluster machines to generate digests

Construct a vector representation of the change pattern of a machine

KPIs of the same module

Digest distillation

Change Quantification → **Digest Distillation** → Digest Ranking



Digest distillation

We choose **DB-SCAN** as the clustering algorithm, because the cluster number can not be determined

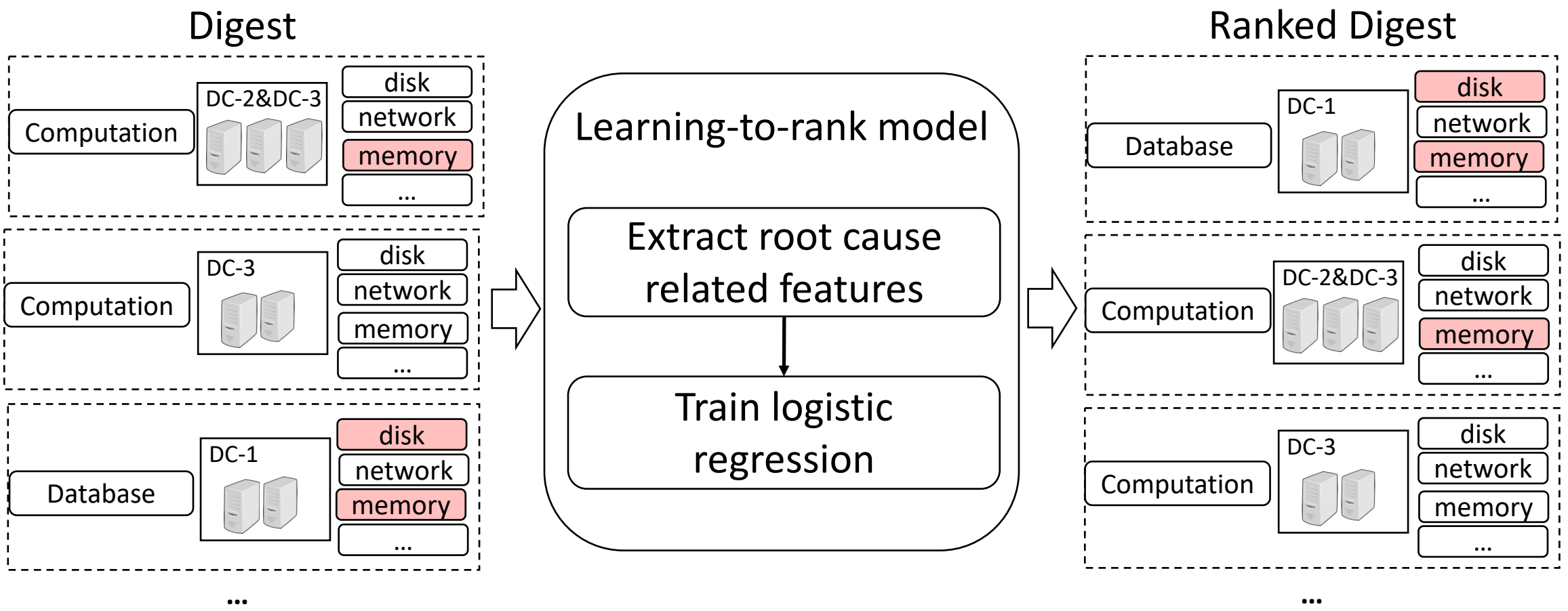
We use **Pearson correlation** as the distance function of clustering algorithm, which can capture the similar change pattern

Digest Ranking

The distilled digests need to be ranked so that the one **most relevant to the root cause** can be listed at the top

Digest Ranking

Change Quantification → Digest Distillation → Digest Ranking



Background

Design

Evaluation

Case Study

Datasets

The largest system contains **11519** machines

TABLE II: Fail… …om five real production systems

Service	#Modules	#Machines	Description	#Case
p1	29	11,519	an application system for desktop clients that handles billions of user requests per day	10
p2	17	2,147	an application system for mobile clients that handles billions of user requests per day	48
p3	91	5,747	a monitoring system A for the whole company	7
p4	85	3,872	a financial service system like paypal	1
p5	7	238	a monitoring system B for the whole company	4

Our dataset contains **70 real failures cases** from **five different online** software services

Datasets

TABLE I: The 47 types of machine KPIs

Type (#number)	KPI
CPU-Related (8)	CPU_IDLE; CPU_HT_IDLE; CPU_CONTEXT_SWITCH; CPU_INTERRUPT; CPU_SERVER_LOADAVG_1; CPU_SERVER_LOADAVG_15; CPU_SERVER_LOADAVG_5; CPU_WAIT_IO.
Disk-Related (15)	DISK_TOTAL_USED_PERCENT; FD_USED_PERCENT; DISK_TOTAL_INODE_USED_PERCENT; DISK_FS_ERROR; FD_USED; DISK_PAGE_IN; DISK_PAGE_OUT; DISK_TOTAL_AVG_WAIT; DISK_TOTAL_IO_UTIL; DISK_TOTAL_READ_KB; DISK_TOTAL_READ_REQ; DISK_TOTAL_READ_AVG_WAIT; DISK_TOTAL_WRITE_AVG_WAIT; DISK_TOTAL_WRITE_KB; DISK_TOTAL_WRITE_REQ.
Memory-Related (6)	MEM_USED_PERCENT; MEM_USED_ADD_SHMEM_PERCENT; MEM_BUFFERS; MEM_CACHED; MEM_USED; MEM_USED_ADD_SHMEM.
Network-Related (13)	NET_MAX_NIC_INOUT_PERCENT; NET_TCP_IN_ERRS; NET_TCP_RETRANS; NET_TCP_LOSS; NET_UP_NIC_NUMBER; NET_TCP_ACTIVE_OPENS; NET_TCP_CURR_ESTAB; NET_TCP_IN_SEGS; NET_TCP_OUT_SEGS; NET_TCP_TIME_WAIT; NET_TOTAL_IN_BITPS; NET_TOTAL_OUT_BITPS; NET_TOTAL_SOCKETS_USED.
OS kernel-Related (5)	SYS_OOM; SYS_PAGING_PROCS; SYS_RUNNING_PROCS; SYS_STOPPED_PROCS; SYS_ZOMBIE_PROCS.

Metric

- **Root cause digest (RCD)**. A root cause digest is a digest satisfying the following conditions:
  - All machines of a digest are root cause machines where the root cause took place.
  - The top-five KPIs of a digest contain one or more root cause relevant KPIs

Root Cause Digest

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5		25.5
			CPU_SERVER_LOADAVG_15	0.2	
			NET_TCP_OUT_SEGS		
			NET_TCP_IN_SEGS		
			MEM_CACHED		
			...		
		0.21 (65/312)	MEM_BUFFERS		
			CPU_SERVER_LOADAVG_15	0.36	9.0
			DISK_TOTAL_READ_REQ	0.45	8.6
			...	...	...
		...	...	...	...

All 27 machines are root cause machines where CPU overload took place

Top five KPIs are root cause related KPIs which can indicate CPU overload

Metric

- We use Recall@K as the evaluation metric

$$\textbf{Recall@K} = \frac{\#\text{ of cases whose top-k } \textbf{digests} \text{ contain } \textbf{RCDs}}{\#\text{ of all cases}}$$

How many cases' root cause digest can be ranked into top K

Offline Evaluation

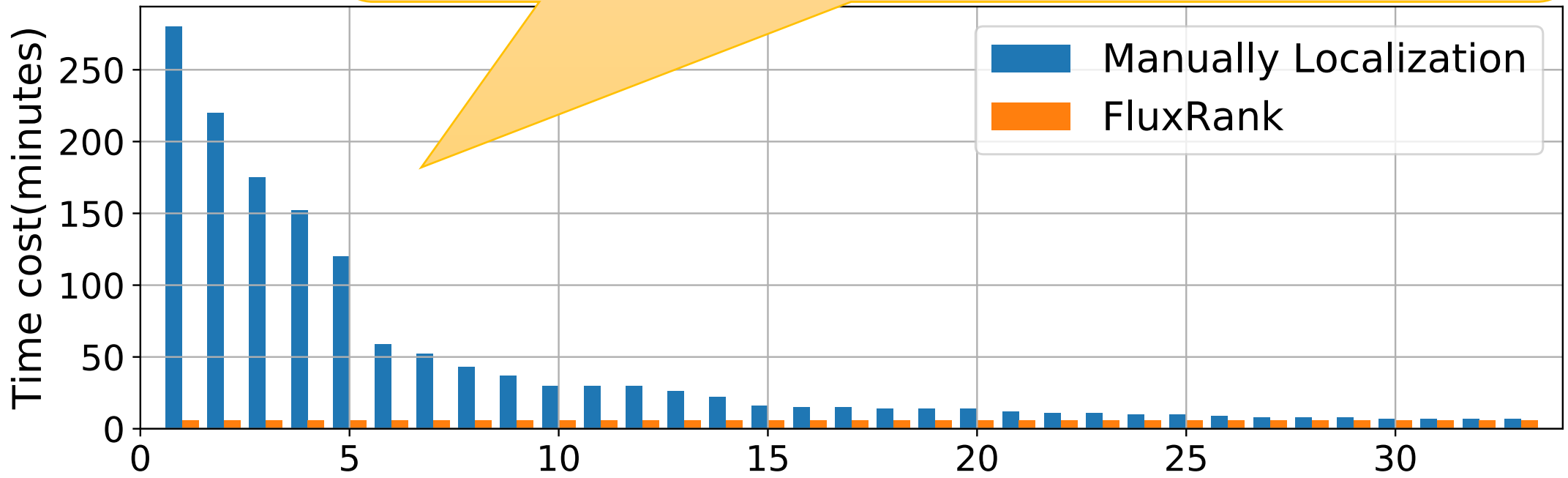
FluxRank is stable among different folds of cross-validations.

Model	Recall@1	Recall@2	Recall@3
FluxRank(5-fold)	0.78(55/70)	0.89(62/70)	0.94(66/70)
FluxRank(3-fold)	0.78(55/70)	0.9(63/70)	0.94(66/70)
FluxRank(2-fold)	0.85(60/70)	0.89(62/70)	0.94(66/70)

66/70 cases' root cause digests are ranked into top 3.

Offline Evaluation

Compared with manual localization, FluxRank cuts the mitigation time by more than 80% on average



Online Evaluation

- FluxRank has been **successfully deployed** online on **one Internet service** (with hundreds of machines) and **six banking services** (each with tens of machines) in two large banks for three months.

TABLE V: The details of 59 online cases from 7 real services. The valid case represents the case whose RCD is ranked first.

	#module	#machine	#KPI/machine	(#valid case) / (#total case)
s1	15	520	591	2/2
s2	3	30	120	1/1
s3	4	40	302	3/3
s4	4	38	520	3/5
s5	3	35	424	1/1
s6	4	38	512	3/5
s7	7	26	311	42/42

55/59 cases' root cause digests are ranked into top 1

Background

Design

Evaluation

Case Study

Case study

- This case is a CPU overload failure. The failure service contains 29 modules and runs on 11,519 machines.

- The root cause of this failure is that 27 machines causes CPU overload exception.

Case study

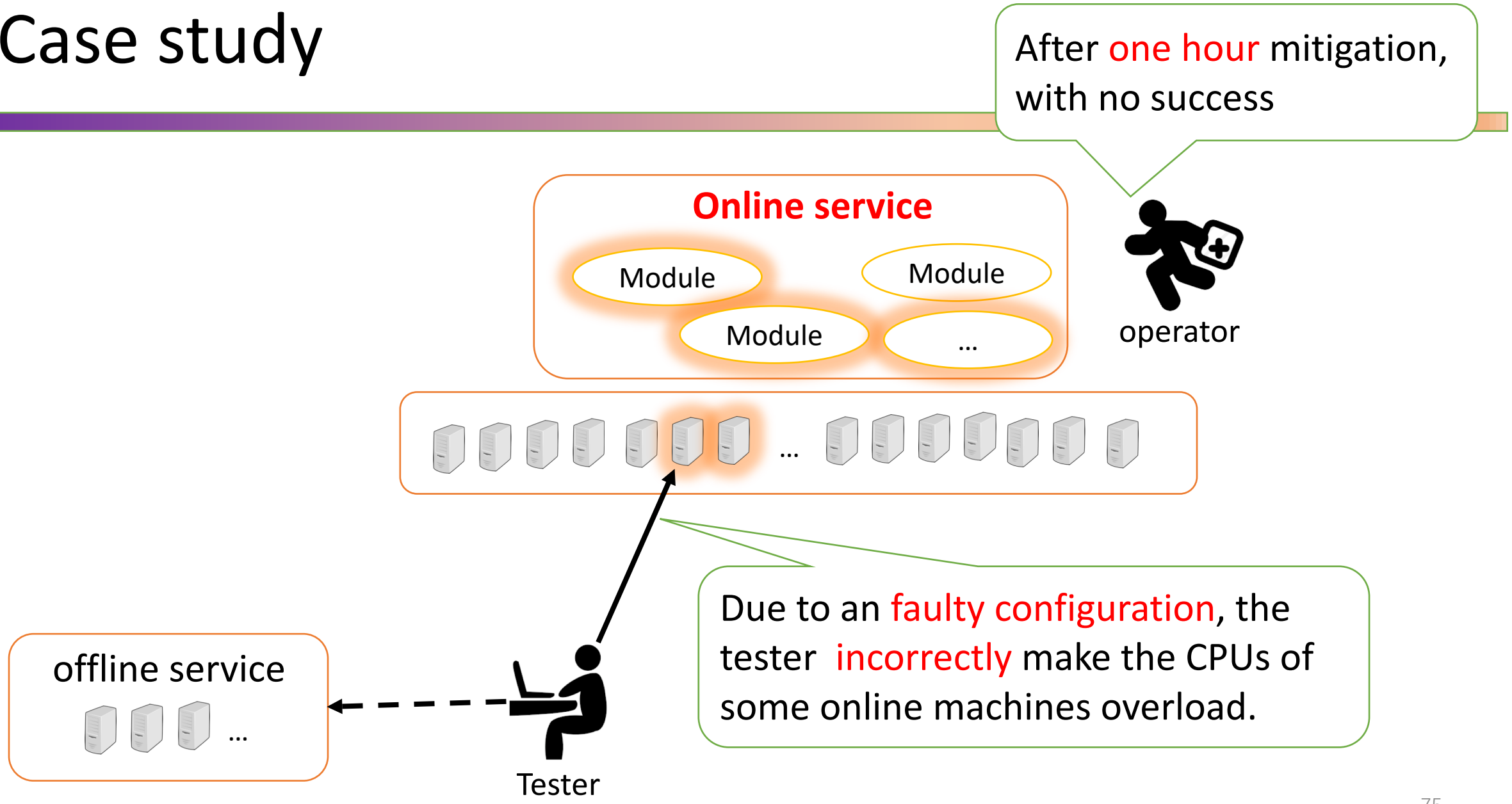
offline service

...

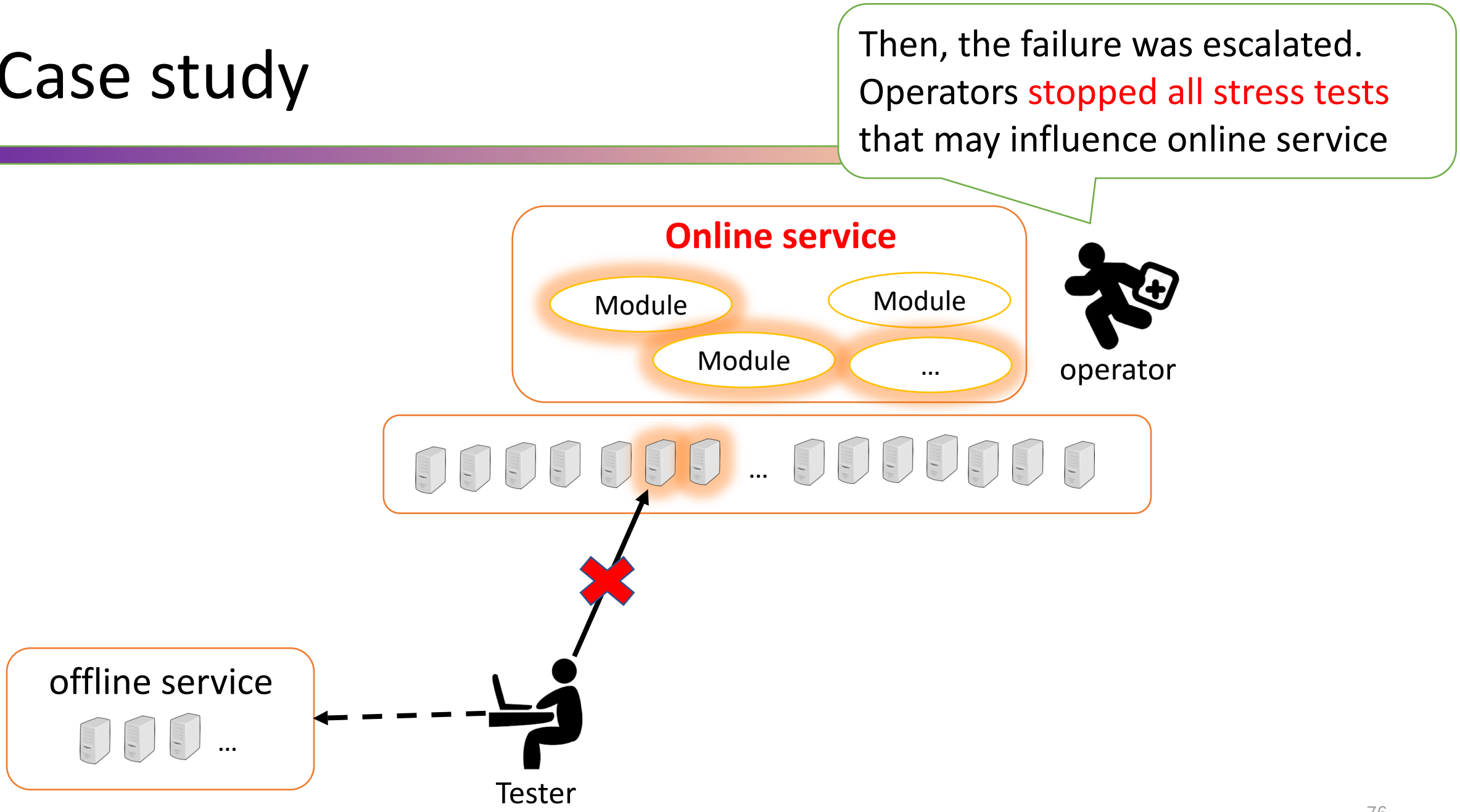
Tester

Start offline CPU stress test on the offline service

Case study

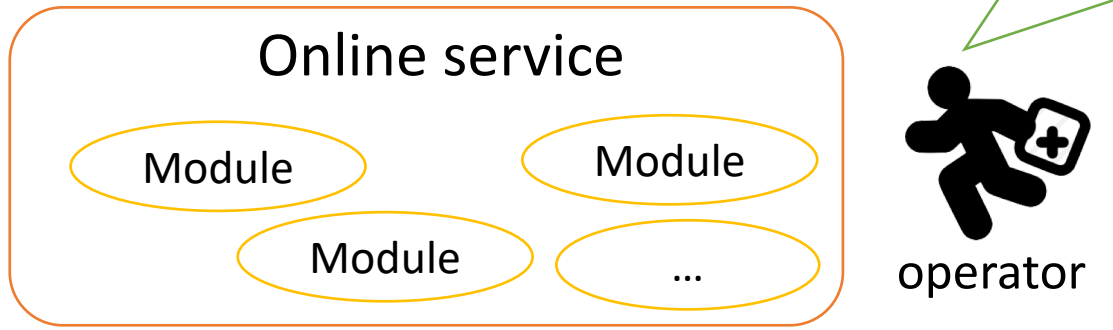

After one hour mitigation, with no success
Online service
Module
Module
Module
...
operator
...
offline service
...
Tester
Due to an faulty configuration, the tester incorrectly make the CPUs of some online machines overload.

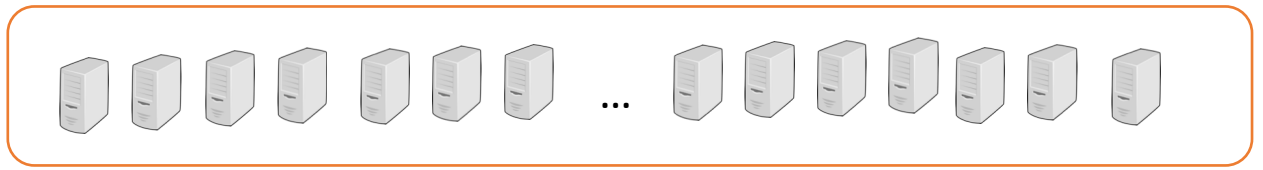
Case study


Then, the failure was escalated.
Operators stopped all stress tests that may influence online service
Online service
Module
Module
Module
...
operator
...
offline service
...
Tester

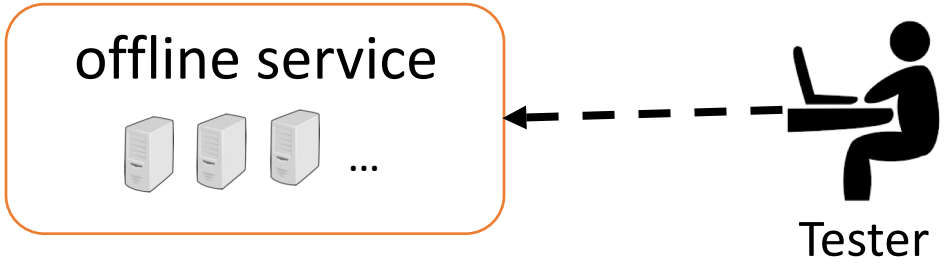
Case study

Eventually, they successfully mitigated the failure, but spent about two hours in total.





operator



Case study

The CPU related KPIs also be ranked to the top of digest's KPI list

FluxRank successfully recommended the 27 CPU overloaded machines to the top

Module	Machine	Ratio	KPI	Downward Change (u)	Upward Change (o)
M1	DC1_m1_1 ; DC1_m1_2; ..... DC1_m1_27;	0.036 (27/750)	CPU_HT_IDLE	45.3	0.0
			CPU_IDLE	34.6	0.0
			CPU_SERVER_LOADAVG_1	0.1	28.3
			CPU_SERVER_LOADAVG_5	0.1	25.5
			CPU_SERVER_LOADAVG_15	0.2	24.3
			NET_TCP_OUT_SEGS	0	22.5
			NET_TCP_IN_SEGS	0	21.4
			MEM_CACHED	6.5	1.8
			...	...	...
					4.6
					9.0
					8.6
					...
					...

Operators can easily understand that **27** machines of module **M1** from **data center 1** have **CPU overload exception.**

Conclusion

- Target
  - **Failure mitigation.**
  - **Not find exact root cause**
- Method
  - **Quantify KPIs -> Cluster machines -> Rank digests**
  - **Not use dependency graph**
- Offline evaluation
  - **66/70** cases' root cause digests are ranked **into top 3**
- Online evaluation
  - **55/59** cases' root cause digests are ranked **into top 1**

Thank you!

Q&A

liuping15@mails.tsinghua.edu.cn

ISSRE 2019

Widely-deployable Framework

- FluxRank can be easily deployed using existing KPI data without any change of the service.

- FluxRank have been quickly deployed on six online service.

Diversified KPI characteristics

- CPU idle KPI
  - **Ratio** KPI
  - Value range: [0, 1.0]
  - It is **anomalous** when the value is **close to 0 or 1**
  - **Beta distribution** is more suitable to describe ratio KPI
- Example
  - If normal range is: [0.3, 0.8]
  - During CPU overload, the value is: 0.1
  - From **Gaussian** distribution, 0.1 is **normal**
  - From **Beta** distribution, 0.1 is **anomalous**