An Intelligent Framework for Timely, Accurate, and Comprehensive Cloud Incident Detection

Yichen Li[1*], Xu Zhang[2], Shilin He[2], Zhuangbin Chen[1], Yu Kang[2**], Jinyang Liu[1], Liqun Li[2], Yingnong Dang[3], Feng Gao[3], Zhangwei Xu[3], Saravan Rajmohan[4], Qingwei Lin[2**], Dongmei Zhang[2], Michael R. Lyu[1]
[1]The Chinese University of Hong Kong, Hong Kong, China, {ycli21, zbchen, jyliu, lyu}@cse.cuhk.edu.hk
[2]Microsoft Research, Beijing 100080, China, {xuzhang2, shilhe, kay, liqli, qlin, dongmeiz}@microsoft.com
[3]Microsoft Azure, Redmond, WA 98052, USA, {yidang, fgao, zhangxu}@microsoft.com
[4]Microsoft 365, Redmond, WA 98052, USA, saravar@microsoft.com

Abstract

Cloud incidents (service interruptions or performance degradation) dramatically degrade the reliability of large-scale cloud systems, causing customer dissatisfaction and revenue loss. With years of efforts, cloud providers are able to solve most incidents automatically and rapidly. The secret of this ability is intelligent incident detection. Only when incidents are detected timely, accurately, and comprehensively, can they be diagnosed and mitigated at a satisfiable speed. To overcome the limitations of traditional rule-based detection, we carried out years of incident detection research. We developed a comprehensive AIOps (Artificial Intelligence for IT Operations) framework for incident detection containing a set of data-driven methods. This paper shares our recent experience of developing and deploying such an intelligent incident detection system at Microsoft. We first discuss the real-world challenges of incident detection that constitute the pain points of engineers. Then, we summarize our intelligent solutions proposed in recent years to tackle these challenges. Finally, we show the deployment of the incident detection AIOps framework and demonstrate its practical benefits conveyed to Microsoft cloud services with real cases.

1 Introduction

In recent years, although enormous efforts have been devoted to maintaining the reliability of cloud systems, incidents at different severity levels are still inevitable [2–4]. Severe incidents will incur poor user experience and huge finical loss [10, 12], which are required to be detected and solved as soon as possible. In practice, incident detection of online services heavily depends on the services' observability data, e.g., logs, metrics, traces, and events. Based on these data, a large number of alerting rules are manually configured by engineers to detect incidents. Once an incident is detected, responsible On-Call Engineers (OCEs) will immediately start the investigation to quickly restore and fix the service.

However, due to complex system architecture and dependencies among service components, it is impossible to manually design complicated and diverse incident detection mechanisms to cover all types of failures, which are often error-prone and unscalable. During the past several years, we have continuously studied the problem of incident detection in large-scale cloud systems. Given that incident detection is data-driven by nature, we tackle this problem in a fashion of AIOps (Artificial Intelligence for IT Operations) [8, 11], which leverages the power of AI to pursue intelligent incident detection. Compared to manual labor, AIOps-powered incident detection enables incidents to be diagnosed and mitigated promptly, accelerating the incident management process. Moreover, due to data security and integrity, manual intervention from public support may not be allowed in private clouds, sovereign clouds [16], and other sensitive clouds [1]. These all advocate the adoption of automated decision making, especially by AI-based solutions. By proposing a series of intelligent approaches [5–7, 9, 13–15, 18, 20, 24, 26–29], we have developed a comprehensive AIOps framework for prompt and accurate incident detection. It has been deployed within Microsoft and serving many services for years, producing high impact on industrial practices.

Throughout years of applying AIOps strategies to detecting incidents, we have gained rich experience, which is valuable for us and the broad community of cloud IT operations to carry out further customization and improvement. This paper shares our experience of conducting this line of research at Microsoft. Specifically, we first introduce the pain points and challenges of incident detection, i.e., *complex alerting logic*, *flooding incidents*, *selective incident enrichment*, and *reactive alerts*. Then, we elaborate on the developed techniques to tackle the aforementioned challenges, which can be categorized into four classes, i.e., *multi-aspect detection*, *incident refinement*, *incident enrichment*, and *proactive detection*. Finally, we report the deployment of the AIOps framework in Microsoft. Real-world incident cases are also shared to present the benefits achieved by our AIOps framework.

To sum up, this work makes the following major contributions:

- We study the problem of incident detection in large-scale production cloud systems. We have summarized the pain

*Work done during internship at Microsoft Research Asia.
**Yu Kang and Qingwei Lin are the corresponding authors.

points and challenges in the real-world detection workflow for critical incidents.
- An AIOps framework is developed to tackle the key challenges, aiming at promoting the timely, accurate, and comprehensive intelligent incident detection.
- We show deployment of the AIOps framework in production along with real-world cases to demonstrate the industrial benefits conveyed to the incident detection of Microsoft.

2 Background

2.1 Cloud Incidents

In cloud systems, an incident is an unplanned interruption or performance degradation of a service or product, such as API timeouts, network jitters, and power outages. Incidents have different severity levels - low, medium, high, and critical, which are set according to the potential impact on customers. Besides severity level, incidents also contain information like the reporting service, location, related logs or customer support tickets. These information plays an essential role in incident resolving.

2.2 Incident Detection

Incident detection is the process of identifying service interruption or performance/quality issues and rendering an incident ticket to record relevant information. For a cloud system with complex dependencies among thousands of components (e.g., microservices, serverless functions), incidents are inevitable. Therefore, timely, accurate, and comprehensive incident detection is the key to reducing service downtime, which serves as the primary step for the follow-up incident management tasks, e.g., incident triage [14] and incident mitigation [23]. Incidents are mainly detected by monitors, which continuously run programs to examine the health status of a service component. Engineers can also submit incidents manually if they observe abnormal system behaviors or confirm customer-reported failure messages. To pursue accurate incident detection, different types of observability data (e.g., logs, metrics, traces, events) are involved, which characterize the status and behavior of a component from different perspectives. Based on these data, engineers need to design appropriate alerting logic for monitors to capture incidents comprehensively. Typical configurations of monitors include determining the detection scope (e.g., cluster, node or software version) with the corresponding measurements (e.g., availability, success rate, latency, throughput), setting alerting thresholds or designing automated detection methods applied on them.

3 Challenges

A typical cloud system is characterized by its enormously large scale beyond general software systems and complicated dependency among different components. Therefore, it is very challenging to identify incidents from such a cloud infrastructure and the diverse services that are built on top of it. In this section, we summarize the real-world problems of incident detection.

Version	Cluster	Node	...	API	Return State	Latency
V_1.1	PrdC01	N01	...	GET-FILES	Success	23ms
V_1.1	PrdC01	N01		POST-RESET	Success	11ms
...			...			
V_1.3	PrdC03	N05		GET-FILES	Failed	58ms
V_1.3	PrdC04	N06		POST-RESET	Success	17ms
V_1.4	PrdC04	N06		GET-PWD	Failed	31ms
...			...			

Table 1. An example of monitoring data in cloud.

3.1 Complex Alerting Logic

When configuring incident detectors for a single service, engineers often suffer from determining the complicated alerting logic to cover various aspects of the service. Ideally, these aspects should reflect the system health status, including different *monitoring metrics* (e.g., availability, request success rate, request latency) upon diverse *monitored subjects* (e.g., clusters, nodes, or software versions). An example is shown in Table 1. On the one hand, the number of detectors would be explosive due to the exponential combinations among these monitored subjects (e.g., detecting incidents based on the request success rates with respect to API "GET-FILES" on Cluster PrdC01.). For compromise, only a few coarse-grained detectors can be set up leaving finer-grained issues often missed. On the other hand, the incident detection models are individually built up based on heterogeneous monitoring metrics. However, the commonalities shared among these metrics (e.g., the similar seasonality pattern across multiple metrics) are not considered, leading to much repetitive work for detection model training and deployment. Moreover, the barriers among the detection approaches are not conducive to improving their detection accuracy through knowledge sharing.

3.2 Flooding Incidents

There are numerous incidents created in the cloud systems, making engineers exhausted from tedious on-call matters. Specifically, service teams in modern cloud systems usually adopt the DevOps practice to implement and maintain their services individually, where each service team tends to report its own incidents. In practice, services in such a large-scale system are never isolated and the failure of one service often propagates to dependent services, leading to related failures. As the real-world example in Figure 1 shows, a configuration synchronization issue of the Network service leads to the failed access of the Storage service, which further causes the SQL database and other storage-dependent service failures.

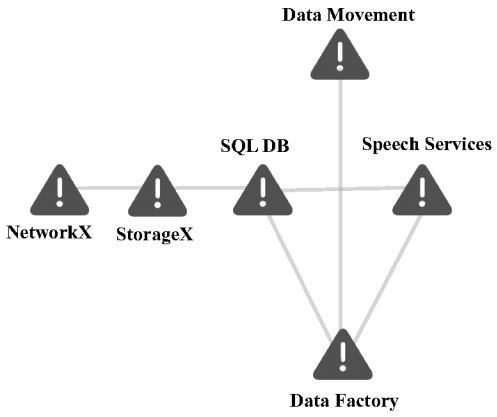

Figure 1. An incident example that affects multiple services.

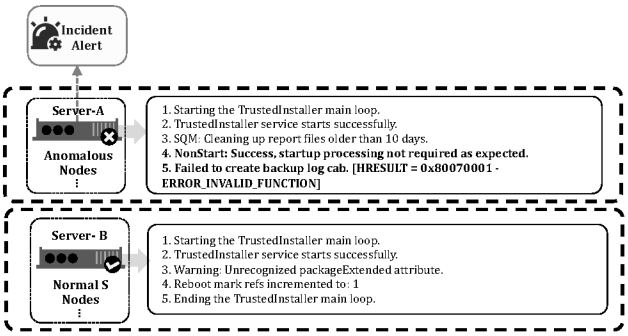

Figure 2. An incident example with relevant service-side console logs (shown in bold).

In this case, multiple incidents are created separately but by nature should be jointly resolved. However, from the view of a single service, it is hard to know whether it affects or is affected by other services without in-detail investigation, causing the waste of engineering efforts and incomplete incident analysis. In addition, engineering resources are often limited while many incidents occur simultaneously. Some of them are more impactful and should be fixed with a higher priority. However, it is non-trivial to decide the priority without domain expertise.

3.3 Selective Incident Enrichment

Most incidents are detected through monitoring metrics, such as a surge of latency or a significant drop on request success rate. However, without further enrichment, they can only show the symptoms of failures, which are far from enough to dig out their root causes or estimate their impact on users. It is necessary to enrich incidents by resorting to extra information sources, such as the console logs from the internal service side and the support tickets from the external customer side. Unfortunately, there exist an extremely large volume of noisy raw log messages and irrelevant customer support tickets co-occurring with the detected incidents. For the example in Figure 2, a large number of continual rebooting nodes triggered an incident due to the incorrect configuration file update. The key log messages, such as "Update networkConfig.json, NodeId <*>", are able to pin point to its root cause directly but overwhelmed by a mass of unrelated log messages. Moreover, the critical information tends to be presented in the form of text data in logs and customer support tickets, which are not well schematized for automated extraction. It is inefficient to correlate all logs or support tickets with the target incident and labor intensive to investigate. As a result, engineers may selectively enrich the incidents based on their knowledge. This could lead to missing information due to limited human knowledge or rapid change of service.

3.4 Reactive Alerts

As mentioned in Section 2.2, current incident detection usually relies on configured alerting logic based on observability data. While the detection is getting smarter (e.g., dynamic thresholding), most monitors are still in a reactive manner. That is to say, such monitors can only detect incidents after the failures and anomalies have appeared. No matter how accurate and rapid the detection is, the incidents have already happened. I.e., the detection lags behind the service failure happens. To avoid the failures from happening, we need the ability to foresee the incidents and take actions beforehand. In this way, we can proactively engage engineers preventing service failures, instead of reactively responding to incidents.

4 Approaches

4.1 Overview

To resolve the challenges listed in Section 3, we present a general and comprehensive framework for intelligent cloud incident detection as shown in Figure 3. The goal is to achieve timely, accurate, and comprehensive incident detection with intelligent methods. The overall incident detection framework is built on top of various observability signals (i.e., metric, log, and trace) and customer reported issues while also incorporating domain knowledge such as the cloud system architecture, network topology, as well as service static and dynamic dependencies.

In general, there are four building blocks in the framework, i.e., *multi-aspect detection*, *proactive detection*, *incident refinement*, and *incident enrichment*. We start with the *multi-aspect detection*. For a service that requires complex alerting logic, we aim to achieve more efficient and accurate configurations of detection scopes and metrics. For some types of failures, there are some early signals that could indicate the occurrence of incidents ahead of time. Therefore, in *proactive detection*, we demonstrate our attempts to proactively predict the hardware and software failures with various signals

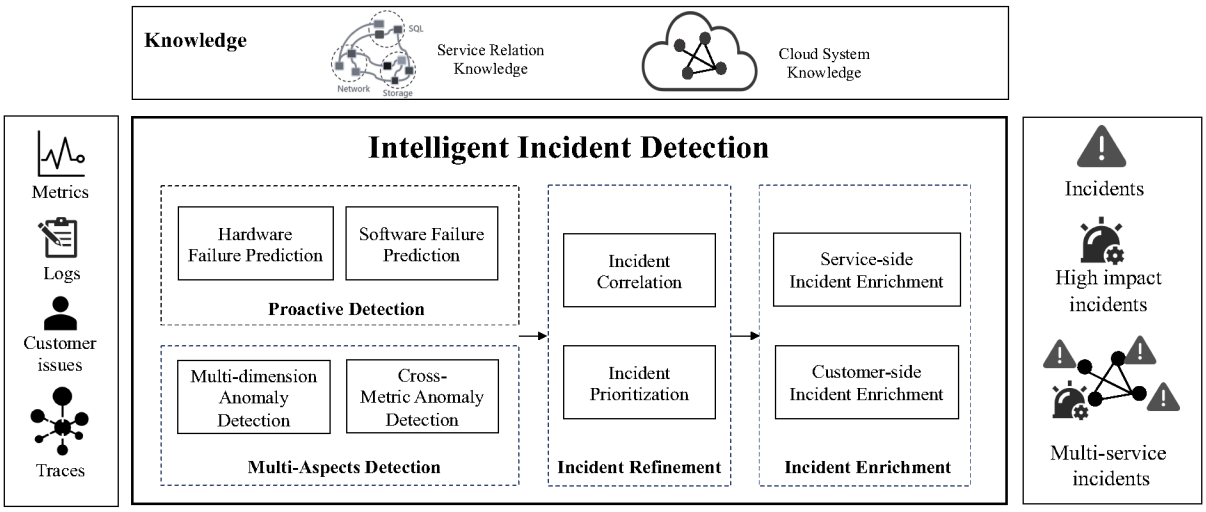

Figure 3. Intelligent incident detection framework.

before a high-impact incident happens. There are enormous incidents in a complex cloud system with thousands of services. In *incident refinement*, the goal is to reduce the flooding incidents by correlating incidents from multiple services in a global view and ranking incidents with priority. Most incidents are created based on the metric signals and need more information for the follow-up fault localization and diagnosis. To facilitate the incident-resolving process, it is important to enrich the incident with clues shortly after its creation. In *incident enrichment*, we propose to enrich the incident with logs and customer issues, which indicate the internal and external information, respectively. Overall, *proactive and multi-aspect detection* enable timely and accurate incident detection, and *incident refinement and enrichment* help us achieve the comprehensiveness goal.

4.2 Multi-Aspect Detection

As introduced in Section 3.1, it is labor-intensive and error-prone to manually configure the complicated alerting logic, including determining the detection scope of the monitored subjects and applying detection models to the monitoring metrics. Therefore, we aim to automatically identify the combinations of the monitored subjects and achieve common knowledge sharing among different monitoring metrics. On the one hand, we proposed iDice [20], an incident detection approach based on multi-dimensional time series, to identify the effective combinations whose monitoring metrics exhibit significant abnormal behaviours. Further, we proposed MID (Multi-dimensional Incident Detection) [13], which leverages the idea from the meta-heuristic search technology to improve the effectiveness and efficiency in combination search process, which is more practical in large-scale cloud systems. On the other hand, to bridge the incident detection modeling among various monitoring metrics, we proposed the ATAD (Active Transfer Anomaly Detection) method [27]. It is able to transfer the knowledge learned from the source monitoring signals to another target one through transfer learning and make the model fast adapt to the new metric data utilizing active learning.

4.3 Proactive Detection

As mentioned in Section 3.4, current reactive incident detection has its limitation - it can only detect after failures have happened when impacts are already made on services. To this end, we have continuously been driving the paradigm shift from reactive incident detection to proactive incident prediction. We have observed that the cloud software and hardware usually do not fail suddenly [18, 25]. There are pieces of evidence that failures are likely to happen. If we could extract the cloud system software and hardware behavior pattern, we can predict cloud incidents. Successful incident prediction enables us to take actions in advance to avoid enlarged impacts.

We have worked on predicting hardware failures and software failures. For hardware, we have first predicted node failure in cloud service systems. This is a challenging task, as a node failure could be caused by a variety of reasons and the failure data are highly imbalanced (i.e., far more normal nodes than failure nodes). We have proposed MING [19] to tackle the challenges. Then we investigated disk failures as a common type of hardware failures. We utilized both disk-level sensor (SMART) data as well as system-level signals (e.g., PagingError, FileSystemError) to design our models [25]. A step further, we proposed NTAM [21] not only utilizing a disk's own status data, but also capturing its neighbors' status data and the temporal nature of the disk status data. For software, we have collected cloud system-wide alerting signals from different services and detect the occurrence of

multi-service high-impact incidents from a global perspective [7, 18]. We generalized the failure prediction approaches as a framework [30]. We have also proposed proactive mitigation actions for the predicted failures. For example, we proposed a reinforcement learning strategy that takes proactive actions from deprioritizing failure nodes, avoiding failure nodes, live migration, moving works to healthy nodes, soft reboot, human investigation, and a mixture of them [17]. We have promoted this proactive paradigm as a prediction-guided design for system [22].

4.4 Incident Refinement

To resolve the challenge of flooding incidents, we propose to 1) provide a global view of incidents by linking related incidents together; 2) prioritize high-impact incidents out of all incidents. In this way, we can greatly reduce the number of incidents, facilitate the incident resolution and make engineers' efforts more focused.

First, we link incidents that are caused by the same underlying issue together. The backbone of incident linking is the historical incident relation data that are labeled by engineers, based on which we could learn the service dependency knowledge. We propose LiDAR [6], a framework to leverage both textual information and components inter-dependency information to calculate the linking probability of two incidents and COT [24], which can further correlate and cluster a set of related incidents among services. With LiDAR and COT in hand, we can provide suggestions to engineers about which set of incidents are likely to be correlated and should be analyzed jointly.

Second, to figure out the importance and urgency of incidents, we first conducted an empirical study on the characteristics of incidental incidents. Based on the empirical results, DeepIP [5] was proposed to prioritize incidents with an advanced deep learning model. The method takes inputs of three types: incident description, key terms (such as API names), and runtime environment information (such as incident-occurring device) to predict the likelihood of incidents to be incidental. The incidents with a low likelihood of being incidental are suggested to engineers for resolving with a high priority.

4.5 Incident Enrichment

As discussed in Section 3.2, systematic and efficient incident enrichment can greatly contribute to the incident fixing process. Therefore, we focus on systematically and precisely locate the fault scope, as well as accurately and efficiently identify service-side console logs or customer-side support tickets as supplementary information for the target incident.

For fault localization in hierarchical structure, we proposed HALO [26] to localize the fault to a proper granularity, which usually suffers from improper aggregation level of incidents. HALO can learn the hierarchical relationship among attributes and leverages the hierarchy structure, and is able to locate the exact failure part precisely and efficiently.

For service-side log correlation, we proposed Onion [29] to localize the incident-indicating logs, which are supposed to be widely distributed on the anomalous components but rare on the normal ones, as shown in Figure 2. Onion clusters logs into different groups based on their text, where the contrast analysis is then performed. Onion can accurately find out a few lines of related logs from millions of raw log data in only several minutes.

For linking incidents with customer support tickets, we proposed LinkCM [14], which can automatically link customer reported tickets with the target incident. The above link prediction problem can be formulated as a binary classification task. LinkCM incorporates an attention-based deep learning classifier, which is capable of capturing the semantic meaning from the customers' issue descriptions written in natural language.

5 Deployment and Case Studies

AIOps has long been adopted in Microsoft incident management and shown its effectiveness. Collaborating with partner team, we have designed and deployed an AIOps-oriented incident management project called BRAIN, empowering real-world incidents in cloud services [8, 9]. An intelligent incident detection framework is a core part of the AIOps incident management system. BRAIN intelligent detection features have been continuously deployed and improved service incident detection significantly. We show two real world cases to demonstrate the application and benefits of intelligent incident detection.

Belated Update of Domain Name Service: This case is an incident caused by a failure in the propagation of the decommission state of an account server group to the downstream dependent services. The domain name service failed to update the state of the account server group of region X (we concealed the name for privacy protection), which had been decommissioned at an earlier time. This results in resolution failures when the requests are routed to these account server groups. Before our intelligent incident detection approaches, there would be several monitors reporting incidents for this problem as multiple services had dependencies on these servers and got impacted. The OCEs from related services would treat the incidents as low severity since there is not much impact on each service at first glance. There were multiple reasons behind it. Without proper multi-dimensional detection, the fault scope was not that precise. Moreover, without proper refinement, especially incident correlation, OCEs from multiple services worked independently and underestimated the incident impact.

Thanks to our intelligent approaches, we reported the incidents accurately and comprehensively. With multi-dimensional anomaly detector, we gave the fine-grained failure

scope. With clearer scoping, we further refined and correlated the incidents according to system and service topology. Then we knew the impacts on multiple services, such as file sharing and mail exchange processes. Moreover, our system identified the account server group shared among the impacted scopes, providing a valuable hint for further diagnosis. Finally, the OCEs were called up to engage and resolve this incident. In summary, our system is able to detect alerts more accurately and timely, also with more enriched information. Owing to this, engineers are capable of taking corresponding actions before the downstream services are severely impacted.

Datacenter Power Failure: In this case, the region of the datacenter was hit by an ice storm, resulting in a large-scale power failure. Counterintuitively, not all incidents caused by power issues are easy to detect. Few power issues had led to real incidents due to resource redundancy. The affected racks lost power when their supporting UPS (Uninterruptible Power Supply) units drained. It took tens of minutes before running out of power supply. Multiple services got affected gradually during this incident. Before our intelligent incident detection approaches, hardware and software were managed independently due to layers of abstraction. At that moment, both service teams and datacenter management teams would receive incidents, but the teams might not know the whole picture at first glance.

Thanks to our intelligent approaches, we reported the high-impact multi-service incident timely and comprehensively. Our intelligent detection system sent out an initial notification upon receiving alerting signals from the Compute service. This immediately attracted the attention of OCEs. Right after that, our system identified more than ten alerting signals from other services. Our intelligent detection approach correlated the signals and set the proper severity level of the incident. Our approach took more and more signals from the downstream and upstream dependent services into consideration. Not only the inference confidence of the intelligent model was raised substantially, but also more OCEs from multiple teams were engaged, including service teams and datacenter management teams. They were on the same page, and a bridge meeting was set up for collaborative problem-solving. In summary, our system is able to detect this high-impact incident more timely, accurately and comprehensively. Owing to this, engineers from different teams are able to understand the real impact on multiple services and work together in an early time before the incident gets worse.

6 Conclusion

Incident detection has become much faster and more accurate in modern cloud. However, some critical incidents still occur in an unexpected manner and thus require intensive engineering effort. This paper provides a comprehensive view of industrial cloud incident detection from the provider's perspective. We summarize our recent intelligent incident detection practices and the corresponding real-world challenges at Microsoft. We also introduce our intelligent framework for timely, accurate, and comprehensive incident detection and demonstrate its practical benefits conveyed to the large-scale Microsoft cloud services. We believe our research work could shed light on future research and engineering efforts towards intelligent incident management.

Acknowledgements

We thank our colleagues at Microsoft Azure and Microsoft 365: John Sheehan, Murali Chintalapati, Andrew Zhou, Victor Rühle, Chetan Bansal, Jeffrey Sun, Bing Hu, Sheila Jiang, Jian Zhang, Youjiang Wu, Randolph Yao, Ze Li, Rakesh Namineni, Mohit Verma, David Wilson, Botao Zhang, Paul Wang, Wenzhong Jiang, Hua Ding, Yining Wei as well as our academic partners: Hongyu Zhang, Shaowei Cai, Yangfan Zhou, for their kind help and support in this work. Yichen Li's work was supported in part by the Key-Area Research and Development Program of Guangdong Province (No. 2020B010165002) and the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK14210920).

References

[1] 2019. Federal Cloud Computing Strategy. https://cloud.cio.gov/.

[2] 2022. AWS Post-Event Summaries. https://aws.amazon.com/cn/premiumsupport/technology/pes/.

[3] 2022. Azure status history. https://status.azure.com/en-us/status/history/.

[4] 2022. Google Cloud Status Dashboard. https://status.cloud.google.com/summary.

[5] Junjie Chen, Shu Zhang, Xiaoting He, Qingwei Lin, Hongyu Zhang, Dan Hao, Yu Kang, Feng Gao, Zhangwei Xu, Yingnong Dang, et al. 2020. How incidental are the incidents? characterizing and prioritizing incidents for large-scale online service systems. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. 373–384.

[6] Yujun Chen, Xian Yang, Hang Dong, Xiaoting He, Hongyu Zhang, Qingwei Lin, Junjie Chen, Pu Zhao, Yu Kang, Feng Gao, et al. 2020. Identifying linked incidents in large-scale online service systems. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 304–314.

[7] Yujun Chen, Xian Yang, Qingwei Lin, Hongyu Zhang, Feng Gao, Zhangwei Xu, Yingnong Dang, Dongmei Zhang, Hang Dong, Yong Xu, et al. 2019. Outage prediction and diagnosis for cloud service systems. In *The World Wide Web Conference (WWW)*. 2659–2665.

[8] Zhuangbin Chen, Yu Kang, Feng Gao, Li Yang, Jeffrey Sun, Zhangwei Xu, Pu Zhao, Bo Qiao, Liqun Li, Xu Zhang, et al. 2020. Aiops innovations of incident management for cloud services. (2020).

[9] Zhuangbin Chen, Yu Kang, Liqun Li, Xu Zhang, Hongyu Zhang, Hui Xu, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, Yingnong Dang, Feng Gao, Pu Zhao, Bo Qiao, Qingwei Lin, Dongmei Zhang, and Michael R. Lyu. 2020. *Towards Intelligent Incident Management: Why We Need It and How We Make It.* Association for Computing Machinery, New York, NY, USA, 1487–1497.

[10] Domenico Cotroneo, Luigi De Simone, Pietro Liguori, Roberto Natella, and Nematollah Bidokhti. 2019. How bad can a bug get? an empirical

analysis of software failures in the openstack cloud computing platform. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 200–211.

[11] Yingnong Dang, Qingwei Lin, and Peng Huang. 2019. AIOps: real-world challenges and research innovations. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Companion Proceedings (ICSE-Companion)*. IEEE, 4–5.

[12] Peter Garraghan, Renyu Yang, Zhenyu Wen, Alexander Romanovsky, Jie Xu, Rajkumar Buyya, and Rajiv Ranjan. 2018. Emergent failures: Rethinking cloud reliability at scale. *IEEE Cloud Computing* 5, 5 (2018).

[13] Jiazhen Gu, Chuan Luo, Si Qin, Bo Qiao, Qingwei Lin, Hongyu Zhang, Ze Li, Yingnong Dang, Shaowei Cai, Wei Wu, et al. 2020. Efficient incident identification from multi-dimensional issue reports via meta-heuristic search. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 292–303.

[14] Jiazhen Gu, Jiaqi Wen, Zijian Wang, Pu Zhao, Chuan Luo, Yu Kang, Yangfan Zhou, Li Yang, Jeffrey Sun, Zhangwei Xu, et al. 2020. Efficient customer incident triage via linking with system incidents. In *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 1296–1307.

[15] Shilin He, Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, Michael R Lyu, and Dongmei Zhang. 2018. Identifying impactful service system problems via log analysis. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 60–70.

[16] Patrik Hummel, Matthias Braun, Max Tretter, and Peter Dabrock. 2021. Data sovereignty: A review. *Big Data & Society* 8, 1 (2021).

[17] Sebastien Levy, Randolph Yao, Youjiang Wu, Yingnong Dang, Peng Huang, Zheng Mu, Pu Zhao, Tarun Ramani, Naga Govindaraju, Xukun Li, Qingwei Lin, Gil Lapid Shafriri, and Murali Chintalapati. 2020. Predictive and Adaptive Failure Mitigation to Avert Production Cloud VM Interruptions. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI)*. 1155–1170.

[18] Liqun Li, Xu Zhang, Xin Zhao, Hongyu Zhang, Yu Kang, Pu Zhao, Bo Qiao, Shilin He, Pochian Lee, Jeffrey Sun, et al. 2021. Fighting the Fog of War: Automated Incident Detection for Cloud Systems. In *2021 USENIX Annual Technical Conference (USENIX ATC 21)*. 131–146.

[19] Qingwei Lin, Ken Hsieh, Yingnong Dang, Hongyu Zhang, Kaixin Sui, Yong Xu, Jian-Guang Lou, Chenggang Li, Youjiang Wu, Randolph Yao, Murali Chintalapati, and Dongmei Zhang. 2018. Predicting Node Failure in Cloud Service Systems. In *Proceedings of the 2018 26th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 480–490.

[20] Qingwei Lin, Jian-Guang Lou, Hongyu Zhang, and Dongmei Zhang. 2016. iDice: problem identification for emerging issues. In *Proceedings of the 38th International Conference on Software Engineering (ICSE)*. 214–224.

[21] Chuan Luo, Pu Zhao, Bo Qiao, Youjiang Wu, Hongyu Zhang, Wei Wu, Weihai Lu, Yingnong Dang, Saravanakumar Rajmohan, Qingwei Lin, and Dongmei Zhang. 2021. NTAM: Neighborhood-Temporal Attention Model for Disk Failure Prediction in Cloud Platforms. In *Proceedings of the Web Conference 2021*. 1181–1191.

[22] Si Qin, Yong Xu, Shandan Zhou, Qingwei Lin, Hongyu Zhang, Saurabh Agarwal, Karthikeyan Subramanian, Eli Cortez, John Miller, Chris Cowdery, et al. 2020. Prediction-Guided Design for Software Systems. (2020).

[23] Weijing Wang, Junjie Chen, Lin Yang, Hongyu Zhang, Pu Zhao, Bo Qiao, Yu Kang, Qingwei Lin, Saravanakumar Rajmohan, Feng Gao, Zhangwei Xu, Yingnong Dang, and Dongmei Zhang. 2021. How Long Will it Take to Mitigate this Incident for Online Service Systems?. In *2021 IEEE 32nd International Symposium on Software Reliability Engineering (ISSRE)*. 36–46.

[24] Yaohui Wang, Guozheng Li, Zijian Wang, Yu Kang, Yangfan Zhou, Hongyu Zhang, Feng Gao, Jeffrey Sun, Li Yang, Pochian Lee, et al. 2021. Fast outage analysis of large-scale production clouds with service correlation mining. In *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*. IEEE, 885–896.

[25] Yong Xu, Kaixin Sui, Randolph Yao, Hongyu Zhang, Qingwei Lin, Yingnong Dang, Peng Li, Keceng Jiang, Wenchi Zhang, Jian-Guang Lou, Murali Chintalapati, and Dongmei Zhang. 2018. Improving Service Availability of Cloud Systems by Predicting Disk Error. In *2018 USENIX Annual Technical Conference (USENIX ATC)*. 481–494.

[26] Xu Zhang, Chao Du, Yifan Li, Yong Xu, Hongyu Zhang, Si Qin, Ze Li, Qingwei Lin, Yingnong Dang, Andrew Zhou, et al. 2021. HALO: Hierarchy-aware Fault Localization for Cloud Systems. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*. 3948–3958.

[27] Xu Zhang, Junghyun Kim, Qingwei Lin, Keunhak Lim, Shobhit O Kanaujia, Yong Xu, Kyle Jamieson, Aws Albarghouthi, Si Qin, Michael J Freedman, et al. 2019. Cross-dataset time series anomaly detection for cloud systems. In *2019 USENIX Annual Technical Conference (USENIX ATC)*. 1063–1076.

[28] Xu Zhang, Yong Xu, Qingwei Lin, Bo Qiao, Hongyu Zhang, Yingnong Dang, Chunyu Xie, Xinsheng Yang, Qian Cheng, Ze Li, et al. 2019. Robust log-based anomaly detection on unstable log data. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 807–817.

[29] Xu Zhang, Yong Xu, Si Qin, Shilin He, Bo Qiao, Ze Li, Hongyu Zhang, Xukun Li, Yingnong Dang, Qingwei Lin, et al. 2021. Onion: identifying incident-indicating logs for cloud systems. In *Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering (FSE)*. 1253–1263.

[30] Pu Zhao, Chuan Luo, Bo Qiao, Youjiang Wu, Yingnong Dang, Murali Chintalapati, Susy Yi, Paul Wang, Andrew Zhou, Saravanakumar Rajmohan, et al. 2021. F3: Fault Forecasting Framework for Cloud Systems. (2021).