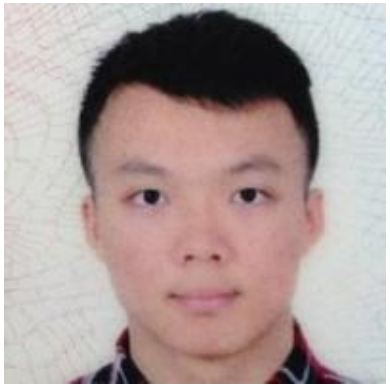
Understanding Exception-Related Bugs in Large-Scale Cloud Systems

ASE'19

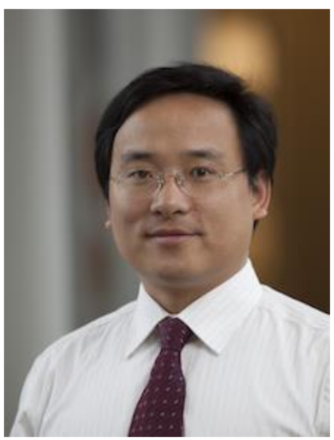
Haicheng Chen, Wensheng Dou
Yanyan Jiang,Feng Qin



[ASE'19] **Understanding Exception-Related Bugs in Large-Scale Cloud Systems [p**
Haicheng Chen, *Wensheng Dou, Yanyan Jiang, Feng Qin*

[FSE'16] **Crash Consistency Validation Made Easy**
Yanyan Jiang, ***Haicheng Chen****, Feng Qin, Chang Xu, Xiaoxing Ma, Jian Lu*

Haicheng Chen
Ohio State University



Feng Qin
Ohio State Universit

highly dependable and secure computer systems

2019

ASE

Understanding Exception-Related Bugs in Large-Scale Cloud Systems
Haicheng Chen, Wensheng Dou, Yanyan Jiang, and Feng Qin
In *Proceedings of the 34th IEEE/ACM International Conference on Automated Software En*

USENIX ATC

Lessons and Actions: What We Learned from 10K SSD-Related Storage System Failures
Erci Xu, Mai Zheng, Feng Qin, Yikang Xu, and Jiesheng Wu
In *Proceedings of the 2019 USENIX Annual Technical Conference,* Jul. 2019

2018

PDSW-DISCS

Understanding SSD Reliability in Large-Scale Cloud Systems
Erci Xu, Mai Zheng, Feng Qin, Yikang Xu, and Jiesheng Wu
In *Proceedings of the 3rd ACM/IEEE Joint International Workshop on Parallel Data Storag*
Nov. 2018

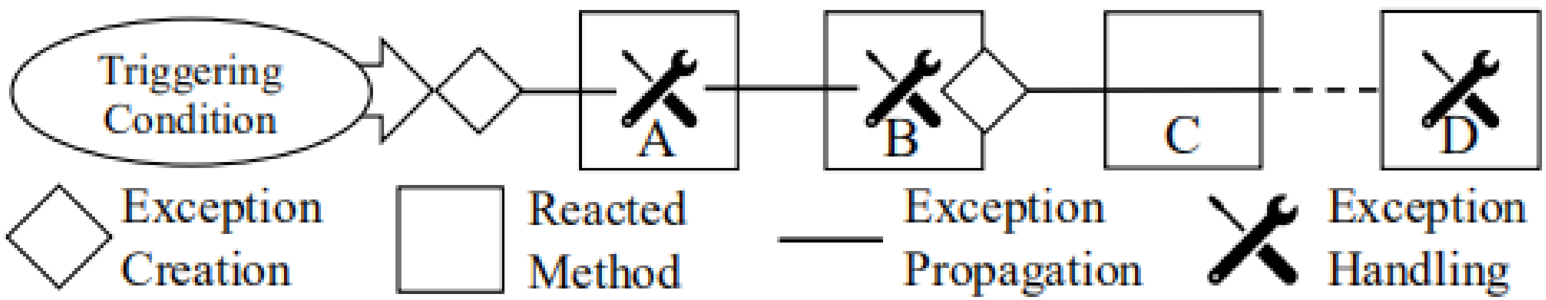
FSE
Distinguished Paper

An Empirical Study on Crash Recovery Bugs in Large-Scale Distributed Systems
Yu Gao, Wensheng Dou, Feng Qin, Chushu Gao, Dong Wang, Jun Wei, Ruirui Huang, Li Z
In *Proceedings of the 26th ACM Joint European Software Engineering Conference and Sym*

Motivation

- Exception mechanism is widely used in cloud systems
  - 7% of the source code in twelve popular open source distributed systems
- The sophisticated logic of cloud systems hinder use of exception mechanism
- Mistakes in the exception mechanism use may lead to severe consequences(eBugs)
  - system downtime，data loss......

Exception Mechanism




```
void B(...) throws OtherException {
  try { A(...);
  } catch (SomeException e) {
    someHandling(...);
    throw new OtherException(e);
  } }
```

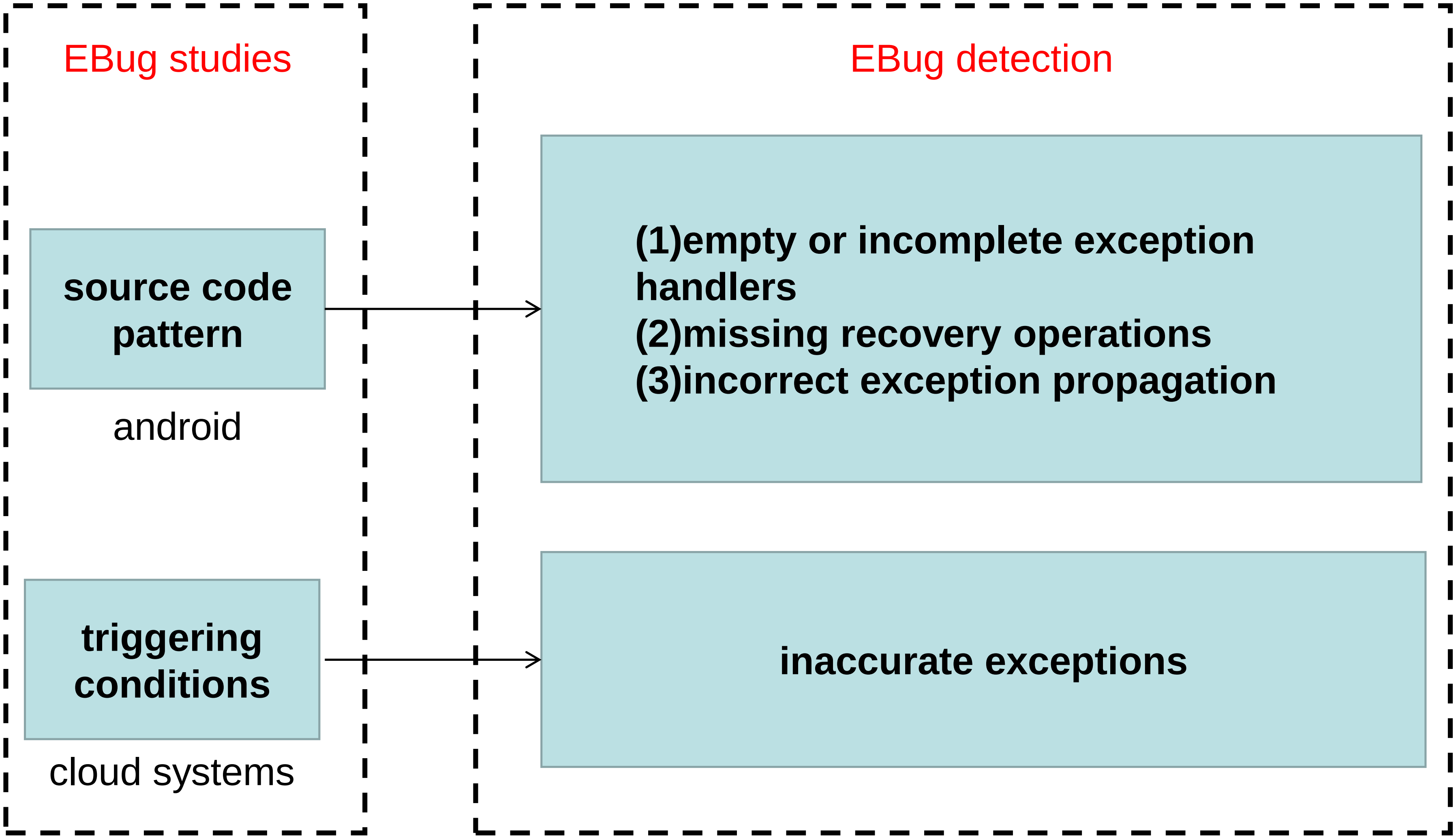
Contributions

- Present the first comprehensive study on eBugs from the perspective of triggering conditions

- Build a static analysis tool, called DIET, and evaluate it using the latest versions of the studied systems

- Provide a large benchmark of eBugs in cloud systems

Related Work



Study on eBugs

- Data source: JIRA

System	Cassandra	HBase	HDFS	MR	YARN	ZK	Total
Retrieved	1,336	1,576	763	460	457	210	**4,802**
Studied	40	92	31	16	23	8	**210**

- eBug Analysis
  - RQ1: Triggering conditions of eBugs
  - RQ2: Relation between the triggering conditions and the root causes of eBugs
  - RQ3: The impacts of eBugs on cloud systems

Triggering Condition Types

Triggering Condition (# eBugs)		Scenario (# eBugs)	CA	HB	HF	MR	YN	ZK
Non-semantic condition (114)	Network error (46)	Premature disconnection (17)	0	8	5	1	1	2
		Local timeout (12)	2	5	2	0	3	0
		Connection refused (11)	1	8	0	1	1	0
		Other network errors (6)	0	6	0	0	0	0
	File system error (40)	File corrupted (23)	10	11	1	1	0	0
		File not found (13)	3	5	3	2	0	0
		Other file system errors (4)	2	0	2	0	0	0
	Out of resource (16)	Out of memory (5)	1	2	1	1	0	0
		Out of disk space (5)	1	0	1	1	1	1
		Port conflicted (3)	1	1	1	0	0	0
		Out of other resources (3)	0	0	3	0	0	0
	Untimely interrupt (12)	Thread interrupted when invoking a blocking method (12)	2	1	0	3	6	0
Semantic condition (96)	-	-	17	45	12	6	11	5
Total	**210**		**40**	**92**	**31**	**16**	**23**	**8**

Timing Requirements on Triggering Conditions

- **Weak**:occur at any consistent global state
  - ZOOKEEPER-2757 can be triggered whenever a user issues a delete command with an invalid pathname
- **Moderate**:occur on a node when it is in certain states
- **Strong**:occur on a node when both the current node and other nodes are in certain states

Timing Requirements on Triggering Conditions

Condition Type	Timing Requirement		
	Weak	Moderate	Strong
Network error	9	36	1
File system error	32	5	3
Out of resource	5	11	0
Untimely interrupt	0	10	2
Semantic condition	39	34	23
Total	**85**	**96**	**29**

Most (86%) of the eBugs do not have strong timing requirements on their triggering conditions.

Root Causes

- Inaccurate Exception
  - creating an inaccurate exception
- Missing Reaction
  - neither catching nor specifying an exception in the method signature
- Overly-General Reaction
  - different exceptions are incorrectly handled in the same way
- Incorrect Reaction Logic
  - a handler is incorrect for all the exceptions it handles

Root Causes

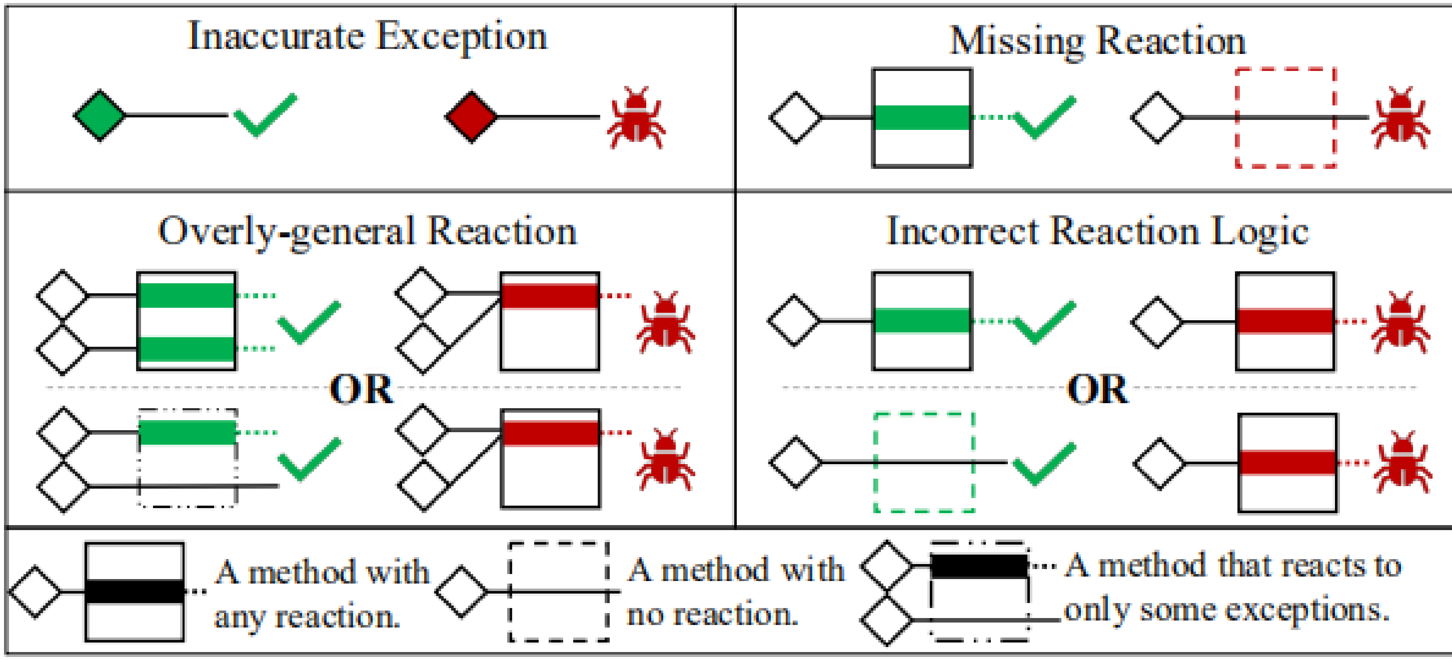


Fig. 3. Four different types of eBug root causes. For each type, we show the correct version on the left (green) and the buggy version on the right (red).

Root Causes

Root Cause	eBug #	CA	HB	HF	MR	YN	ZK
Inaccurate exception	21	3	8	4	3	3	0
Missing reaction	36	12	11	3	3	4	3
Overly-general reaction	87	13	42	14	6	11	1
Incorrect reaction logic	66	12	31	10	4	5	4
Total	**210**	**40**	**92**	**31**	**16**	**23**	**8**

Relation between triggering conditions and root causes

- Inaccurate Exception

Type	Wrong Class	Wrong Message	Lacking Cause	Total
eBug #	13	5	3	**21**

```
 void updateMetaLocation() throws IOException {
   if (waitForRootServerConnection() == null)
-    throw new NullPointerException(...);
+    throw new IOException(...);
 }
 void process() {
   try { updateMetaLocation();
   } catch (IOException e) { cleanup(); }
 }
```

In half of the eBugs that create a totally misleading exception, the exception class is inconsistent with its triggering condition.

Relation between triggering conditions and root causes

- Overly-General Reaction

Relation	Same Type	Different Types	Unknown	**Total**
eBug #	48	30	9	**87**

  - In many (34%) overly-general reaction eBugs, the incorrectly reacted exception and the correctly reacted ones are caused by different types of tiggering conditions.

Bug Impacts

Symptom	**eBug #**
Node downtime	48
Incorrect error message	44
Data loss or potential data loss	31
Hang or performance downgrading	26
Resource leak/exhaustion	10
Operation failure†	51
Total	**210**

Priority	Blocker	Critical	Major	Minor	Trivial	**Total**
eBug #	21	42	110	33	4	**210**

Detecting eBugs in Cloud Systems

- Inaccurate exceptions
  - checking if the exceptions are consistent with their triggering conditions.

- Overly-general reactions
  - checking if the handled exceptions are triggered by different condition types.

DIET: Detecting Inaccurate Exceptions

the root exception and triggering conditions

$$P_{c,t} = \frac{\text{number of } (c_i, t_j) \text{ where } c_i = c, t_i = t}{\text{number of } (c_i, t_i) \text{ where } c_i = c} \qquad (1)$$

error message triggering conditions

$$P_{w,t} = \frac{\text{number of } (w_{i,j}, t_i) \text{ where } w_{i,j} = w, t_i = t}{\text{number of } (w_{i,j}, t_i) \text{ where } w_{i,j} = w} \qquad (2)$$

$$P_{m,t} = \frac{\sum_{w_i \in m} P_{w_i,t}}{n}$$

Detect

$$P_{\mathit{same\text{-}type}} = \sum_{t \,\in\, \text{Five types}} \min(P_{c,t}, P_{m,t}) \qquad P_{\mathit{same\text{-}type}} \leq 0.2$$

DIET: Detecting Inaccurate Exceptions

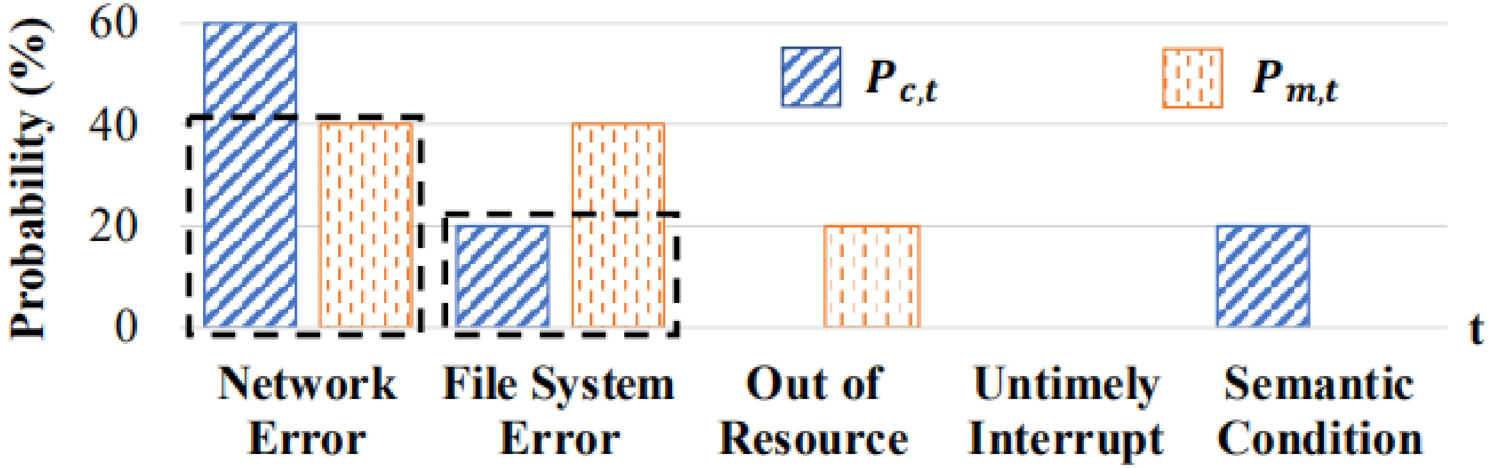


Fig. 8. The $P_{c,t}$ and $P_{m,t}$ of an exception. For example, when t is network error, $P_{c,t}$ is 60% and $P_{m,t}$ is 40%. The overlapping areas are highlighted with the dashed boxes. The total overlap is 60%.

Result

APPLYING DIET ON REAL-WORLD CLOUD SYSTEMS

System (Version)	Cassandra (3.11)	Hadoop[†] (3.1.2)	HBase (2.1.4)	ZooKeeper (2.4.14)	Total
`Throw`	2,823	9,853	5,020	429	**18,125**
Root ex.	1,282	3,090	1,374	159	**5,905**
Calculated	550	1,579	716	84	**2,929**
Reported	100	136	73	5	**314**
Candidate	9	20	2	0	**31**