An Empirical Study on Performance Bugs for Highly Configurable Software Systems

Xue Han and Tingting Yu
Presented by: Geanderson E. dos Santos

Universidade Federal de Minas Gerais
Instituto de Ciências Exatas

Software Reuse

Introduction

- Modern computer systems are highly-configurable, complicating the testing and debugging process
- Performance bugs are difficult to expose, primarily because detecting them requires specific inputs
- Authors analyzed 113 real-world performance bugs, randomly sampled from three highly-configurable open-source projects: Apache, MySQL and Firefox.
- They found out a set of lessons learned and guidance to aid practitioners and researchers to better handle performance bugs in highly-configurable software systems

Introduction

- Performance bugs occur due to the complexity of the configuration space and the sophisticated constraints
- They differ from functional bugs because these bugs can cause significant performance degradation, leading to poor user experience, long response time, and low system throughput
- Functional bugs typically lead to only system crashes or incorrect results

Introduction

- The main challenging in performance bugs is the fact typical coverage based test are mostly not able to handle them
- Furthermore many different configurations could be applied making testing even more difficult
- For instance, the Apache open source project has over 1000 different configuration options

Motivation

1. Do performance bugs require specific configurations to manifest?
2. Are configuration-related performance bugs different from configuration-related general bugs for testing, debugging, and fixing?
3. Do developers need special tools to handle performance bugs?

Motivation

- **Wasted loop computation**: Triggered on Apache when users set the *Start-Servers* with a large number
- **Page not cleaned**: In MySQL, the root cause of this bug is due to the wrong implementation of the page cleaner feature
- **Lock contention**: When an option in MySQL is set to 2, both logging functions (commit and write) use the *mutex* lock to write to a buffer
- **Cache not purged**: In Apache, the *LDAPCacheEntries* is too small then the cache entries will not get purged
- **Abnormal termination**: In Apache, configuring multiple listening ports causes the server to hang during a restart
- **Slow autocomplete feature**: Firefox autocomplete becomes extremely slow. The bug is due to misconfigurations on user preferences

Motivation

1. Do performance bugs require specific configurations to manifest?
   - Performance bugs are triggered by not only inputs, but also specific configurations.
2. Are configuration-related performance bugs different from configuration-related general bugs for testing, debugging, and fixing?
   - These bugs have shown similarities to general configuration bugs.
3. Do developers need special tools to handle performance bugs?

   - Existing configuration testing or performance testing may not be effective to handle these bugs.

Research Questions

- RQ1. How prevalent are performance bugs related to configurations?
- RQ2. What are the types of configurations that can influence performance?
- RQ3. What are the causes of configuration-related performance bugs?
- RQ4. How complicated is it to fix configuration-related performance bugs?

Data Collection

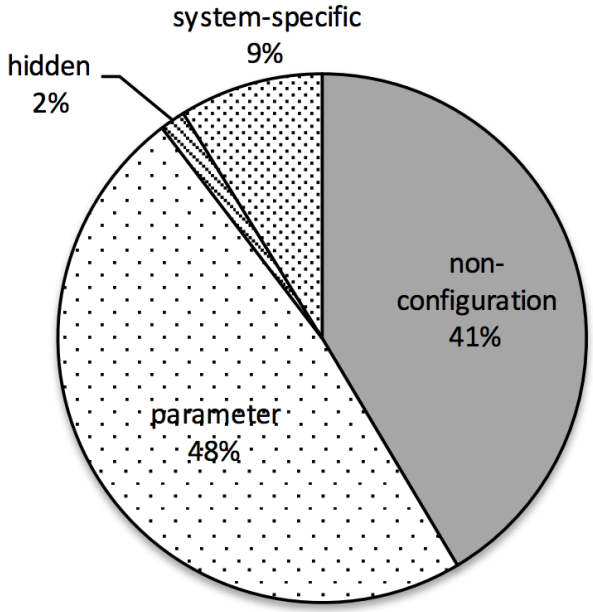
1. **Bugs**:
   - Source of bugs changelogs and bug repositories
     - Data mining of known words such as “slow” and “performance” found 887 bugs
     - Manual classification of performance bugs from the dataset
     - 193 performance bugs found where 113 are related to configuration
2. **Configurations**:
   - Study of the configuration space of each subject
     - Study done by inspecting the artifacts of the source code
     - Read documentation and user manual available publicly
     - Total number of options were 3700
     - Only 90 studied because they were common to all three subjects

Dataset

Application	Sampled Bugs	Used Bugs	Config Bugs
Apache Suite	323	63	60
MySQL Suite	241	77	41
Mozilla	323	53	12
Total	**887**	**193**	**113**

RQ1. How prevalent are performance bugs related to configurations?

- *Prevalence of Bugs*



- **Finding 1:** *A significant percentage of performance bugs are related to configuration issues (59%). Compared to general configuration bugs (27%), if these results generalize, performance bugs are more relevant to configurations*

RQ2. What are the types of configurations that can influence performance?

- *Configuration Types*

Project	Parameter	Hidden	Sys-spec	**Total**
Apache	50	1	9	**60**
MySQL	32	2	7	**41**
Mozilla	11	0	1	**12**

- **Finding 2:** *The parameter configurations account for a majority of the examined configurations (78% to 92%)*
- **Finding 3:** *However, the system-specific configurations account for a non-trivial portion (8% to 17%), and should be of particular concern for performance*

Parameter Configuration

- Refer to the configurations whose values can be changed by end users through configuration files
- As it was the type of configuration most found in the analyzes
- It was studied from three perspectives:
  - Types of parameters: Only considered legal options for configurations.
  - Number of incorrect parameters: Involved multiple options.
  - Problem domains of configurations: Domains classified in memory, network, I/O, concurrency, and graphics.

Parameter Configuration

- **Finding 4:** *All studied parameter configuration bugs results from legal options. The ratio of the number of bugs caused by legal configuration values over all the studied configuration performance bugs is much higher than the finding (46.3% - 61.9%) for general configuration bugs*

Parameter Configuration

Project	one option	two options	>2 options	**Total**
Apache	36	11	3	**50**
MySQL	23	7	2	**32**
Firefox	8	2	1	**11**

- **Finding 5:** *The majority (72% to 73%) of studied parameter configuration bugs is related to only one option, whereas about 27% to 28% of the examined parameter configuration bugs involve two and more options*

Parameter Configuration

Project	Memory	Network	I/O	Concurrency	Graphics
Apache	2	27	1	1	0
MySQL	12	12	8	4	0
Firefox	1	0	0	0	3
Total	**15**	**39**	**9**	**5**	**3**

- **Finding 6:** *The majority of the examined parameters fall into the Memory and Network domains. However, the distribution depends on the characteristics of applications (e.g., Firefox is more relevant to the graphics domain)*

Hidden Configuration

- Found only on the source code and related files
- All hidden configurations bugs are related to performance bugs
- They are very difficult to track down because they are written in different language (e.g., C++, Java and JavaScript)
- Only possible to track using Reverse Engineering

- **Finding 7:** *A small portion (3%) of the examined parameter configuration bugs involve hidden options*

System-level Configurations

Project	Environment	HW	Components
Apache	1	0	8
MySQL	1	2	4
Firefox	0	1	0
Total	**2**	**3**	**12**

- **Finding 8:** *There are various system-level misconfigurations that can cause performance bugs. The majority (70%) of the examined system-specific configurations involve incompatible components/libraries*

RQ3. What are the causes of configuration-related performance bugs?

Project	First-time	Runtime	Upgrade
Apache	60	0	4
MySQL	37	0	4
Firefox	11	12	2
Total	**108**	**12**	**10**

- **Finding 9:** *The majority 95% bugs are related to the first-time use. However, in applications that allow runtime reconfiguration, the performance bugs caused by runtime misconfiguration are non-negligible (11% in Firefox)*

RQ3. When are Bugs Introduced?

Project	Memory	CPU	Synchronization
Apache	24	11	1
MySQL	13	3	15
Firefox	2	5	0

- **Finding 10:** *A majority of performance bugs involves inefficient memory usage (up to 35%), and a significant portion of performance bugs are caused by high CPU utilization (up to 17%) and synchronization (up to 14%)*
- **Finding 11:** *Performance bugs are caused by a small subset of configuration options*

RQ4. How complicated is it to fix configuration-related performance bugs?

Project	Options	Source	OptionESource	**Total**
Apache	5	54	1	**60**
MySQL	2	39	0	**41**
Firefox	5	6	1	**12**

- Three way to fix performance bugs:
  - Changing configuration values
  - Patching source code
  - Fixing both configuration values and soource code
- Studies Suggest that simple patches are the best form to fix bugs

- **Finding 12:** *A dominate majority (88%) of performance bugs involve fixing the code*
- **Finding 13:** *Fixing configuration-related performance bugs is more complex than fixing general performance bugs*

Threats to Validity

Two main threats to the validity of this study:

- Other subjects may have different bugs and configurations
- Data recorded in bug tracking systems and code version histories can have a systematic bias relative to the full population of bug fixes and can be incomplete or incorrect
- The age of bug reports (they span over 10 years)

Implications for Practitioners

- Performance testing should consider key configurations
  - Even though configuration-aware testing may be expensive, it is a key factor for avoiding bugs in these systems
- System-level configurations are important
  - This implies that developers need to test their applications under different system settings that can closely resemble production
- Profiling is helpful for identifying misconfigutions
  - Provided insights about linking each configuration domain with its internal symptoms

Implications for Researches

- Testing and debugging tools are needed
  - One possibility is to leverage existing static analyses to identify performance-sensitive configuration options based on code patterns
- Configuration-aware regression testing is needed
  - As new bugs may be introduced, regressions test could perform re-validation of evolving software
- Building performance-influence configuration models
  - These black-box techniques can be improved if performance sensitive configuration options and their associated code elements can be better understood
- Fixing and avoiding performance bugs
  - A self-adaptive system should be able to reconfigure its settings to meet the performance requirements
- Extracting new configurations
  - We need new techniques that can both infer performance features from the code, and determine which features are useful as configuration options.

Conclusions

- The study covers a wide spectrum of characteristics, including types, causes, symptoms, and fixes.
- It also provides guidance for future research on performance testing and debugging.
- They intend to help developers and practitioners to extend and improve tools that can address performance issues in highly-configurable applications.
- As a future research: they will extend their study on more subject programs and propose techniques to handle these bugs

References

X. Han and T. Yu

An Empirical Study on Performance Bugs forHighly Configurable Software Systems.

Proceedings of the International International Symposium on Empirical Software Engineering and Measurement , 2016.